Contents

1 Reviewing Data Warehouse Basics
   Introductions 1-2
   Course Objectives 1-3
   Lessons 1-5
   Let’s Get Started 1-6
   Objectives 1-7
   Definition of a Data Warehouse 1-8
   An Alternate Definition 1-9
   The Basic Plan 1-10
   Basic Elements of the Data Warehouse 1-11
   Basic Form of the Data Warehouse 1-12
   Data Warehouse and OLTP Database Design Differences 1-13
   Data Warehouse Features 1-14
   Exploring Data Warehouse Characteristics 1-15
   Subject-Oriented 1-16
   Integrated 1-17
   Nonvolatile 1-18
   Time-Variant 1-19
   Oracle Data Warehouse Components 1-20
   Oracle’s Common Warehouse Model (CWM) 1-21
   Load from Many Sources 1-22
   Manage Data 1-23
   Decision Support System (DSS) 1-24
   Desirable Warehouse Functionality 1-25
   Data Warehousing Process 1-26
   Comparing Warehouses and Data Marts 1-27
   Flow of Data 1-28
   Enterprise Model Architecture 1-29
   Dependent Data Mart Model 1-30
   Independent Data Mart Model 1-31
   Development Models Compared 1-32
   Spiral Approach 1-34
   Data Warehouse Database Design Phases 1-36
   Summary 1-37
   Practices 1-1 Overview 1-38

2 Defining the Business and Logical Models
   Objectives 2-2
   Designing the Conceptual Model 2-3
   Design Challenges 2-4
   Business Modeling 2-5
   Performing Strategic Analysis 2-6
   Doing Your Homework 2-7
   Recognizing Critical Success Factors 2-8
   Understanding Business Imperatives 2-9
Making the Right Choice  4-15
Symmetric Multiprocessing (SMP)  4-16
SMP  4-17
Nonuniform Memory Access (NUMA)  4-18
NUMA  4-19
Clusters  4-20
Massively Parallel Processing (MPP)  4-22
MPP nCube Arrangements  4-23
MPP Benefits  4-24
MPP Limitations  4-25
Windows NT  4-26
Architectural Tiers  4-27
Middleware  4-28
Database Server Requirements  4-29
Parallelism  4-30
Further Considerations  4-31
Server Environments  4-32
Parallel Processing  4-33
Parallel Database  4-34
Parallel Query  4-35
Parallel Load  4-36
Parallel Processing  4-37
Summary  4-39
Practice 4-1 Overview  4-40
Practice 4-2 Overview  4-42
Practice 4-3 Overview  4-45

5 Storage Considerations for the Physical Model
Objectives  5-2
Database Sizing  5-3
Estimating the Database Size  5-4
Validating Database Size Assumptions  5-5
Example: Estimating the Database Size  5-6
Applying the Test Load Sampling  5-7
Test Load Sampling Using the Server  5-8
Indexes  5-9
Indexing Types  5-10
Indexing Strategies  5-12
B*tree Index  5-13
Bitmap Indexes  5-14
Bitmap Index Example  5-15
Data Placement  5-17
Oracle Tablespaces  5-18
Physical Database Structure  5-19
Avoiding Fragmentation  5-20
Creating Objects in a Tablespace  5-21
Tablespace Features  5-22
Transportable Tablespace Features  5-23
Changing the Tablespace Size  5-24
Other Factors Impacting Size  5-25
Partitioning Tables and Indexes  5-27
Advantages of Partitioning  5-28
Methods of Partitioning  5-29
Horizontal Partitioning  5-30
Vertical Partitioning  5-31
Range Partitioning  5-32
Hash Partitioning  5-33
Composite Partitioning in Oracle8i  5-34
List Partitioning  5-35
Summary  5-36
Practice 5-1 Overview  5-37

6 Strategies for Extracting, Transforming, and Transporting
Objectives  6-2
Extraction, Transformation, and Transportation Processes (ETT)  6-3
Data Staging Area  6-4
Preferred Traditional Staging Model  6-5
Extracting Data  6-6
Examining Source Systems  6-7
Mapping  6-10
Designing Extraction Processes  6-11
Importance of Data Quality  6-12
Benefits of Data Quality  6-13
Guidelines for Data Quality  6-14
Transformation  6-15
Transformation Routines  6-16
Why Transform?  6-17
Integration (Match and Merge)  6-21
Transformation Techniques  6-22
Where to Transform?  6-27
When to Transform?  6-28
Designing Transformation Processes  6-29
Transporting Data into the Warehouse  6-30
Extract versus Warehouse Processing Environment  6-31
First-Time Load  6-32
Refresh  6-33
Building the Transportation Process  6-34
7 Summary Management
   Objectives  7-2
   What Is a Summary?  7-3
   Implementing Summaries  7-4
   Summary Navigation  7-5
   Managing Historical Summary Data in the Warehouse  7-6
   What is Summary Management?  7-7
   Summary Management  7-8
   Materialized Views  7-9
   Why Use Materialized Views?  7-10
   Guidelines for Using Materialized Views  7-11
   Creating a Materialized View Summary  7-12
   Query Rewrite  7-13
   Oracle Optimization Methods  7-14
   Cost-Based Optimizer  7-15
   Star Query  7-16
   Star Query Optimization  7-17
   Star Join Optimization Implementation  7-18
   Oracle9i Bitmap Join Index  7-19
   Dimensions  7-20
   Dimensions and Hierarchies  7-21
   Dimension Example  7-22
   Defining Dimensions and Hierarchies  7-23
   Dimensions with Multiple Hierarchies  7-24
   Rewrites Using Dimensions  7-25
   Summary Advisor  7-26
   Summary  7-27
   Practice 7-1 Overview  7-28

8 Analytical Capabilities
   Objectives  8-2
   What Is Business Intelligence?  8-3
   Evolution to BI  8-4
   Categories of Business Intelligence Tools  8-5
Reviewing Data Warehouse Basics

ILT Schedule:

<table>
<thead>
<tr>
<th>Timing</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>55 minutes</td>
<td>Lecture</td>
</tr>
<tr>
<td>15 minutes</td>
<td>Practice</td>
</tr>
<tr>
<td>70 minutes</td>
<td>Total</td>
</tr>
</tbody>
</table>

Instructor Note

This lesson is intended to be a quick review of data warehousing. Students who have already taken *Data Warehouse Fundamentals*, the prerequisite course, should recognize the topics reviewed in this lesson.
Introductions

Tell us about yourself

• What is your name and company?
• What is your role in the organization?
• What is your level of Oracle expertise?
• What is your database design expertise?
• What is your data warehouse experience?
• Why are you considering building a data warehouse?
• What are your expectations for this class?

Instructor Note

Ask each student to introduce himself or herself and answer the following questions:

• What is your name and company?
• What is your role in your organization?
• What is your level of Oracle expertise?
• What is your database design expertise?
• What is your data warehouse experience?
• Why are you building a data warehouse or data mart?
• What do you hope to get out of this class?

As the students introduce themselves, assess their level of database design background and data warehouse knowledge. Adjust the material appropriately in the later lessons, especially the lesson “Reviewing the Data Warehouse Basics.”
Course Objectives

After completing this course, you should be able to do the following:

• Define the process of designing a data warehouse database model
• Describe the role of metadata in data warehouse design and strategies used to define and maintain metadata
• Explain the central concepts of dimensional data models
• Analyze and transform business requirements into a business model
Course Objectives

- Use entity relationship diagrams to transform the business model into a dimensional model
- Transform the dimensional model into a physical data design
- Evaluate summaries and understand the value of materialized views
- Understand multidimensional query concepts
Lessons

1. Reviewing Data Warehouse Basics
2. Defining the Business and Logical Models
3. Creating the Dimensional Model
4. Creating the Physical Model
5. Storage Considerations for the Physical Model
6. Strategies for Extracting, Transforming, and Transporting
7. Summary Management
8. Analytical Capabilities
Let’s Get Started
Objectives

After completing this lesson, you should be able to do the following:

• Describe the features and characteristics of a data warehouse

• Identify the components and functionality of an Oracle data warehouse environment using the Common Warehouse Model

• Identify the three types of data warehouse models

• List the major tasks of data warehouse database design

Lesson Aim

This lesson is a high-level review of some of the basic data warehousing concepts.
Definition of a Data Warehouse

“A data warehouse is a subject-oriented, integrated, nonvolatile, time-variant collection of data designed to support management DSS needs.”

- Bill Inmon

Instructor Note
The major points of this definition are discussed in more detail on the following page; therefore, save your detailed discussion for the next page.
An Alternate Definition

Oracle’s Data Warehouse Definition:
“An enterprise-structured repository of subject-oriented, time-variant, historical data used for information retrieval and decision support. The data warehouse stores atomic and summary data.”

Oracle’s Data Warehouse Method
The Oracle definition of a data warehouse is a strategic collection of all types of data in support of the decision-making process at all levels of an enterprise.
• All types of data infers text, image, video, audio, and even spatial data.
• Decision-Making means that the data warehouse is intended for knowledge workers or those people who must analyze information provided by the warehouse to make business decisions. A warehouse is not intended for day-to-day transaction processing. Knowledge workers must access information to plan, forecast, and make financial decisions. They are often people who are reasonably authoritative or are in influential positions, such as financial controllers, business analysts, or department managers.
• All levels of the enterprise means anyone within the business organization, not just those making decisions. This empowers all workers.

Instructor Note
Point out that Oracle’s definition differs in that it does not specify integrated and nonvolatile characteristics. Additionally indicate that time-variant and historical are fundamentally the same.
The Basic Plan

• Architecture of the design
  – Choose a process
  – Choose the grain fact table
  – Choose the dimensions
  – Choose how to handle updates

• Critical success factors
  – Focus on the business, not the technology
  – Use an iterative development methodology
  – Find a champion for the project and include end users

The Basic Plan
The design of a data warehouse should begin with a brief architectural phase that identifies the scope of the project. The initial business process or subject matter on which IS will focus development should be defined. It is not prudent to try and develop a “be-all, end-all” enterprise data warehouse that will take five years to complete. In fact, this is the primary reason for data warehouse development failures. Find the business process that will yield the highest return and that has a fairly accessible level of data, and target the implementation for no more longer than three to four months hence.

Next, choose the granularity for the development of the data warehouse or data mart. As you will see later on in the course, you are really establishing the primary fact table now. Identifying this at an early stage allows you to accurately decide on the dimensions. Include all business processes initially when defining dimensions for the data warehouse. By having the lines of business within the enterprise agree, the underlying definitions of the categories become conforming dimensions. When applied correctly, as other data marts come online, they will fit together as pieces in a puzzle “conform” to build a single picture rather than being disparate and disjointed.

Finally, determine how updates to the dimensions should be made. This is called slowly changing dimensions and will be discussed at length in a later lesson.
Basic Elements of the Data Warehouse

- **Source**: Source database or other source form
- **Data staging area**: Intermediate area
- **Target**: Presentation server for the new data warehouse or data mart

**Basic Elements**

**Source Database**: An operational or production database, centralized warehouse, legacy system, and so on that feeds a staging area

**Data Staging Area**: An operational data store that contains normalized data. This data store can be raw data that has not yet been scrubbed or summarized or it can contain data that has been preprocessed using Oracle tools such as Pure*Extract (an MVS legacy system migration tool) or Pure*Integrate (a powerful data cleansing, transformation tool). Eventually this data will be exported to one or more data warehouses.

**Target**: A presentation server for the new data warehouse or data mart where the data is organized and stored for direct query by end users, report writers, and other applications.
Basic Form of the Data Warehouse

A star schema (dimensional model) is a typical warehouse model. It is portrayed as a star, with a central table containing fact data, and multiple dimensional tables radiating out from it, all connected by keys.

Dimensional models are represented by two main types of data:
- Fact data
- Dimension data (This data is denormalized, unlike that in a typical relational model.)

Instructor Note
The main types of data are discussed later in greater length in the dimensional modeling lesson.
Data Warehouse and OLTP Database Design Differences

Unlike an OLTP database design, a warehouse database design must:
• Focus on queries
• Allow incremental development
• Be a nonvolatile structure
• Provide historical data

Data Warehouse and OLTP Database Design Differences

There are a number of major differences between designing a warehouse database and an operational (OLTP) system database. For the warehouse, you must reconsider your approach and thought processes:
• You must focus on designing for optimum query access. For the operational environment, you also optimize for data manipulation tasks.
• You develop a warehouse incrementally. Do not develop an enterprisewide warehouse initially because the risks are too great; the time that it takes to deliver the benefits is too long.
The final result is a dynamic structure, rather than the static operational structure.
The warehouse stores historical data by time period, unlike transactional data, which is current. Time becomes a design consideration in its own right. You must consider the time dimension and partitioning strategies. Partitioning will be discussed in a later lesson.
Data Warehouse Features

A data warehouse:
• Is a repository for information
• Improves access to integrated data
• Ensures integrity and quality
• Provides an historical perspective
• Records results
• Is used by a broad spectrum of end users for a variety of purposes
• Reduces the reporting and analysis impact on operational systems
• Requires a major systems integration effort

Data Warehouse Features
A data warehouse:
• Is an information repository that contains integrated and validated data for the entire organization
• Improves access to totally integrated data
• Ensures data integrity and quality (A primary concern of the warehouse is to provide quality data that can be reliably used for analysis, business planning, and decision making.)
• Provides a historical perspective that enables trend analysis, forecasting, and decision making based on data stored over a period of many years
• Records results and gathers information for an organization
• Is used by a wide variety of people, with a wide range of job functions and responsibilities for decision support, using specialized access tools
• Reduces the impact on operational systems because the data warehouse resides on a separate computing environment that is tailored to meet warehousing needs
• Is not an off-the-shelf product but a major systems integration effort
Exploring Data Warehouse Characteristics

Recall that Bill Inmon’s definition of a data warehouse focused on four major characteristics:

- **Subject-oriented**: Data related to the major subjects of the enterprise (customers, products, sales, and so on) is gathered together in one place.
- **Integrated**: Consolidated data from diverse independent systems is connected through surrogate keys, and so on.
- **Nonvolatile**: Data changes little if at all (slowly changing dimensions will be discussed at a later time).
- **Time-variant**: Data is stored over a period of time (generally a long period) and thus is historical.
Subject-Oriented

Data is categorized and stored by business subject rather than by application.

Subject-Oriented
Subject-oriented data is organized around major subject areas of an enterprise and is useful for an enterprisewide understanding of those subjects. For example, a banking operational system maintains independent records of customer savings, loans, and other transactions. A warehouse pulls this independent data together to provide financial information. You can access subject-oriented data related to any major subject area of an enterprise, for example:

- Customer financial information
- Toll calls made in the telecommunications industry
- Airline passenger booking information
- Insurance claim data

The data from diverse sources is transformed so that it is consistent and meaningful for the warehouse. This is a different way of looking at the data. As you may note from the diagram, this information comes from the online transactional processing (OLTP) environment. Information is extracted from OLTP systems, which are typically operational systems. These systems are geared to process hundreds, thousands, or even millions of transactions per hour, as opposed to data warehousing which process fewer, but long running jobs.
Integrated

Data on a given subject is integrated.

- Savings
- Customer
- Current account
- Loans

In many organizations, data resides in diverse independent systems, making it difficult to integrate the information into a single set of meaningful data for analysis. A key characteristic of a warehouse is that data is completely consolidated or integrated. Data is structured in a globally accepted manner, even when the underlying source data is structured differently (conforming dimensions). Integration and transformation processes can be time-consuming and costly. It requires commitment from every part of the organization, particularly top-level managers who make the decisions and allocate resources and funds.

Facts and dimensions are integrated through surrogate keys.

**Data Consistency** You must deal with data inconsistencies and anomalies before the data is loaded into the warehouse. Consistency is applied to naming conventions, measurements, encoding structures, and physical attributes of the data.

**Data Redundancy** Data redundancy at the detail level in the warehouse environment is eliminated; the warehouse contains only data that is physically selected and moved into it; however, selective and deliberate redundancy in the form of aggregates and summaries is required in the warehouse to improve the performance of queries, especially drilldown analysis.
Nonvolatile

Typically, data in the data warehouse is read-only (less volatile than operational systems). Data is loaded into the data warehouse for the first-time load, and then refreshed regularly. Warehouse data is accessed by business users. Warehouse operations typically involve:

- Loading the initial set of warehouse data (often called the first-time load)
- Refreshing the data regularly (called the refresh cycle)

Accessing the Data After a snapshot of data is loaded into the warehouse, it rarely changes. The physical warehouse is optimized for data retrieval and analysis.

Refresh Cycle The data in the warehouse is refreshed; that is, snapshots are added. The refresh cycle is determined by business users. A refresh cycle need not be the same as the grain (level at which the data is stored) of the data for that cycle. For example, you may choose to refresh the warehouse weekly, but the grain of the data may be daily.

Changing Warehouse Data There are occasions when you must change some reasonably static data. For example, your warehouse may contain information about the products your company sells. If a product description changes, the warehouse must also change to provide accurate information. This process requires careful planning and management (slowly changing dimensions is discussed in further detail in lesson 4).
Time-Variant

Warehouse data is by nature historical; data is retained for a long time, from two to ten years, compared with one to three months of data for a typical operational system. The data allows for analysis of past and present trends, and for forecasting, using what-if scenarios.

**Time Element** The data warehouse always contains an element of time, such as quarter, month, week, or day. Organizations use a variety of time periods as qualifiers for performing business analysis. The date may be a single snapshot date, such as 01-JAN-01, or a range, such as 01-JAN-01 to 31-JAN-01. This time dimension governs the granularity of the data warehouse. The designer should always anticipate that the user will want one level more of detail than they originally asked for.

**Special Dates** A time dimension usually contains all the dates required for analysis, including special dates such as holidays and events.
Oracle Data Warehouse Components

The goals of the Oracle data warehouse architecture are to load data from any source, manage any data, and analyze any information.

**Source** The primary source of data is internal operational or transaction data. Other data can be introduced and integrated, including external data. Oracle Pure*Extract is an MVS migration tool and Oracle Pure*Integrate is a deep-cleansing, matching tool that provides flexible and sophisticated manipulation of data sources.

**Data** All types of data (including text, image, spatial, audio, video, Web, and so on.) can be managed in a central repository and made available for access by end users for a variety of purposes. Oracle Universal Server provides a scalable, high-performance deployment platform for business applications. Oracle Express, an integrated component of Oracle9i Application Server, offers query, analysis, and reporting. Oracle Darwin, an integrated component of Oracle9i Database, is a robust data mining tool.

**Access** Data in the warehouse can be accessed through a variety of applications or tools, including relational tools, online analytical processing (OLAP) tools, and specific applications. Oracle Discoverer is an end-user query tool that provides the most flexible access to the relational data warehouse.
Oracle’s Common Warehouse Model (CWM)

- Oracle standard for data warehousing metadata
- Open standard for data warehousing and OLAP
- Enables tighter integration of metadata among the Oracle products

Oracle’s Common Warehouse Model (CWM)

CWM integrates all aspects of warehousing, incorporating both technical and business metadata into a single model. Unlike other approaches to metadata management which attempt to encompass all types of software and systems, CWM concentrates on the needs of data warehousing and decision support. By targeting a specific area, CWM is able to provide a rich model for warehousing that simplifies the development of warehouse tools and applications. As Oracle’s products become more tightly integrated, including with other industry-leading tools from Oracle partners, reduced implementation complexity and greater user productivity will result. To enable this open functionality, Oracle submitted a Request for Proposal for a Common Warehouse Metadata Interchange (CWMI) standard to the Object Management Group (OMG). The CWMI standard enables the interchange of warehouse metadata among data management and analysis tools, and among warehouse metadata repositories.

The acquisition of One Meaning by Oracle was a significant step towards the accomplishment of standardizing the CWM. The One Meaning product specialized in metadata that provided technology to enable metadata interoperability and transfer, reducing the cost of managing information resources, and enhancing the value of stored proprietary information. The Oracle metadata strategy provides essential integration and continuity, and adds ongoing value to data warehousing implementations.

Note: Metadata is defined as descriptive information about data stored in a central repository. Generally, the repository encompasses all corporate metadata such as, database catalogs, data dictionaries, navigation services, and so on, and includes elements such as name, length, and other data attributes.
Load from Many Sources

The data warehouse consists of data from many different sources and locations.

**Nonrelational Systems** Production systems include internal applications that maintain day-to-day data in such areas as human resources, manufacturing, finance, inventory, vendors, and customers, on a variety of hardware platforms. Additionally, these systems can be from rich data sources such as legacy, VSAM, flat files, and so on.

**Relational Databases** Production data can come from many different operating system and hardware platforms, file systems, and database systems, such as Oracle or DB2, and vertical applications such as Oracle Financials.

**External Data** External data is important if you want to compare the performance of your business against others. There are many sources for external data, such as purchased database data, periodicals, competitors’ information, newspapers, and magazines. Two examples of external data sources include Dunn & Bradstreet and the U.S. Census.

**External Formats** The format of the data can vary, from flat file to unstructured text, structured text, and graphical formats.

**Archive Data** Archive data of any type can be useful to the enterprise as a source of historical data.
Manage Data

The Oracle data warehouse solution can store a variety of data formats, providing added value to end users. Relational data forms are traditional formats in which data becomes information through the interpretation of numbers and text. Other data formats offer the end user a greater variety of interaction, such as audio and video data, spatiotemporal data (which describes objects in space and time), and Web-formatted data such as click-streaming.
Decisions Support Systems (DSS)

DSS provides a system that enables decision making. Every warehouse implementation requires a storage methodology and analytic tools for end user access to facilitate the consolidation of data across many sources. Depending on the business requirements, the tools can be simple reporting tools or can be more complex. Listed here is additional information for DSS:

- Operation Data Store (ODS) is a database of operational data that is formatted as it is collected. An ODS is not a data warehouse; it is missing the nonvolatile characteristic, in that it can change as users update information.
- Data warehouse (DW) is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management’s decision-making process.
- Online analytical processing (OLAP) is a loosely defined set of principles that provides a dimensional framework for decision support. OLAP analyzes data to reveal business trends and statistics that are not immediately visible in operational data. OLAP is also known as multidimensional analysis. Other forms of OLAP, such as MOLAP and ROLAP, also exist. These analytic tools will be discussed in detail in lesson 8.
- Data mining (DM) is a technique that discovers previously unknown patterns and relationships in data.
Desirable Warehouse Functionality

**Scalable** Data warehouses are capable of storing and managing enormous volumes of transactional and summarized data. A data warehouse can grow by an order of magnitude over time through the need to add new types of data, or because historical data must be retained for long periods.

**Manageable** The warehouse requires products that enable management decision making.

**Available** The server must be robust to ensure that the warehouse is available whenever it is needed.

**Extensible** It must be possible to add new types of transactional data and introduce new aggregations and summarized data to a warehouse as information requirements evolve. It must be easy to reconfigure.

**Flexible** The data access paths to detailed and summarized data must be flexible to support multidimensional analysis, drilldown, and true ad hoc querying.

**Integrated** A data warehouse must be fully integrated with the existing systems and operational environment so that data from multiple sources can be loaded efficiently.

**Accessible** A data warehouse must be accessible, providing front-end tools that are flexible and intuitive and cover the requirements of a wide range of users.

**Reliable** Discrepancies in data that is loaded from multiple sources must be reconciled so that the warehouse always contains information that is accurate and consistent for a given point in time. Operation and management processes should be considered iterative in nature.
The Enterprise View
A data warehouse provides the enterprise with information for strategic decision making. Organizations often use a combination of data warehouses and data marts. Data warehouses store enterprisewide collections of data, whereas data marts typically store line-of-business data for specific departments or functional groups.

Data Warehouse A data warehouse is a strategic collection of all types of data in support of the decision-making process at all levels of an enterprise. It is a single data store containing the subject-area data of an enterprise. The subject-area data can represent a single subject or cover all subject areas of the enterprise and is independent of the operational systems. Keep in mind that data warehouses are created for two primary reasons: analytical reporting and decision support.

Data Marts A data mart is a type of data warehouse that fits into the enterprise architecture in a variety of ways. It can be a subject-oriented data warehouse for functional or departmental information needs, consisting of a subset of the enterprisewide data warehouse, or it can be a mini-enterprisewide data warehouse, combining data from multiple subject areas and acting as a kernel to feed the enterprise warehouse. Data marts can be categorized as dependent, sourcing data from an enterprise warehouse; or independent, sourcing data directly from legacy systems. The preferred method of building a data mart that can be grown from a kernel into an enterprise through incrementally successful short-term implementations, generally materializes return on investment faster and adheres to the development of each project by limiting scope creep.
Comparing Data Warehouses and Data Marts

**Scope** A data warehouse deals with multiple subject areas and is typically implemented and controlled by a central organizational unit such as the corporate information technology group. It is often called a central or enterprise data warehouse.

**Subjects** A data mart is a simpler form of a data warehouse, designed for a single line of business (LOB) or functional area such as sales, finance, or marketing, for example, it provides localization, which means that it serves the users in a specific level, department, or region.

**Data Sources** A data warehouse typically assembles data from multiple source systems. A data mart typically assembles data from fewer sources.

**Implementation Time** Data marts are typically smaller and less complex than data warehouses, and therefore are easier to build and maintain. A data mart can be built as a proof-of-concept step toward the creation of an enterprise-wide warehouse.

Data marts are sometimes differentiated from data warehouses based on size; however, some data marts can be larger in size than some data warehouses. The main distinction between data marts and data warehouse involves scope, usage, and which staff initiates, creates, and manages the project.
**Flow of Data Through the Warehouse**

Data is extracted from the source, transformed as needed, then transported to the appropriate place within the data store. Different types of data facilitate the operations of the warehouse. Raw data provides the lowest level of data loaded by the Extraction, Transformation, and Transportation (ETT) process. Summary-level data is pre-aggregated raw data stored to facilitate analysis. Metadata provides the road map to the warehouse environment, including specific details about the data. Tools access the data in the warehouse to support analysis and other queries.
Enterprise Model Architecture

Two types

• Centralized
• Decentralized

Enterprise Model Architecture

The enterprise model is a data model that encompasses the entire organization. Metadata is shared with all departments in the organization, creating a cohesive model. There are two forms of the enterprisewide model:

• Centralized
  – Data is stored in one place
  – Subjects are collocated
  – Presently of benefit due to network constraints for accessing large volumes of data over a LAN or WAN

• Decentralized
  – Disperse data marts geographically, generally making analytic analysis faster at each location
  – Dimension and fact tables are conformed
Dependent Data Mart Model

Data marts can be categorized into two types: dependent and independent. This diagram depicts separate data marts that are being formed from an original enterprise level. The data in the marts is dependent upon the enterprise for refresh. An ODS (Operational Data Store) can not be used for anything other than staging.
Independent Data Mart Model

Independent data marts are iterative instances developed from using the top-down approach of data warehouse development. If analysis is performed using the strategic methodology mentioned previously and dimensions are conformed properly, these independent data marts can eventually form an enterprise warehouse.
Development Models Compared

Traditional software development

- Waterfall approach

Model #1: Waterfall

- Changes have little impact during the initial phase.
- Requirements are defined, documented, and frozen.
- Some assumptions are made:
  - Business requirements must remain static
  - Lengthy time constraints due to meticulous analysis and design requirements
  - Potentially requiring years for implementation
  - Must have a consensus on system requirements with the user community
Development Models Compared

Data warehouse development
  • Spiral Approach

Development Models Compared
Model #2: Spiral
  • Define the Problem/Identify the business requirements
    – Initial presumptions are established and explored
    – Proof of concept evolves
    – Gather requirements and specifications
  • Functionality is initially limited for short implementation cycle.
  • With each pass, new functionality is explored, new evaluations are forced, and adjustments are made causing the spiral to expand outward.
  • As one data mart is implemented, others are being forged through successive iterations. Decision support systems are implemented quickly giving rise to other requirements.
  • IS credibility increases as short term projects are successfully implemented.
Spiral Approach

• Phase I: Strategy
  – Identify business requirements.
  – Define objectives and purpose.

• Phase II: Definition
  – The project is scoped and planned.

• Phase III: Analysis
  – Information requirements are defined.

• Phase IV: Design
  – Database structures to hold base data and summaries are created and translation mechanisms are designed.

Spiral Approach
You must be able to articulate the needs of users to ensure that you have interpreted the needs correctly. Focus on their needs and not the technology.
• Reiteration is crucial.
• Define the problem by seeking information from the business community.
• Anticipate the design challenges.
• Model the business.
Spiral Approach

• Phase V: Build and document
  – Warehouse is built and documentation is developed.

• Phase VI: Populate, Test and Train
  – Warehouse is populated and tested.
  – Users are trained.

• Phase VII: Discovery and Evolution
  – Warehouse is monitored and adjustments are applied
    or future extensions are planned.
Data Warehouse Database Design Phases

1. Defining the business model
2. Defining the logical model
3. Defining the dimensional model
4. Defining the physical model

Data Warehouse Database Design Phases

In the past several years, a number of methods for designing a data warehouse have been published. Although various sources use alternate syntax or define certain terms differently, all generally include the same tasks that are required to produce a sound data warehouse database design. This course attempts to focus on the major tasks associated with the data warehouse database design process. These tasks have been grouped into four phases that include:

- Defining the business model (conceptual model). A strategic analysis is performed in to identify business processes for implementation in the warehouse. Then, a business requirements analysis is performed, where the business measures and business dimensions for each business process are identified and documented.
- Defining the logical model. A graphic-intensive technique that results in a data model representing the definition, characteristics, and relationships of data in a business, technical, or conceptual environment. Its purpose is to describe end-user data to systems and end-user staff. This method is generally referred to as third normal form (3NF).
- Defining the dimensional model. The business model is transformed into a dimensional model. Warehouse schema tables and table elements are defined, relationships between schema tables are established, and sources for warehouse data elements are recorded.
- Defining the physical model. The dimensional model is transformed into a physical model. This includes the documentation of data element formats, database size planning, and the establishment of partitioning strategies, indexing strategies, and storage strategies.
Summary

In this lesson, you should have learned how to do the following:

• Describe the features and four characteristics of a data warehouse
• Identify the components and functionality of an Oracle data warehouse environment using the Common Warehouse Model
• Identify the three types of data warehouse models
• List the major tasks of data warehouse database design
Practices 1-1 Overview

This practice covers the following topics:

• Defining a data warehouse
• Identifying the seven phases in a spiral approach to data warehouse design
• Describing Oracle’s Common Warehouse Model

Practice 1-1
Solutions for this exercise are located in Appendix B.
Practice 1-1
Answer the following questions.

1. What is perceived to be the single biggest cause for data warehouse database design failure?
   ___________________________________________
   ___________________________________________
   ___________________________________________
   ___________________________________________
   ___________________________________________

2. What are the two basic data forms of a data warehouse star schema?
   1. ___________________________________________
   2. ___________________________________________

3. What would Bill Inmon consider to be the four most important characteristics of a data warehouse?
   1. ___________________________________________
   2. ___________________________________________
   3. ___________________________________________
   4. ___________________________________________

4. What are the chief differences between an OLTP database and a data warehouse?
   ___________________________________________
   ___________________________________________
   ___________________________________________
   ___________________________________________
   ___________________________________________

5. What are four components of a decision support system?
   1. ___________________________________________
   2. ___________________________________________
   3. ___________________________________________
   4. ___________________________________________
6. What is the Common Warehouse Model and why is it important?

___________________________________________
___________________________________________
___________________________________________
___________________________________________
___________________________________________

7. Define scalability.

___________________________________________
___________________________________________
___________________________________________
___________________________________________

8. A _________ is a simpler form of a data warehouse and is designed for a single line of business or functional area.

9. a) What are the seven phases in the spiral approach to data warehouse design?
   1. ___________________________________________
   2. ___________________________________________
   3. ___________________________________________
   4. ___________________________________________
   5. ___________________________________________
   6. ___________________________________________
   7. ___________________________________________

   b) Why is this approach preferable to the waterfall approach?
   ___________________________________________
   ___________________________________________
   ___________________________________________
   ___________________________________________
Defining the Business and Logical Models

ILT Schedule:

<table>
<thead>
<tr>
<th>Timing</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>75 minutes</td>
<td>Lecture</td>
</tr>
<tr>
<td>60 minutes</td>
<td>Practice</td>
</tr>
<tr>
<td>135 minutes</td>
<td>Total</td>
</tr>
</tbody>
</table>

Instructor Note

The Oracle Warehouse Method is provided as an Appendix to this material. You may review that methodology should you choose to at this point.
Objectives

After completing this lesson, you should be able to do the following:

- Describe enterprise-level strategic analysis tasks
- Define components of business modeling
- Discuss the role of metadata in the data warehouse and methods of tracking metadata
- Define a logical model and entity relationship modeling

Lesson Aim

Upon completing this lesson, participants should be able to list the major phases of the data warehouse design process and describe the specific tasks associated with the first phase: defining the business model. Participants should learn how to:

- Identify primary business processes after performing an enterprise-level strategic analysis
- Perform business requirements analysis to create the business model
- Begin the metadata documentation process by creating a metadata document
Designing the Conceptual Model

Phase I: Defining the business model
Design Challenges

- Query requirements
- Data restructuring
- Response times
- Managing data volumes
- Operating costs and interfaces
- Controlling interfaces and tools
- Backup and restore

Design Challenges
You are unlikely to know all the queries that are going to be made against the warehouse, and users are unlikely to be able to tell you immediately. In addition, users may find that the results of one query drives the need to answer other questions, which cannot easily be predicted early on. Users require retraining in this area.

Design for flexibility. It is unrealistic to continually restructure the data to aid query performance, so ensure that the model is flexible (or generic) and able to adapt to changing and evolving requirements.

Additional design challenges include the following:

- Querying response time requirements
- Managing the volumes of data
- Controlling and managing operating costs
- Controlling interfaces and tools
- Controlling the effect of long-running processes such as ETT tasks
- Backup and recovery tasks with large volumes of data and required high levels of availability
Business Modeling

- Performing strategic analysis
- Creating the business model
- Creating the metadata

Business Modeling
The first phase, business modeling, includes at least three tasks, each with associated deliverables. These tasks include strategic analysis, business model creation, and metadata document creation.

Strategic Analysis Performed at the enterprise level, strategic analysis identifies and prioritizes the major business processes (also called business events or subject areas) that are most important to the overall corporate strategy. The primary business process (or processes) candidate is selected for implementation in the warehouse.

Business Model Creation The business (conceptual) model is developed by uncovering detailed business requirements for a specific business process and verifying the existence of source data needed to support the business analysis requirements.

Metadata Creation The metadata is created in this first phase of the design process. The results of business model are summarized in the metadata tool, and this information serves as the essential resource for subsequent phases in the design process.
Performing Strategic Analysis

• Identify crucial business processes
• Understand business processes
• Prioritize and select business processes to implement

Performing Strategic Analysis
This task is essential for producing a warehouse design that returns value, is achievable, and is deliverable within an acceptable time frame. It includes the following steps:

• Identify the business processes that are most important to the organization from the standpoint of decision making. A business process is a major operational process in your organization that is supported by a legacy system (or systems) from which data can be collected for the purpose of the data warehouse. Examples of business processes are orders, invoices, shipments, inventory, sales, account administration, and the general ledger.
• Understand business processes by drilling down on the analytic parameters, called business dimensions, associated with each business process. The creation of a business process matrix can aid in this effort.
• Prioritize and select the business process to implement in the warehouse, based on which one will require the least amount of time to implement while providing the largest return on investment (ROI).

The goal of strategic analysis is a simplified and targeted scope for the data warehouse design. Note: In certain warehouse initiatives, such as a targeted data mart, the strategic analysis task may be complete and a primary business process already identified.
A key strategic initiative is to reduce costs. Determine how to do this in your organization by analyzing data, fixing anomalies, and predicting the effect of changes.
Doing Your Homework

• Determine a compelling business motivation for building the warehouse
• Read the company annual report
• Collect pertinent external marketing literature
• Investigate earlier data warehousing attempts
• Evaluate the feasibility of adding necessary hardware

Do Your Homework
Before attempting to identify crucial business processes by working with individuals from the organization, you must do your homework. There are a number of resources that can provide context for the strategic analysis task. These include:
  • The company annual report, which may provide a narrative review of business operations, insight regarding organizational structure, and reporting hierarchies.
  • The organization’s external marketing literature, Web site, and internal intranet.
  • Earlier data warehousing attempts. Earlier efforts may not have been called warehouses, but nevertheless they were business-oriented reporting or analysis systems. Find out what worked, what didn’t, and why. Ask why did it fail or succeed?
Recognizing Critical Success Factors

• Focus on the business
• Determine who your sponsors are for the internal business plan and forge strategic alliances with their business units
• Use an iterative methodology with a short deliverables time frame

Recognize Critical Success Factors
Each data warehouse project has critical success factors that must be identified and followed to breed success.
• Do not focus on technology. Business decisions are focal.
• Identify and establish unblocked, communicative relationships with the your business partners by creating a team of partners with business users. From these business users, seek out an advocate from the user community to head the project effort.
• Read your sponsor’s internal business plan. This will help you stay focused for your interviews and give you a better grasp of their concerns. Include short phases that have frequent deliverables to help manage expectations and to stay on track. Remember that the user’s business is dynamic in nature and shorter project lifecycles can take new, more pressing changes into consideration.
Understanding Business Imperatives

You must understand several imperatives that exist for the business.

• Seeking a competitive advantage implies that businesses must constantly monitor the market and have data available to sustain this task.
• Tracking competitors products and satisfying customers’ needs is critical for sustained growth.
• Reacting quickly and effectively to market changes is important to remain competitive. Product life-cycles are becoming shorter, and customer outlooks for new products and services have increased markedly, whereas expectations for time to market have drastically been reduced.
• The ability to adapt to changing customer demands, internal pressures, and regulatory requirements is essential.
• Having the ability to query all source data (regardless of format) in a standard format using their individual query tools
• Providing the user with timely information to make good business decisions for rapid payback
Selecting Strategic-Level Interviewees

Identify executives from across the organization:

- Sales
- Marketing
- Manufacturing
- Logistics
- Finance

Selecting Strategic-Level Interviewees

Your initial interviews should be targeted at business executives who can provide an overall understanding of the organization and where it is headed. If possible, you should interview horizontally across the organization beyond the sponsoring group. This cross-functional perspective helps ensure that your warehouse will be extensible across the enterprise, especially with regard to common data such as product or customer, even if you are already focused on a single primary business area for your initial data warehouse project.

For example, if marketing is designated as the initial or sponsoring user group, you should also speak with business executives from sales, customer service, manufacturing, finance, and other groups to better understand the business as a whole.
Developing Strategic Questionnaires

- Why do you need a data warehouse?
- What specific problem are you trying to solve?
- What are your objectives?
- What are the available resources?
- How will the results be measured?

Developing Strategic Questionnaire
The questions you ask in these initial interviews will be of a strategic nature, because your objective is to get an overall understanding of the organization and identify crucial business processes.

High-level questions you might ask include the following:
- Why do you think you need a data warehouse? What are you trying to accomplish?
- What are the objectives of your organization?
- How do you measure success? How do you know you’re doing well? How often do you measure success?
- How do you identify problems or know that you might be headed for trouble?
- What are the key business issues you face today? What could prevent you from meeting these objectives? What is the impact on the organization?
- What is your vision to better leverage information within your organization?

Answers to these questions provide insight into key business processes and goals such as increasing sales, decreasing inventory expenses, reducing shipping costs, improving manufacturing production, and so on.

They help identify success metrics, the core data elements that must be tracked in the warehouse. Some processes may have only a single metric, whereas others may have a host of metrics. Examples include sales volume, market share, profit, response rate, and so on.

The last question helps identify whether the executives really want to enforce fact-based, measurable results instead of gut-based decision making. It clarifies expectations of their staff’s interaction with the data.
Creating the Business Model

- Defining business requirements
- Determining granularity
- Documenting metadata

Creating the Business Model
Now that the strategic business process or processes have been identified for implementation in the warehouse, a business model is created.

Defining Business Requirements The business model is created by defining business analysis requirements for each selected process. Here, you meet with business managers and business analysts who are directly responsible for the specific business processes in order to:

- Define and document examples of their business measures
- Create a detailed listing of the analytic parameters for each measure
- Identify the granularity required to satisfy the analysis requirements
- Clarify business definitions and document business rules

Verifying Data Sources Concurrently, you must perform an information technology (IT) data audit, a systematic exploration of the underlying source systems to verify that the data required to support the business requirements is available.
Establishing Business Requirements

- Primary input
  - Interviews with business users

- Secondary inputs
  - Existing metadata
  - Production ERD model
  - Research

Establishing for Business Requirements
There are several resources that can serve as inputs in the exercise of defining business requirements.

**Primary Input** The primary input for business requirements are interviews with the business users and analysts who are responsible for driving, measuring, and analyzing the business process. Prioritize the business requirements to narrow the focus.

**Secondary Input** Requirements gathered from user interviews can be clarified or augmented with existing source data information and further research regarding how data is currently used. Other sources may be:
- Legacy systems metadata
- Source ERD from OLTP systems
- Research from previous warehouse attempts or existing business applications such as the “management reporting system” or “marketing information system.”

**Caution:** Existing production systems or source data documentation should serve only as a verification or supporting resource for the business user interviews. Do not attempt to use this kind of information as the primary input for business requirements. Reverse-engineering a business model from secondary inputs rarely produces a warehouse design that returns real value to the users.
Gathering Business Information Requirements

• How does your department measure success and how often?
• Describe your products (and other key business dimensions). Is there a natural way to categorize or group these?
• What type of routine analysis do you currently perform?
• What essential reports do you use?

Gathering Business Information Requirements
The tactical interviews used to gather business information requirements differ from strategic executive interviews in scope and depth. More detailed questions are asked to uncover specific measures and analytic requirements for individual business processes. For example:

Business measures:
• How does your department measure success? Sales dollars? Production cost dollars? Units sold? Units shipped?
• What is the scale of these metrics? Hundreds? Thousands? Millions?

Business dimensions:
• How often do you measure success? Is the same time frame associated with each measure? Do you work on a calendar or fiscal year?
• Describe your products (and other parameters such as customers, vendors, stores, and so on).
• Is there a natural way to categorize or group these parameters? How often do the major categories change? Do you evaluate major dimension categories at different levels?

Reports and analyses:
• What type of routine analysis do you currently perform? What data is used? What do you do with the information? How much historical information is required?
• What essential reports do you use? What data is important and what’s missing?
Gathering IT Information Requirements

Every organization has IT people who know the data. You should speak with the data experts and people who are responsible for core operational systems. IT information interviews are intended to provide a reality check—does the data exist to support business requirement themes? Questions for these interviews might include:

- Where is the data located today, on what systems, and at which physical locations?
- What is the format of the data? Is it in a database or flat files?
- How often is the data refreshed?
- How volatile is the data (how often does it change)?
- How is the data accessed today?
- What networking and protocols exist for accessing the data?
- What is the quality of the data? Is it accurate?
When Do You Say No?

- Operational systems
- Operational data stores
- Daily updates

When do you say no?

When current operational systems are not meeting the needs of business users, a common pitfall for data warehouse architects is to retain data at the most minute level of detail. Additionally, business users want this data posted within hours, if not minutes of creation. Although this most certainly benefits these business users in the short term, it will cause failure of the data warehouse project for the following reasons:

- Violates one of the four principal aspects of a data warehouse—nonvolatile
- Level of granularity—detail is lost over time and mass amounts of unnecessary data are retained
  - Time to run queries, backup, and recovery is greatly increased
  - Costly to store

Bill Inmon recognized the deficiencies of some operational systems and suggested that these deficiencies could be resolved by providing an intermediate storage area that he called operational data stores (ODS). In setting up an ODS, detailed data could be stored for a short time frame and then removed. This data would be rolled up to the data warehouse in summary fashion, maintaining the integrity and intended use of the warehouse on a periodic basis, and providing a much longer term perspective for business users. Additionally, this would also be of benefit because it would limit the number of short, transient transactions against the data warehouse. The ODS stores tactical data from production systems and are subject-oriented and integrated to address operational needs. Some operational data stores are in third normal form.
The Importance of Business Requirements

Business requirements determine:
- The data that must be stored in the warehouse to answer business questions
- How the data is organized
- How often data is updated
- End-user application templates
- Maintenance and growth plans

The Importance of Business Requirements
Business users and their requirements affect almost every decision that is made throughout the implementation of a data warehouse. The entire scope of the data warehouse initiative must be driven by business requirements, including (and especially) the warehouse design process.
Identifying Potential Business Processes

Sample business processes:
- Regional store-front sales
- Internet sales
- Promotions
- Warehousing
- Shipping

Identifying Potential Business Processes
As stated previously, a primary goal of strategic interviews is to identify the organization’s crucial business processes. Strategic analysis interviews reveal the core business processes that are potential candidates for the warehouse.
In the example shown here, a hypothetical business case combines two businesses in the process of completing a merger. One business is a “dot com” wine vendor, whereas the other is a traditional wine importer. To obtain insight into both customer bases, the combined company wants to see its total customer base, promotions, sales, and so on.
Examining the Business Processes: Measures and Business Dimensions

After the primary business processes are identified, you should make an attempt to identify the measures and business dimensions associated with each business process.

**Measures** Business measures are the success metrics of a business process, and are the core data elements that must be tracked in the warehouse. Some processes may have only a single measure, whereas others may have a multitude of measures. Examples include sales volume, profit, response rate, and so on.

**Business Dimensions** Business dimensions are the analytic parameters that categorize business processes for analysis purposes. They can be discovered by asking the business managers how they look at the processes of booking orders, invoicing customers, picking shipments, or stocking the stores. Business dimensions provide the metadata definitions for the data warehouse.

For example, if the important business process measures are sales dollars, volumes, dollar costs, or margins, then how do managers evaluate these measures? By product? By country? Over time? By customer type? By vendor? By channel?
Using the Business Process Matrix

A useful tool to understand and quantify business processes is the business process matrix (also called the process/dimension matrix). This matrix establishes a blueprint for the data warehouse database design to ensure that the design is extensible over time.

The business process matrix aids in the strategic analysis task in two ways:

- Helps identify high-level analytical information that is required to satisfy the analytical needs for each business process, and serves as a method of cross checking whether you have all of the required business dimensions for each business process.
- Helps identify common business dimensions shared by different business processes. Business dimensions that are shared by more than one business process should be modeled with particular rigor, so that the analytical requirements of all processes that depend on them are supported. This is true even if one or more of the potential business processes are not selected for the first increment of the warehouse. Model the shared business dimensions to support all processes, so that later increments of the warehouse will not require a redesign of these crucial dimensions.

A sample business process matrix is developed and shown in the slide, with business processes across the top and dimensions down the column on the very left side.
Selecting the Business Process

- Which will provide the best ROI?
- Which will provide the greatest strategic value?
- What kinds of business decisions can you make with this information?
- Which will be the easiest to implement in terms of human resources?
- How long will it take to implement this business process?

Selecting the Business Process

After the primary business processes have been broken down in terms of measures and analytic parameters, they should be prioritized based on the following:

- Which business process will provide the greatest return on investment (ROI) in the warehouse? It is necessary to perform a cost analysis and ROI analysis to evaluate these business processes (conducted by the business users). Prioritize the business processes based on the greatest ROI.
- What strategic value might be returned by implementing this business process in the warehouse?
- What kinds of business decisions can you make with this information?
- What development effort is required in human resources?
- How long will it take to implement this business process in the warehouse? The initial warehouse implementation should be achievable within 90 to 120 days.

Note: Although the data marts are developed incrementally, capacity planning must still reflect an overall budget for other future data marts.
Process Modeling

Process model components:
- Organization units
- Process steps
- Events
- Flows

Process Modeling
Strategic analysis, as explained in this lesson, is process modeling, for example, a technique that focuses on the detail of how the business works. You must understand the processes in order to specify business requirements. Process models describe business processes as systems that receive input, produce output, and are triggered by events.
  - Organization units represent agents, whether internal to the business or outside of it, who perform process steps.
  - Process steps are activities within the business that either produce a product or provide a service.
  - Events are occurrences inside or outside the business environment that either initiate process steps or result from them.
  - Flows show the order in which steps are performed.
Granularity and Business Dimensions

While gathering more specific information about measures and business dimensions, you must understand the level of detail that is required for analysis and business decisions. This level of detail is called granularity. The greater the level of detail, the finer the level of granularity. During your interviews, you should discern the level of detail that users need for near-term future analysis. After that is determined, identify whether there is a lower level of grain available in the source data. If so, you should design for at least one grain finer, and perhaps even to the lowest level of grain. Remember that you can always aggregate upward, but you cannot decompose the aggregate lower than the data that is stored in the warehouse. The level of granularity for each dimension determines the grain for the atomic level of the warehouse, which in turn will be used for rollups.
Measures and Key Reports

Following the interviews, review the reports and analysis gathered from your primary contacts. These reports and analyses typically provide a rough translation into your business data model.

- Measures equate to the numeric report values. In this example, the measure is sales dollars.
- Business dimensions can be read from the report, row, and super column headings. In this example, month, region, district, and store are the parameters.
- It is often helpful to review these reports offline for further insight into the business users’ analytic requirements. Sometimes you see business dimension values in these reports that never came up in interviews.
- Follow-up: The potential for deeper granularity or further drilldown on business dimensions should then be discussed in follow-up conversations with business users.
- Finally, you should mark each report with the donor’s name, department, and date received.
Documenting Business Model Elements

For the selected business processes, the following business model elements should be documented:

• A list of the business measures
• Detailed lists for each business dimension
• Business definitions and rules

Identifying Business Model Elements

As stated previously, the strategic analysis task creates two deliverables: a list of potential business processes and the selection of the business processes to implement in the current warehouse iteration. You should consider drafting an initial list of warehouse users at this point as well. Initially, you should limit the user list for the warehouse.

Measures and Business Dimensions For the selected business processes, the business requirements analysis task should result in:

• A list of the business measures
• A detailed list of values for each business dimension, down to the finest level of grain

Business Rules and Definitions Business model elements should also be documented with agreed-upon business rules and definitions. For example, a retail customer sales process might include the following:

• Product items are grouped by product category.
• Products are also grouped by brand.
• March, April, and May make up the first quarter in the fiscal year.
• Stores are organized and evaluated along a straight geographic grouping of districts and regions, as well as a market grouping that combines geographic areas.
Identifying Business Definitions and Rules

When you are conducting interviews, pay attention to vocabulary. Often, the same term has multiple meanings, or different terms are used to describe the same thing. Business vocabulary standardization issues become especially apparent as you conduct interviews horizontally across the organization. You should:

- Record business definitions during your interviews, making notes of any inconsistencies in vocabulary. These inconsistencies must be resolved in follow-up sessions with cross-department decision makers.
- Require exact definitions for business dimension terminology, because this has a direct impact on the grain and structure of the eventual dimensional model.
- For measures, document time retention requirements for online, near-line (disk), and archive (tape) storage.
- Listen for and document implicit business rules during your interviews. The business user interview is one of the best forums for gathering these rules, which serve as direct inputs for the development of ETT processes.

<table>
<thead>
<tr>
<th>Credit Rating</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+</td>
<td>0 bad checks or bank credit failures</td>
</tr>
<tr>
<td>A</td>
<td>1 bad check or bank credit failures</td>
</tr>
<tr>
<td>B</td>
<td>2 bad checks or bank credit failures</td>
</tr>
<tr>
<td>C</td>
<td>3 or more bad checks or bank credit failures</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>A customer with a credit rating of A or above will receive a 10% discount on any order totaling $500 (U.S.) or more</td>
</tr>
<tr>
<td>Rule 2</td>
<td>A customer with a credit rating of A or above will receive a 5% discount on any order totaling $250 (U.S.) but less than $500</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Rule 5</td>
<td>A customer with a credit rating of C will not receive any discounts on purchases</td>
</tr>
</tbody>
</table>
### Documenting Business Measures

**Current Increment**
- Sales
  - Cost
  - Discount
  - Margin
  - Sales amount
  - Channel
  - Units

**Future Increments**
- Promotions
  - Type
  - Description
- Suppliers
  - Name
  - Address
  - Contact name

---

**Documenting Business Measures (Facts)**
Determine which process is the easiest to implement yet provides the largest ROI. This example shows the sales process as the choice with future increments focusing on suppliers and promotions.

**Note:** Facts or fact data are the measurements within the core data warehouse on which all OLAP queries depend. Generally, facts are additive measures characterized by composite keys which join to keys in the dimension tables.
### Documenting Business Dimensions

Establish the primary business processes and document these using a metadata tool of your choice. (Only a few selected processes are shown in the slide.) These business dimension values should be identified during the process when business information requirements are analyzed (from interviews, report analysis, and so on).

**Note:** Dimensions or dimension data is the data by which a user queries business measurables. The dimensions are stored in tables and are joined to facts by a key value.

**Important:** Dimensions are data dictionary structures that define relationships between columns in existing tables. Dimensions are highly recommended because they enable additional rewrite possibilities.

<table>
<thead>
<tr>
<th>History</th>
<th>Product</th>
<th>Sales</th>
<th>Order</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Product_Name</td>
<td>Cost</td>
<td>Item_Nbr</td>
<td>Name</td>
</tr>
<tr>
<td>Day</td>
<td>Category</td>
<td>Discount</td>
<td>Item_Desc</td>
<td>Type</td>
</tr>
<tr>
<td>Month</td>
<td>Type</td>
<td>Margin</td>
<td>Quantity</td>
<td>Credit_Rating</td>
</tr>
<tr>
<td>Quarter</td>
<td>Code</td>
<td>Sales_Units</td>
<td>Discount_Price</td>
<td>Address</td>
</tr>
<tr>
<td>Year</td>
<td>Promotion_Code</td>
<td>Unit_Price</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day_Desc</td>
<td>Warehouse_Loc</td>
<td></td>
<td>Order_Amount</td>
<td></td>
</tr>
<tr>
<td>Quarter_Day_Desc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Month_Day_Desc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Year_Day_Desc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please note that this is not an all-inclusive list.
Creating the Metadata

Metadata should:
• Document the design process
• Document the development process
• Provide a record of changes
• Record enhancements over time

Creating the Metadata
Metadata has been described as “data about data.” This is a rather generic definition however. Warehouse metadata is descriptive data about warehouse data and the processes that are used in creating the warehouse. It contains information used to map the data between the source systems and the warehouse, and additionally, contains transformation rules. The metadata repository (or document) should be created in the business modeling phase and used to record the first layer of business metadata. These business modeling results are summarized within the metadata and serve as the essential resource for subsequent phases in the design process.
The metadata repository eventually contains detailed descriptions of the sources, content, structure, and physical attributes of the data warehouse.
It is important to identify the business users who are the stewards or caretakers of the metadata. This keeps the business involved in the process while providing a clear, coherent understanding of metadata usage and definitions.
Types of Metadata

There are three types of metadata: business, ETT, and operational.

**Business (End-User) Metadata** Business metadata (also called end-user metadata) contains two layers of information:
- First, it contains a summary of the business modeling phase, including the business processes identified for the warehouse, measures and business dimensions for each business process, business definitions and rules, and source data verification.
- Second, it contains the location and structure of warehouse data for user access. Knowledge workers use this metadata as a catalog for warehouse navigation.

**Extraction, Transformation, and Transportation (ETT) metadata** ETT metadata contains the dimensional model, physical model (detail and summary data definitions), and mapping information (detailing how data is transformed from source systems into its new format for the warehouse). It contains all the rules (including the business rules recorded in the business metadata) for extracting, scrubbing, summarizing, and transporting data.

**Operational (Technical Administration) metadata** Operational metadata is used by the load, management, and access processes for scheduling data loads and end-user access. It contains information about housekeeping activities, statistics of table usage, and information about every aspect of performance.
Metadata Documentation Approaches

• Automated
  – Data modeling tools
  – ETT tools
  – End-user tools
• Manual

Metadata Documentation Approaches
Regardless of the tools you use to create a data warehouse, metadata must play a central role in the design, development, and ongoing evolution of the warehouse.

Automated There are three types of tools that automatically create and store metadata:
• Data modeling tools record metadata information as you perform modeling activities with the tool. Some tools are more sophisticated than others, containing specific functionality designed for warehouse modeling.
• ETT tools can also generate metadata. These tools also use the metadata repository as a resource to generate build and load scripts for the warehouse.
• End-user tools generally require the administrator to create a metadata layer that describes the structure and content of the data warehouse for that specific tool.

Each of the tools used in your warehouse environment might generate its own set of metadata. The management and integration of different metadata repositories is one of the biggest challenges for the warehouse administrator.

Manual You can also create and manage your own metadata repository, using a tool that does not dynamically interface with the warehouse, such as a spreadsheet, word processor document, or custom database. The manual approach provides flexibility, however, it is severely hampered by the labor-intensive nature of managing a manual approach with the ongoing maintenance of metadata content.
Business Metadata Elements

The first layer of business metadata is recorded during the business modeling phase. Each business process implemented in the warehouse should be documented with the following entries:

- **Measure**—The name of the measure (such as dollar sales, units sold, and so on).
- **Business dimension (analytic parameter)**—The name of the high level business dimension (such as product, time, customer, and so on).
- **Dimension (parameter) attribute**—The name of the business dimension attribute.
- **Sample data**—An example of the source data for the dimension attribute or measure.
- **Business definition and rules:**
  - For business dimension attributes, a definition of the attribute in business terms, along with any business rules directly associated with that parameter (later, business rules are recorded in the ETT metadata for data transformation purposes)
  - For measures, a definition of the measure and time retention requirements for online, near-line (disk), and archive (tape) storage
  - Source system verification: Verify the source system file or database for each measure and dimension attribute. If the source system is a database, attempt to document the database table and column for each attribute
  - Source data expert: The name and contact information of the data source expert
Designing the Logical Model

Phase II: Defining the logical model

Designing the Logical Model
Various methods of data modeling exist, each using a variety of conventions and tools. The most popular approach is called the entity-relationship (ER) approach developed by Peter Chen in the late 1970s. The introduction of CASE tools such as Designer has introduced a number of diagrammatic conventions that the data modeler must master.
Entity Relationship Modeling

- Bottom-up approach
- Transaction processing
- 3rd Normal Form

Entity Relationship Modeling

Entity relationship modeling (ERM) is a powerful technique for designing transaction processing systems in relational environments. Data is normalized to remove all redundancy by using an age-old methodology called 3rd normal form. This methodology is driven by IT as a bottom-up approach and seriously violates the belief that business needs along with key business users drive the development of the warehouse.

The operational systems evolving from this methodology are developed for high performance throughput. However, the data does not always have the same meaning from operational system to operational system within the same organization. These transactional systems are designed to simplify update and insert operations that impact single records, and generally employ an ERD diagram (entity relationship diagram) to record the interaction of the entities. Relationships are normalized from many-to-many into disjoint tables, making analysis and decision support difficult.

This course does not focus on ERM (or the logical model). In constructing a relationship by this means, the logical becomes a high-level framework, whereas the physical is the implementation model, namely the metadata, fact tables, dimension tables, and so on. Additionally, dimensional modeling is a top-down approach that identifies and uses business processes, not data, as the core.
Entity Relationship Diagram

An ERD is a visual representation of the information requirements of an ERM system. It is a picture through which you model data. This diagram communicates data requirements, describes data, shows links between data, and serves as a basis for database design.

The ERD is made up of three components:

- **Entity** is a thing of significance about which information needs to be known or held.
- **Relationship** is a significant way in which two things are associated.
- **Attribute** is a piece of information that serves to qualify, identify, classify, quantify, or express the state of an entity.

Example:

- In a business, the customer places orders.
- Order and customer are both entities.
- Customer name and address are attributes of the customer entity.
- An order placed makes up the relationship between the customer and order entities.

**Note:** Oracle Warehouse Builder and Oracle Designer adhere to these diagrammatic standards.
Relationships Link Entities

Each ORDER must be for one and only one customer.

Each CUSTOMER may be the initiator of one or more orders.

Relationships Link Entities
A relationship signifies the association between two entities. Each end of the relationship represents how the entities are related.

Name of the relationship—Each end of the relationship has a name that enables you to understand the relationship easily, for example, an ORDER is for a CUSTOMER.

- The degree of a relationship—indicates how many entity instances may exist at one end of the relationship for each entity instance at the other end. A crow’s foot signifies a relationship end of degree many, while a single point signifies a relationship end of degree one. You can read the end of the relationship as being “one and only one.”
- The optionality indicates the minimum of entity instances that are possible at one end of the relationship for each entity instance at the other end. A broken line signifies an optional relationship end. You can read the broken line as “may be,” while a solid line signifies a mandatory relationship end. You can read the solid line as “must be.”
- Relationships are bi-directional, that is, you should be able to read the relationship starting at either end.

Formal rules for reading relationships:
Each entity1 { must be or may be } relationship name { one or more or one and only one } entity2
Summary

In this lesson, you should have learned how to do the following:

• Describe enterprise-level strategic analysis tasks
• Define components of business modeling
• Discuss the role of metadata in the data warehouse and methods of tracking metadata
• Define a logical model and entity relationship modeling
Practice 2-1

This practice covers the following topics:

• Creating a business model
• Defining and documenting the process flow diagram to represent the flow of activities within Vintners Ltd.

Practice 2-1
One of the best ways to learn concepts is to put the concept to use and develop a real-time model or case. The case study provided in Appendix A will facilitate this learning process by providing classroom exercises to reinforce the material delivered by the instructor. The case study depicts two hypothetical companies, one a traditional brick and mortar company, and the other, an internet company, who have agreed to merge. Company profiles are provided and some business assumptions are made up front.
In this practice, you answer a series of questions and then explore how to develop a business model as it relates to the merger of these two companies. 
Note: It is possible for several solutions to exist for this exercise. Solutions for this exercise are located in Appendix B.
Practice 2-1
Answer the following questions and discuss your results with other students.

1. What are the three tasks involved in business modeling?
   1. ___________________________________________
   2. ___________________________________________
   3. ___________________________________________

2. Why is strategic analysis essential?
   _____________________________________________
   _____________________________________________
   _____________________________________________
   _____________________________________________
   _____________________________________________

3. Many resources are available within your organization to aid your analysis and provide necessary information. Name at least four.
   1. ___________________________________________
   2. ___________________________________________
   3. ___________________________________________
   4. ___________________________________________

4. Why is an ODS not a data warehouse?
   _____________________________________________
   _____________________________________________
   _____________________________________________

5. Business ___________ are the success metrics of a business process, while business ___________ provide the metadata definitions or analytic parameters for the processes.

6. ___________ is the level of detail or the atomic level of the data warehouse.

7. An important activity while identifying business definitions and rules is ___________ metadata which provides a clear, coherent understanding of the metadata usage and definition.

8. ___________ modeling is a top down approach while ___________ ___________ modeling is a bottom up approach to design.
Practice 2-1 (continued)

Goal
Develop a business process model for accepting a purchase order from a customer for the newly formed Vintners Ltd.

Scenario
The management team for Vintners Ltd. has been assembled. Their recommendations, along with both companies’ existing system ERDs, are in Appendix A. The management team has decided that they would like to see a first cut, high level sales process model for their newly merged company. Furthermore, the management team also believes that a limited number of organizational units are necessary to provide this high level view. The business analysts have developed a business process matrix to assist you with future analytical tasks. This is the first cut matrix for the management team recommendations.

<table>
<thead>
<tr>
<th>Business Dimension</th>
<th>Business Process</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sales</td>
</tr>
<tr>
<td>Customer</td>
<td>X</td>
</tr>
<tr>
<td>Order</td>
<td>X</td>
</tr>
<tr>
<td>Warehouse</td>
<td>X</td>
</tr>
<tr>
<td>Product</td>
<td>X</td>
</tr>
</tbody>
</table>

Your Assignment
Your task is to review the two ERD diagrams for both organizations in Appendix A and determine which entities are needed. You must then create a process model. To assist you in developing the customer sales process model, answer the following question:

9. What four entities (organizational units) are most important for the development of the sales process model? (Hint: Use the high-level business matrix in Appendix A to answer this question.) Once you have answered the question, draw the model (use a page in your student note book).

___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________

10. Create the process model.
Creating the Dimensional Model

ILT Schedule: Timing Topic
90 minutes Lecture
30 minutes Practice
120 minutes Total

Instructor Note
It is very important to define the different models: business, dimensional, and physical. There is little debate about the definition of the dimensional and physical models, but there is much debate about the business or conceptual model and about the difference between the conceptual and dimensional models. The definition specified here is the Oracle definition for the dimensional and physical models. The Oracle Data Warehouse Method refers to the business model as business subject areas. In his book, The Data Warehouse Lifecycle Toolkit, Ralph Kimball refers to the business or conceptual model as the “Data Warehouse Bus Architecture matrix.” The rows of this matrix are all the possible subject areas (or potential data marts) and the columns are the dimensions needed for the business. The intersections where a dimension exists for a data mart or a subject area are marked. You can also find more information on this subject in Building the Data Warehouse, W.H. Inmon, Wiley Computer Publishing. ISBN 0-471-14161-5.
Objectives

After completing this lesson, you should be able to do the following:

• Describe attributes of a star model
• Identify fact tables and their attributes from business measure entities
• Identify dimension tables, attributes, and constraints from business dimension entities
• Describe slowly changing dimensions
• Define database keys for the data warehouse
• Describe hierarchies within the data warehouse
• Discuss analysis methods for the data warehouse

Lesson Aim
In the last lesson, you defined the business model and learned about the logical model. The work you do on the business model sets the stage for the next phase in the design process, developing the dimensional model (also known as the star model). The goal of this lesson is to show how to translate the business model into a dimensional model. A dimensional model, unlike an Entity Relationship Model, packages the data in a symmetric format that enables business user understanding, query performance, and resistance to change.
Data Warehouse Database Design Objectives

• Easy to understand
• Optimum performance
• Adaptable

Data Warehouse Database Design Objectives

• You must provide an easy-to-understand structure, with good-quality metadata that helps users navigate around the database and interpret the available information easily.
• You must ensure optimum performance for all types of queries, particularly complex queries and data mining.
• You must make the structure as easy to adapt as possible.
• You must have a predetermined plan for updating data, for example, slowly changing dimensions.
Data Warehouse Data Types

<table>
<thead>
<tr>
<th>Data Warehouse Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact (detail, atomic, raw) base data</td>
<td>Business measures</td>
</tr>
<tr>
<td>Dimension data</td>
<td>Query drivers</td>
</tr>
<tr>
<td>Derived fact data</td>
<td>Calculated data</td>
</tr>
<tr>
<td>Summary (aggregate) data</td>
<td>Pre-calculated data</td>
</tr>
<tr>
<td>Metadata</td>
<td>Warehouse map</td>
</tr>
</tbody>
</table>

Data Warehouse Database Tables
The data warehouse tables contain various types of data:

- Fact data, often called detail, base, atomic, or raw data, are the numerical measures of the business.
- Dimension data offers an attribute by which data can be analyzed.
- Derived data is a calculation involving base facts.
- Summary or aggregate data is highly or lightly summarized fact data.
- Metadata is descriptive data about the data in the warehouse.
Designing the Dimensional Model

Phase III: Defining the dimensional model

Designing the Dimensional Model
The database design process begins with the enterprise view of the business and the specific subject areas that are to be implemented. These business needs determine the business model (sometimes referred to as the conceptual model or subject areas), which is a representation of business subjects and relationships. The dimensional modeling process is a top-down design approach.

The business model translates into a more specific design called the dimensional model (sometimes referred to as a star model). This is where you identify the fact and dimension tables and the relationship between them, and determine the grain of both fact and dimension tables. The potential hierarchies of the dimension are also identified. The outcome of the dimensional model is usually the star or snowflake model. The dimensional design is the foundation of the database design for the data warehouse.

Note: For the purpose of this course, we refer to the dimensional model as the star dimensional model.
Star Dimensional Modeling

Star Dimensional Model

Star dimensional modeling is a logical design technique that seeks to present the data in a standard framework that is intuitive and provides high performance. Every dimensional model is composed of one table called the fact table, and a set of smaller tables called dimension tables. This characteristic, denormalized, star-like structure is often called a star model. Within this star model, redundant data is posted from one object to another for performance considerations. This is called transference or inheritance.

Important: A fact table has a multipart primary key composed of two or more foreign keys and expresses a many-to-many relationship. Each dimension table has a single-part primary key that corresponds exactly to one of the components of the multipart key in the fact table.

There are a variety of models designed for decision support systems (DSS). In your exercises, you will use the star as the dimensional modeling vehicle for understanding database design and implementing the data warehouse.

Note: Two other variations of the star dimensional model such as the snowflake and the constellation models will be discussed later in this lesson.

Instructor Note

Point out to the students that the star, snowflake, and constellation dimensional models will be explained in more detail later in this lesson. This page along with the next page are meant to be an overview of the star dimensional model.
Advantages of Using a Star Dimensional Model

- Supports multidimensional analysis
- Design improves performance
- Optimizers yield better execution plans
- Parallels end-user perceptions
- Provides an extensible design
- Broadens the choices for data access tools

Advantages of Using a Star Dimensional Schema
- Provides rapid analysis across different dimensions for drilling down, rotation, and analytical calculations for the multidimensional cube.
- Creates a database design that improves performance
- Enables database optimizers to work with a more simple database design to yield better execution plans
- Parallels how end users usually think of and use the data
- Provides an extensible design which supports changing business requirements
- Broadens the choices for data access tools, because some products require a star schema design

Instructor Note
The slides list the high level advantages of using a star dimensional model. More specific advantages and the characteristics of the star model are listed later in this lesson.
Analyze Source Systems for Additional Data

Data Sourcing Data sourcing focuses on the meaning of the data by identifying the source files and fields that support the business objectives and requirements of the data warehouse target system. The deliverable is the correct source files and fields that fulfill the business objective of the target system.

You must perform a complete source data analysis before creating your star dimensional model. You should examine the source data structures to identify all of the measures and dimension attributes necessary to support the analytical requirements communicated by business users. Often, you will find additional required data elements in the source system information that were not identified in the business requirements analysis.

Source system information resources include:
- Source ER diagrams from OLTP production systems
- Legacy systems data
- Flat file systems
- Existing metadata
- Spreadsheets
- Researched reports
Analyze Source Data Documentation (Metadata)

Assess the quality of existing source data documentation:

- Completeness
- Depth
- Accessibility
- Currency
- Consistency

Instructor Note

For the purpose of this course, we are providing a quick overview of data sourcing as part of the extraction, transformation, and transportation (ETT) process. Typically, you should perform some data consolidation and conversion on some of the source data as part of the ETT process. ETT is explained in more detail in the Data Warehouse Fundamentals ILT course.
Fact Tables

- Contain numerical metrics of the business
- Can hold large volumes of data
- Can grow quickly
- Can contain base, derived, and summarized data
- Are typically additive
- Are joined to dimension tables through foreign keys that reference primary keys in the dimension tables

Fact Tables

Facts are the numerical measures of business performance. The fact table is composed of large volumes of data. Base (also referred to as raw or atomic) facts, such as dollar sales, can be combined or calculated with other facts to create measures. Measures can be stored in the fact table or created when necessary for reporting purposes. Many fact tables exist in a warehouse. Summary tables that hold precalculated values are also referred to as fact tables. Fact tables:

- Contain the numerical measurements of the performance of the business, such as sales figures, customer banking transactions, and units sold (Each of these measurements is taken at the intersection of all of the dimensions.)
- Contain many rows (possibly millions)
- Can grow quickly in size (how quickly and by how much depends on the amount of data that is introduced into the warehouse every refresh, and by the amount of history that is required.)
- Can contain base (atomic or raw), derived, and summarized data (aggregated data)
- Can hold large volumes of data
- Are typically additive
- Are joined to dimension tables through foreign keys that reference primary keys in the dimension tables
Factless Fact Tables

- Event tracking
- Coverage

Factless Fact Tables
A factless fact table is a fact table that does not contain numeric additive values, but is composed exclusively of keys. There are two types of factless fact tables: event-tracking and coverage.

Event-tracking records and tracks events that have occurred, such as college student class attendance, whereas coverage factless tables support the dimensional model when the primary fact table is sparse, for example, a sales promotion factless table. In the latter case, the events did not occur. Although promotional fact data can be stored within the fact table itself, creating a coverage factless fact table is far more efficient because the complex many-to-many relationship formed through the dimensions for a promotion require massive amounts of detail with zero detail.
More on Factless Fact Tables

The factless fact table represents the many-to-many relationships between the dimensions, so that the characteristics of the event can be analyzed. Materialized views are generally built on factless fact tables to create summaries. Materialized views, a powerful tool within Oracle Warehouse Builder, are discussed in lesson 7.

Examples

- Human resources—Studies of the labor force composition are often conducted for reporting and planning purposes. Analysis of employees with different characteristics can be conducted using the illustrated star.
- Most of the resulting information from this kind of table is a series of counts. In the example illustrated selecting `COUNT(Emp_FK)` gives the number of employees, whereas selecting `COUNT(Sal_FK)` gives the number of employees on a specified salary grade.
- Retail store—Promotions are typical within the retail environment. An upscale retail chain wants to compare its customers who do not respond to direct mail promotion to those who make a purchase. A factless fact table supports the relationship between the customer, product, promotion, and time dimensions.
- Student attendance—Factless fact tables can be used to record student class attendance in a college or school system. There is no fact associated with this; it is a matter of whether the students attended.

Note: FK = foreign and PK = primary key
Identify Base and Derived Measures

- Identify candidate facts
- Remove duplicate facts
- Discover and document the underlying calculations
- Cross reference base facts
- Obtain final derived fact approval

Identify Base and Derived Measures
Use the following process to identify the base and derived facts. The steps include:

- Identifying candidate facts—Identify all possible base and derived facts. Collect all reports submitted in the interview process. You can compile a spreadsheet to contain all of the facts that were listed on reports or requested in the interviews. Capture the fact name, where it was identified (report name or interview), and the name of the person interviewed or providing the report.
- Removing duplicate facts—Eliminate any duplicate facts. You can group the duplicate facts together and then work with business users to remove duplicates. You also identify base versus derived facts.
- Discovering and documenting the underlying calculations—After identifying the derived facts, document the formula to be used in the calculations.
- Cross-referencing base facts: You must ensure that all the base facts that are needed in the calculations are included.
- Obtaining final derived fact approval—You should ensure that the business users and management approve the list of the derived facts that you have identified.
Base and Derived Data

The fields in your fact tables are not just source data columns. There may be many more columns that the business wants to analyze, such as year-to-date sales or the percentage difference in sales from a year ago. You must keep track of the derived facts as well as the base facts.

**Derived Data**

Derived data is data that is calculated or created from two or more sources of data. A derived value can be more efficiently stored for access, rather than calculating the value at execution time. Derived data in the warehouse is very important because of its inherent support for queries. When you store derived data in the database, values are available immediately for analysis through queries.

In this example, you see the employees’ monthly salary and monthly commission figures. In the OLTP system, you would not store the monthly compensation, but you might do so for the warehouse. Derived values can be calculated at an instance in time, as with an OLTP system. More derived values feature prominently in data warehouses.
### Translating Business Measures into Fact Tables

The slide shows the translation of some of the business measures in a business model into an Order fact table. In addition, each measure in the fact table is identified as either a base or derived measure, for example, Profit = Order_amount – Order_cost. Derived values can be created during the extraction or transformation processes.

<table>
<thead>
<tr>
<th>Business Measures</th>
<th>Fact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Items</td>
<td>Base</td>
</tr>
<tr>
<td>Amount</td>
<td>Base</td>
</tr>
<tr>
<td>Cost</td>
<td>Base</td>
</tr>
<tr>
<td>Profit</td>
<td>Derived</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fact</th>
<th>Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Items</td>
<td>Base</td>
</tr>
<tr>
<td>Item Amount</td>
<td>Base</td>
</tr>
<tr>
<td>Item Cost</td>
<td>Base</td>
</tr>
<tr>
<td>Profit</td>
<td>Derived</td>
</tr>
</tbody>
</table>
Fact Table Measures

Additive: Added across all dimensions

Semiadditive: Added along some dimensions

Nonadditive: Cannot be added along any dimension

Fact Table Measures

Additive  Additive data in a fact table can be added across all of the dimensions to provide an answer to your query. Additive facts are fundamental components of most queries. Typically, when you query the database you are requesting that data be summed to a particular level to meet the constraints of your query. Additive facts are numeric and can be logically used in calculations across all dimensions. Additive data examples are units sold, customer account balance, and shipping charge.

Note: You should choose the facts in a fact table to be numeric and additive (more useful).

Semiadditive  Semiadditive data can be added along some but not all of the dimensions such as a bank account balance. The bank records balances at the end of each banking day for customers, by account, over time. This allows the bank to study deposits, as well as individual customers. In some cases, the account balance measure is additive. If a customer holds both checking and savings accounts, you can add together the balances of each account at the end of the day and get a meaningful combined balance. You can also add balances together across accounts at the same branch for a picture of the total deposits at each location; however, you cannot add together account balances for a single customer over multiple days.

Nonadditive  Nonadditive data cannot logically be added between records. Nonadditive data can be numeric and is usually combined in a computation with other facts (to make it additive) before being added across records. Margin percent is an example of a nonadditive value. Nonadditive are also factless fact tables.
Dimension Tables

• Contain descriptors of the business
• Contain relatively static data
• Contain textual and discrete data
• Are usually smaller than fact tables
• Are joined to a fact table through a foreign key reference

Dimension Tables
Dimension tables provide the analytical perspective for all business processes. They contain the descriptors for a specific business dimension, called dimension attributes. Dimension tables are relational database tables with a single part key that is joined to a fact table. Dimension tables have the following characteristics:
• Identify and provide user-oriented descriptions business dimension attributes
• Provide descriptive information about each row in the fact table and are typically textual, providing documentation for constraints and row headers
• Typically are geometrically smaller than the fact table
• Contain data that generally changes much less frequently than fact table data. Whereas dimension tables contain relatively static data, certain dimensions, such as customer, change more frequently.
• All warehouses usually include a time dimension table

Dimension Attributes The basis for a powerful star dimensional model lies in the richness of the dimension attributes because they determine how facts can be analyzed. Dimension attributes can be considered as the entry point into “fact space,” because they are the source of constraints for analytical queries and provide values for pick lists in query tools. If you name attributes in the users’ vocabulary, the dimension documents itself, and its expressive power is apparent.
Translating Business Dimensions into Dimension Tables

Translating a list of business dimension attributes into a dimension table is not a simple one-to-one mapping process. As described previously, you must understand that source data structure to identify all of the source data elements that need to be included in the warehouse in order to support user analytical requirements.

For example, during a business requirements analysis for an organization, a number of attributes are uncovered for the product business dimension. Product, category, and type are three of the required attributes. Upon examination of the source system ERD, it is determined that product really consists of two data elements: product ID and product description.
Translating a Product Business Dimension into a Product Dimension Table

Business Dimension
- Product
- Category
- Supplier
- Type
- Warehouse
- Promotion
- Catalog

Business Dimension for Product
- Product_Id (Natural Key)
- Product_name
- Product_desc
- Category
- Supplier_Id
- Product_status
- List_price
- Catalog_Id
- Product_type
- Product_code
- Promotion_Code
- Warehouse_location

Translating the Product Dimension into the Product Dimension Table
- A list of attributes for this business dimension are uncovered during the business modeling phase.
- All of the product related source data information is gathered and used to help translate the business dimension requirements into dimension table attributes.
- In the example, “ID” fields (codes that uniquely identify product attributes) and description fields are available in the source data, and should be included in the warehouse so that users familiar with the production system can cross-reference the source system ID fields with warehouse data elements.
Updating Dimension Data

Dimension data is not refreshed in the same way as fact data. Over a given number of years, the dimension data changes. This is referred to as a slowly changing dimension. The data warehouse must accurately describe the history of the data.

There are three ways to manage slowly changing dimensions:

- Overwrite the existing dimensional attribute with the change. This does not affect the keys, nor does it insert records.
- Add a new record each time the dimension data changes. This preserves the history of the old record and accurately partitions history across time, however, a significant increase to the database’s size is incurred.
- Examine the grain of the data very closely when first designing the data warehouse; otherwise partitioning of the data will not occur properly.
- Caution should be used when implementing this change. If an ID is used as the primary key, a potential integrity violation could occur.
- Preserve the current record information, but include some critical fields when initially designing the data warehouse. These fields retain previous and current information, and should include a time attribute to signify when the change occurred thus allowing a history of the change to be preserved. It should be evident that this method increases complexity and the component size of the table.
Slowly Changing Dimensions

A typical example of a slowly changing dimension is product and SKU. Simply adding a date-time stamp to the product does not work because this only manages the version of the product. By adding an SKU “from” date (valid_from_date) and a new field to hold the new SKU (product_PK), the data warehouse can track the changing dimension and activity within the fact table. Examine the additional fields added to the Product dimension. product_PK is a surrogate key that maintains the old and new identities of the item and product_PK_date stipulates when the change occurred for reporting purposes.

**Original fact record:**

<table>
<thead>
<tr>
<th>Product_PK</th>
<th>Product_Id</th>
<th>Product_Desc</th>
<th>Valid_from_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>6241</td>
<td>Stags Leap</td>
<td>01312000</td>
</tr>
</tbody>
</table>

**Changed fact record:**

<table>
<thead>
<tr>
<th>Product_PK</th>
<th>Product_ID</th>
<th>Product_Desc</th>
<th>Valid_from_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>6241</td>
<td>Stags Leap</td>
<td>02032001</td>
</tr>
</tbody>
</table>

A query to retrieve both records might be written as follows:

```sql
<fact>.Product_PK = <dimension>.Product_PK and 
<dimension>.Title_ID = 'Seven';
where <fact>.Product_PK in (select <dimension>.Product_PK
from <dimension>
    where Title_ID = 'Stags Leap');
where <fact>.Product_PK in (32, 64)
```
Types of Database Keys

1. **Primary keys (PKs)**
2. **Foreign keys (FKs)**
3. **Composite keys**
4. **Surrogate keys**

**Types of Database Keys**

**Primary Key**: A logical or natural primary key is the column or columns that uniquely identify a row in a table. A primary key constraint is used to enforce uniqueness for all rows in that table. It is also critical to performance, granting quick access compared to the unacceptable amount of time that would be required if the RDBMS had to scan the entire table every time a row was queried. The choice of primary key is an important design consideration, because it forces data integrity and eliminates data duplication within the table.

**Foreign Key**: This key column in a table references a primary key for another table, establishing relationships between tables.

**Composite Key**: The composite key consists of a number of columns. In the case of a concatenated primary key, combinations of values between the columns must be unique. These keys are sometimes referred to as concatenated or segmented keys. All keys identified here are important for the efficiency of all systems, whether the systems are operational or warehouse. Composite keys are commonly used in the warehouse.

**Surrogate Key** (Warehouse, Synthetic, or Generalized keys): A surrogate key is a system-generated key. The key itself has no meaning, and therefore, you cannot ascertain anything about the underlying dimension to which it is associated. Generally speaking, a 4-byte integer is sufficient (containing $2^{32}$ values or more than two billion positive integers) when assigning attributes for the key. An example of a system-generated key is the ROWID value that is generated by the Oracle server for every row inserted into the database. You should use surrogate keys instead of natural keys in the data warehouse. Include the surrogate key in the dimension table in addition to the natural key. Then use the surrogate key in the fact table as the column that joins the fact table to the dimension table.
Using Surrogate Keys

Advantages of surrogate keys include:
• Control over data
• Reduced fact table size

Avoid using the following as data warehouse keys:
• OLTP Production (natural) keys
• Smart keys (embedded meaning)

Using Surrogate Keys

Smart (Intelligent) Keys Smart keys are values that have embedded meaning such as the value of a book ISBN number. These keys present a challenge to the data warehouse developer when defining extraction and transformation rules for the data. Often the real meaning of the values has been lost over time. These values must be broken down into the key’s individual components (meanings) for the warehouse. Intelligent keys are not recommended for the data warehouse.

Advantages of Using Surrogate Keys Instead of Production Keys Natural keys are often long character strings, such as a product code whereas surrogate keys are integers. As a result, response time to queries is improved. Additionally, surrogate keys can significantly reduce the size of fact tables.

The data warehouse administrator retains control over the surrogate key, even if the keys change in the operational system. If a manufacturing group changes the product-code naming conventions, for example, the changes do not affect the structure of the data mart.
Surrogate Keys Example

<table>
<thead>
<tr>
<th>Emp_FK</th>
<th>Salesperson_ID</th>
<th>Salesperson_Name</th>
<th>Manager_ID</th>
<th>Emp_Change_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Smith</td>
<td>200</td>
<td>030199</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Jones</td>
<td>300</td>
<td>050599</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Harvey</td>
<td>300</td>
<td>060599</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>Smith</td>
<td>400</td>
<td>061001</td>
</tr>
</tbody>
</table>

Surrogate Keys

<table>
<thead>
<tr>
<th>Prod_FK</th>
<th>Prod_ID</th>
<th>Prod_Name</th>
<th>Prod_Grouping</th>
<th>Brand_Code</th>
<th>Prod_Change_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>073258</td>
<td>Coffee</td>
<td>Hot</td>
<td>YUBN</td>
<td>032200</td>
</tr>
<tr>
<td>2</td>
<td>073258</td>
<td>Coffee</td>
<td>Hot</td>
<td>MAXH</td>
<td>110100</td>
</tr>
<tr>
<td>3</td>
<td>011172</td>
<td>Pop</td>
<td>Cold</td>
<td>SCHW</td>
<td>061001</td>
</tr>
<tr>
<td>4</td>
<td>011173</td>
<td>Tea</td>
<td>Hot</td>
<td>RRSE</td>
<td>061001</td>
</tr>
</tbody>
</table>

Surrogate Keys

In some instances the methods you use to maintain historical could potentially allow duplicate keys. To ensure uniqueness a surrogate key is generated. In the warehouse this key is generated by the program that maintains the data, and is application generated. As the data warehouse administrator, you assign the first key the value 1, the next value 2, and so on. You assign the key as you encounter the need for it. The keys themselves have no meaning. If all you have is the key for a dimension, you cannot ascertain anything about the underlying record. You must use the key to retrieve the record to see what it contains.

Example: In the employee table shown on this slide, Smith has changed managers. It is necessary to maintain history, but by duplicating his employee ID (100) we lose data integrity. The Emp_FK now uniquely identifies this Smith record.

Similarly, with the product example: The Prod_FK uniquely identifies Prod_Name. It does not uniquely identify the Prod_Name and Brand_Code affiliation. Therefore, Prod_ID cannot be used as a primary key in this example.

In both examples, date fields have been added to reflect slowly changing dimensions.
Adding a Surrogate Key

Based on the interview with business users during the business modeling phase, the business rules and the query/information requirements can be used to identify the hierarchies needed for each dimension. For example, the Sales dimension has a hierarchy that enables users to drill down through channel to promotion to product, but exclusive of product. These are also considered foreign keys.
Bracketed Dimensions

- Enhance performance and analytical capabilities
- Create groups of values for attributes with many unique values, such as income ranges and age brackets
- Minimize the need for full table scans by pre-aggregating data

Bracketed Dimensions

To enhance performance and analytical capabilities, create bracketed dimensions for categorical attributes. Creating groups of values for attributes with many unique values, such as income, reduces the cardinality (fewer bitmap indexes) and creates ranges that are more meaningful to the end user.

For example, even though the actual income amount is available at the customer level, the business requirements do not need discrete values for analysis. Income can then be grouped in $10,000 ranges. Similarly for ages you can have a 21–30 bracket and a 31–40 bracket. To extract a person’s actual age, you can perform a calculation on birth date.

Attributes can be grouped into brackets as well, creating unique identifiers for combinations of attributes.

For some queries, preaggregating the data can minimize the need for full table scans. The bracketed definition is selected as the query constraint and the index for the table is scanned for records that satisfy the query. If the data was not preaggregated into brackets, the table must be scanned multiple times to select each range of values.

Assumptions:
- There are far fewer brackets than actual occurrences of the value bracketed
- The ranges are agreed upon
- Values that will constitute summary tables should be preaggregated.
Bracketing Dimensions

Bracketing dimensions are typically used to support complex analytical models, because they:
• Contain information used to classify, categorize, or group multiple attributes
• Are frequently used in end-user queries where the actual value is too discrete to be meaningful

Therefore, consider preaggregating to enhance performance and analytical power. If you have the data, you can use a query tool or data mining tool to determine and define the brackets based on actual values.

If connected to the fact table, the Bracket_PK field must be part of the fact table composite key; otherwise, no modifications to the fact table are required. If the bracketed dimension is not connected to the fact table, it can be built at anytime, as required.

In this example, the bracketed dimension is created from a combination of attributes. The bracketed table is related to the fact table, requiring that the Bracket_PK field be part of the composite key. The example also presupposes that these brackets have been agreed upon during requirement analysis. Examples of other applications for which you might use bracketing include:
• Marketing applications with customer salary ranges
• Risk assessment applications containing credit card limits

Note: Bracketing data in this way makes it more difficult to manage, load, or access with drill capabilities.
Models for Hierarchical Data

• Analytical activities using hierarchies are supported through different models:
  – Business
  – Multiple
  – Multiple time

• Hierarchical data is stored in dimension tables.

• Dimensions can contain one or more hierarchies.
Identifying Analytical Hierarchies

Business hierarchies describe organizational structure and logical parent-child relationships within the data.

Example
In the example on the slide, a store dimension has one hierarchy, the organization hierarchy.
Multiple Hierarchies

Multiple hierarchies can be stored in a single dimension, using the flat model approach. The definition of the relationship between the data in this model must be well maintained and stored in the metadata.

All the levels of the hierarchy have been collapsed into one table.

Example: Multiple hierarchies within the same dimension are so common that they may be the rule rather than the exception. In this example, the store dimension has two hierarchies:

- Organization
- Geography

These relationships were most likely created to manage a particular business process. The organization hierarchy may support the evaluation of sales groups or personnel, whereas the geography hierarchy may be used to manage distributors or estimate tax exposure.
Multiple Time Hierarchies

Representation of time is critical in the data warehouse. You may decide to store multiple hierarchies in the data warehouse to satisfy the varied definitions of units of time. If you are using external data, you may find that you create a hierarchy or translation table simply to be able to integrate the data. Matching the granularity of time defined in external data to the time dimension in your own warehouse may be quite difficult.

A simple time hierarchy corresponds to a calendar approach: days, months, quarters, years. A hierarchy based on weeks seems fairly simple as well: weeks, four-week period. What is the definition of a week? Does the week start on Sunday or Monday? Internally, you may define it one way; however, when you try to integrate external data that is defined in a different way, you may get unexpected or misleading results.

Are there not 13 weeks in a quarter? Why can I not map 13-week periods to a quarter? Typically the start and stop date of a quarter corresponds to a calendar date—the first of the month, the last day of a month. Thirteen-week periods may start at any time but are usually consistent with the start day of the week definition.

Example: The time dimension is described by multiple hierarchies and can be used to support both calendar and fiscal year.
Drilling Up and Drilling Down

Drilling refers to the investigation of data to greater or lesser detail from a starting point. Typically, in an analytical environment you start with less detail, at a higher level within a hierarchy, and investigate down through a hierarchy to greater detail. This process is drilling down (to more detailed data). Drilling down means retrieving data at a greater level of granularity. Using the market hierarchy as an example, you might start your analysis at the group level and drill down to the region or store level of detail.

Drilling up is the reverse of that process. Consider the market hierarchy example. If your starting point were an analysis of districts, drilling up would mean looking at a lesser level of detail, such as the region or group level.

Drilling can occur in three ways:

- Through a single hierarchy
- From a single hierarchy to nonhierarchical attributes
- From a single hierarchy to another hierarchy

The example of the market hierarchy displays one drill path.
Drilling Across

Beginning the analysis with the group level of the market hierarchy, you drill down to a region within the same hierarchy. Your analysis then leads you to find all stores (part of the market hierarchy) with greater than 20,000 square feet (a nonhierarchical attribute). Having identified those stores, you now want to identify the cities in which those stores are located. In this analysis you have gone from drilling in one hierarchy, to an independent attribute, and finally to another hierarchy.

The risk in drilling across is that you are not necessarily going to enter another hierarchy at the same level, so the totals may be different. Therefore, you would not try to balance the results from your final query, which looked at the city level within the geography hierarchy, with totals at different levels within the market hierarchy.

Drilling across can be a very powerful approach to analysis. If dimensions have been conformed, for example, the grain and detail are identical, reliability for drilling across is greatly enhanced.
Documenting the Granularity of Dimensions

• Is an important design consideration
• Determines the level of detail
• Is determined by business needs

Low-level grain
(Transaction-level data)

High-level grain
(Summary data)

Granularity
The single most important design consideration of a data warehouse is the issue of granularity. Granularity, or the grain, determines the lowest level of detail or summarization contained in the warehouse. A low level of granularity means greater detail; a high level of granularity means less detail. Granularity affects the warehouse size and the types of queries that can be performed (the dimensionality of the warehouse). Additionally, the type of query that is required determines the granularity of the data. The data can be stored at different levels of granularity: individual transactions, daily snapshots, monthly snapshots, quarterly snapshots, or yearly snapshots. Maintaining a low level of grain in the warehouse is expensive and requires more disk space and more processing in access operations. Because the data may even exist at the transaction level, there is less demand during the ETT process than there would be with a higher level of granularity. A high level of granularity requires less space and fewer resources for access, but may prevent users from accessing the level of detail that they require to answer their business questions. Consider starting your data warehouse with the lowest granularity possible that is available from the source systems. After the warehouse is running, users can determine what level of granularity is necessary. Keep in mind however, that different users have different requirements.
Defining Time Granularity

Fiscal Time Hierarchy

- Fiscal Year
- Fiscal Quarter
- Fiscal Month
- Fiscal Week
- Day

Current dimension grain
Future dimension grain

Defining Time Granularity

The grain you choose for the time dimension can have a significant impact on the size of your database. It is important to know how your data has been defined with regard to time, to accurately determine the grain. This is particularly true with external data that is commonly aggregated data. Even when you think there is no gap in the granularity of your systems, you may find that basic definitions between systems differ.

The primary consideration here is to keep a low enough level of data in your warehouse to be able to aggregate and match values with other data that has a predetermined aggregate. The most common use of partitioning for a data warehouse is to range-partition the fact table of the star schema by time providing huge benefits in querying, manageability, and performance when loading.

Note: A good practice is to store data at one level of granularity lower than your business user has requested. For example, if the user requests monthly data, you may want to store weekly data. However, the impact of the increased storage can be prohibitive.
A Star Dimensional Model

The Star Model
- Central fact table
- Radiating dimensions
- Keys link dimensions and fact
- Queries join dimensions to fact

The star model is a database design. It is a central fact table consisting of a series of atomic values, which reside in columns. The fact keys join to a number of single-level tables, known as dimension tables. This model is a good starting place for your warehouse design. In this model, normal relational rules apply, with primary and foreign keys linking the fact and dimension tables. When a user issues a query against a star model, the dimension data constrains and drives the query.

Creating a bitmap index on each foreign key column in the fact table allows you to take advantage of star join transformations. When a query is issued that joins the dimension tables to the fact table containing the foreign keys with bitmap indexes, Oracle first uses the specified WHERE clause criteria to limit the rows to form each dimension. The primary key values of the remaining dimension table rows that are then compared to the bitmap indexes to identify the fact table rows needed using the efficient bit-merge operations to resolve these sets into a single result set of qualifying rows. The final step is to join that result set back to each dimension table, using the most appropriate join technique (nested loop, hash join, or sort merge). To allow Oracle to use bitmap indexes to perform this star query, the STAR_TRANSFORMATION_ENABLED parameter must be set to TRUE. Bitmap indexes will be discussed in lesson 5.
Star Dimensional Model Characteristics

- The model is easy for users to understand.
- Primary keys represent a dimension.
- Nonforeign key columns are values.
- Facts are usually highly normalized.
- Dimensions are completely denormalized.
- Fast response to queries is provided.
- Performance is improved by reducing table joins.
- End users can express complex queries.
- Support is provided by many front-end tools.

Star Model Characteristics

- Each foreign key column on the fact table represents a dimension.
- The nonprimary key columns in the fact table are values that can be aggregated. Fact tables do not contain character values; these belong with the dimensions.
- The star model structure is similar to how the users understand the information.
- The model provides better performance for analytical queries by reducing the number of joins.
- It allows complex queries to be expressed by end users, because the data is arranged in a way that is easy to understand and the relationships between entities are very clear.
- It restricts the numerical measurements of the business to the fact table.

Note: The definitions of star and snowflake models vary among practitioners. Here the assumption is that the star model contains a fact table with one level of related dimensions. An example is sales fact and product dimension. The snowflake, on the other hand, has more than one level of dimension; that is, a hierarchy, for example, Sales Fact, Product Dimension, and Product Group.
Snowflake Model

- Is an extension of a star design
- Is a normalized star model
- Can cause potential performance degradation

A dimension is said to be *snowflaked* when the low cardinality fields in the dimension have been removed to separate tables and linked back to the original tables with artificial keys. The snowflake model is closer to an entity relationship diagram than the classic star model, because the dimension data is more normalized. Developing a snowflake model means building hierarchies from each dimension.

The snowflake model is an extended, more normalized star model. The dimensions are normalized to form class hierarchies. For example, a Location Dimension may be normalized to a hierarchy showing Countries, Counties, and Cities; or Region, District, and Office. The slide shows an example where the Order dimension is normalized to a hierarchy of Channel and Web. The two tables are then joined together.

**Note:** We do not recommend using snowflaking because usually this model almost always interferes with the user’s comprehension. The exception is when a user access tool requires the use of a snowflake model.
**Constellation Configuration**

The constellation configuration consists of a central star surrounded by other stars. The central star comprises base-level or atomic data. The surrounding stars can be stars that share dimensions or summary data. The surrounding stars can share dimension attributes with the atomic star.

A galaxy is composed of those constellations that do not share dimensions and a universe is composed of galaxies.

**Note:** Summaries will be discussed in detail in lesson 7, *Summary Management.*
Updating the Meta Data

• Dimensions and attributes detail (Primary key, attribute definition, and so on)
• Facts and measures detail (Measure description, additivity, and so on)
• Data source definitions (Business owner, platform, description, and so on)
• Source to target data mappings (Data type, length, target column description, and so on)

Updating the Metadata
During the dimensional modeling task, you have identified numerous attributes to include in the repository. Some of these are listed below:
• Dimensions
  – Dimension or attribute name—The official business attribute name
  – Attribute definition: A brief but meaningful description of the business attribute
  – Primary key—An indication whether this attribute is a primary key
  – Grain: The granularity of each dimension is identified
  – Hierarchy—Whether or not a dimension has an analytical hierarchy and shows the levels of the detail in the hierarchy
  – Sample values—Sample data that this attribute will contain
• Fact and measures
  – Fact or measure name—The name of the base or derived fact as will be seen in the data access tools and on reports
  – Measure description—A brief description of the fact or measure
  – Based or derived—Indicates whether a fact is based or derived
  – Additivity—Indicates whether a fact is additive, semadditive, or nonadditive
  – Formula—Identifies the formula used in the calculation of a derived fact
Summary

In this lesson, you should have learned how to do the following:

• Describe attributes of a star model
• Identify fact tables and their attributes from business measure entities
• Identify dimension tables, attributes, and constraints from business dimension entities
• Describe slowly changing dimensions
• Define database keys for the data warehouse
• Describe hierarchies within the data warehouse
• Discuss analysis methods for the data warehouse
Practice 3-1 Overview

This practice covers identifying star model concepts.

Practice 3-1
Solutions for this exercise are located in Appendix B.
Practice 3-1 (continued)
Answer the following questions.

1. What are the four essential data types found in a data warehouse?
   1. _____________________________________________
   2. _____________________________________________
   3. _____________________________________________
   4. _____________________________________________

2. What are the two table types that comprise a star dimensional model?
   1. _____________________________________________
   2. _____________________________________________

3. Fact tables are joined to dimension tables through _______________ keys that reference _______________ keys within the dimension.

4. Explain the difference between additive, semi-additive, and non-additive fact tables.
   ______________________________________________
   ______________________________________________
   ______________________________________________
   ______________________________________________
   ______________________________________________
   ______________________________________________
   ______________________________________________
   ______________________________________________
Practice 3-1 (continued)

5. What are the three ways to handle slowly changing dimensions?

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

6. Explain why surrogate key use is important in a data warehouse.

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

7. What is a bracketed dimension?

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

8. ________________ determines the lowest level of detail in a data warehouse and is considered to be the single most important design consideration.

9. A __________ ____________ contains a fact table with one level of related dimensions.
Practice 3-2 Overview

This practice covers the following topics:

• Creating a dimensional model
• Defining the entities and the relationships between entities forming the basis for the data warehouse database design
• Identifying dimension and fact tables

Practice 3-2

Solutions for this exercise are located in Appendix B.
Practice 3-2 (continued)

Diagram Assistance

Entity relationship modeling involves identifying the things of importance in an organization (entities), the properties of those things (attributes) and how they are related to one another (relationships). The resulting information model is independent of any data storage or access method.

The diagram is made up of components:

- **Entity** is a thing of significance about which information needs to be known or held.
- **Relationship** is a significant way in which two things are associated. Each end of the relationship has a name that enables you to understand the relationship easily, for example, an ORDER is for a CUSTOMER.
  - The **degree** of a relationship—indicates how many entity instances may exist at one end of the relationship for each entity instance at the other end. A crow’s foot signifies a relationship end of degree *many*, while a single point signifies a relationship end of degree *one*. You can read the end of the relationship as being “one and only one.”
  - The **optionality** indicates the minimum of entity instances that are possible at one end of the relationship for each entity instance at the other end. A broken line signifies an optional relationship end. You can read the broken line as “may be,” while a solid line signifies a mandatory relationship end. You can read the solid line as “must be.”
- **Attribute** is a piece of information that serves to qualify, identify, classify, quantify, or express the state of an entity.
Practice 3-2 (continued)

Goal
Develop a star model for the business process model created in practice 2-1.

Scenario
The management team has reviewed the business process model. They are now recommending that you go a step further and develop a basic star dimensional model. They also want to remind you that this is not an order entry system and that they want you to create a customer sales model.

After considerable analysis and clarification, you have determined the following:
• A customer must own a purchase order
• Many orders can exist for a customer
• A warehouse must own a product
• Many products exist in a warehouse
• Sales is dependent upon the order and contains high-level order information

Your Assignment
Draw a simple star model in your student notebook after determining the participating entities.

Hint: Order_Item, although important in a data entry system or an operational data store, is not pertinent for the overall customer sales process.

If you are having difficulty determining the entities, use the table on the next page to develop the star model.
**Practice 3-2 (continued)**

**Table Definitions**

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMER</td>
<td>Dimension</td>
</tr>
<tr>
<td>ORDER</td>
<td>Dimension</td>
</tr>
<tr>
<td>WAREHOUSE</td>
<td>Dimension</td>
</tr>
<tr>
<td>PRODUCT</td>
<td>Dimension</td>
</tr>
<tr>
<td>SALES</td>
<td>Fact</td>
</tr>
</tbody>
</table>
Practice 3-3 Overview

This practice covers adding attributes to entities enhancing the basis for the data warehouse database design.

Practice 3-3
In this practice, you explore how to enhance and refine your star model created in the previous exercise. You will document the attributes of the entities to provide important meta data for the continued success of the project.
Solutions for this exercise are located in Appendix B.
Practice 3-3 (continued)

Enhancing the Star Dimensional Model

Entities need attributes to define them. As mentioned in practice 3-2, an attribute is used for one of five things:

- To identify
- To classify
- To describe
- To quantify
- To express the state of the entity.

Specific notations are common in diagramming. The following is a list of these notations:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Mandatory</td>
<td>Value must always be specified</td>
</tr>
<tr>
<td>°</td>
<td>Optional</td>
<td>Value must not always specified</td>
</tr>
<tr>
<td>#</td>
<td>UID</td>
<td>Attribute is part of, or is a unique identifier</td>
</tr>
</tbody>
</table>
Practice 3-3 (continued)

Goal
Enhance the star model created in practice 3-2 by adding appropriate attributes to the entities.

Scenario
The management team is pleased with the results of the initial model created for Vintners Ltd. However, they would like to see more detail and development of the diagram.

Your Assignment
Using the attribute tables provided in Appendix A with the balance of the historical documentation, enhance the star model to define the attributes for CUSTOMER and SALES.

To assist you in enhancing the model, answer the following question:

1. What attribute should be added to each table to ensure uniqueness in the database?

2. Define the attributes for CUSTOMER and SALES.
Creating the Physical Model

ILT Schedule:  

<table>
<thead>
<tr>
<th>Timing</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>90 minutes</td>
<td>Lecture</td>
</tr>
<tr>
<td>30 minutes</td>
<td>Practice</td>
</tr>
<tr>
<td>120 minutes</td>
<td>Total</td>
</tr>
</tbody>
</table>
Objectives

After completing this lesson, you should be able to do the following:

- Translate the dimensional model into a physical model
- Discuss the architectural requirements for the data warehouse
- Consider the benefits of various hardware architectures
- Describe the database server characteristics required in a warehouse environment
- Review the importance of parallelism for the data warehouse environment

Lesson Overview

In this lesson, you map the dimensional model into the physical elements and examine a number of options for data storage, reviewing several different computer architectures.
Designing the Physical Model

Designing the Physical Model

The physical model resides in the relational database server (RDBMS). You must ensure that each object stored (primarily tables) is held in the appropriate manner and contains all the necessary indexes to ensure optimal performance. There are other considerations that you should consider for performance, such as data partitioning.

Physical data partitioning, segmentation, and data placement are evaluated against business and user requirements and operational constraints. Indexes and key definitions are decided. The database definition language (DDL) is generated and used to build and implement the development, testing, and production of data warehouse database objects.
Database Object Naming Conventions

- Keep the logical and physical names similar and descriptive.
- Capitalize table and attribute names.
- Use underscores instead of spaces to delineate separate words in an object’s name.
- Use a suffix of _PK to indicate primary keys.
- Use a suffix of _ID to indicate production keys.
- Find a good balance between using very specific and very vague names.

Develop Object Name Standards

Dimensional Model to Physical Model Map the dimensional model to the physical elements by performing the following tasks to the dimensional model:
  - Use a naming standard for database objects
  - Add the format, such as datatypes and lengths, to the attributes of each entity

It is very important to have database object naming standards and to follow them. In general, we recommend that logical and physical names be identical and as descriptive as possible. If your company’s IS department has database object naming standards, you should use them; however, existing standards were probably developed for OLTP systems and they might need to be modified for your purposes.

Capitalize table and attributes names. In this course we assume case sensitivity from the start. Keep table name in plural form and attribute names in singular form.

Use underscores rather than spaces to delineate separate words in an object’s name.
  - Use a suffix of _PK to indicate primary keys. For example, use PRODUCT _PK to indicate the primary (surrogate) key for the PRODUCT dimension.
  - Use a suffix of _ID to indicate production keys. For example, the PRODUCT dimension has PRODUCT_ID as an attribute. This is probably the key used in the production systems.
  - Find a good balance between being too specific such as COMPANY_CUSTOMER_ID_LIST, and far too vague, such as C_ID.
Database Object Naming Conventions

- Develop a reasonable list of abbreviations.
- List all the objects' names, and work with the user community to define them.
- Resolve name disputes.
- Document your naming standards in the metadata document.
- Plan for the naming standards to be a living document.

Develop Object Name Standards
- List all the object names and work with the user community to define them.
- Resolve name disputes. The data steward is responsible for seeking business group representation to gather suggestions, make choices, and get their buy-ins.
- Document your naming standards in the metadata document, especially any choices that provoke controversy within the team.
- Plan for the naming standards to be a living document. Follow the standards but be willing to modify them as real needs arise. Develop a reasonable list of abbreviations that would be obvious to a new employee the first day on the job.
Translating the Dimensional Model into a Physical Model

- Apply the naming standards to the tables and attributes of the dimensional model.
- List table columns with primary keys listed first.
- Label primary keys consistently.
- Identify the format and length of columns.
- Label unique keys with a (#).
- Label column optionality with NULL (o) or NOT NULL (*) constraints.
- Label foreign keys with _FK.
- Use synonyms for user tables.

Translating the Dimensional Model into a Physical Model
- Apply the naming and database standards to the tables and attributes of the dimensional model.
  - An example of a naming conventional might be to construct the object using the generally accepted form PrimeWord_Qualifier_ClassWord, where:
    - PrimeWord is the subject like Customer or Product
    - Qualifier is an optional further description like Average or Total
    - ClassWord is a major classification for the element like Start or End
- List columns inside the box, with primary keys listed first.
- Label primary keys with a _PK and (* #).
- Identify the format and length of columns: n for number, c for character, v for variable character, and d for date.
- Label unique keys with a (#).
- Label column optionality with NULL (o) or NOT NULL (*) constraints.
- Label foreign keys with a _FK and (* #) if part of the PK.
- Use synonyms for business users’ tables to differentiate them from programmers’ tables to preclude any problems with restructures against large tables.

Note: This is an overview; not all conventions are mentioned.
Physical Model
Product

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># *Product_PK</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td># *Channel_PK</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td># *Promotion_PK</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>* PRODUCT_ID</td>
<td>v(11)</td>
<td></td>
</tr>
<tr>
<td>* PRODUCT_DESC</td>
<td>v(125)</td>
<td></td>
</tr>
<tr>
<td>* PRODUCT_NAME</td>
<td>v(35)</td>
<td></td>
</tr>
<tr>
<td>* CATEGORY_ID</td>
<td>v(20)</td>
<td></td>
</tr>
<tr>
<td>* CATEGORY_DESC</td>
<td>v(25)</td>
<td></td>
</tr>
<tr>
<td>* SUPPLIER_ID</td>
<td>v(20)</td>
<td></td>
</tr>
<tr>
<td>* PRODUCT_STATUS</td>
<td>v(10)</td>
<td></td>
</tr>
<tr>
<td>* LIST_PRICE</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>* CATALOG_ID</td>
<td>v(20)</td>
<td></td>
</tr>
<tr>
<td>* PRODCUT_TYPE</td>
<td>v(20)</td>
<td></td>
</tr>
<tr>
<td>* PRODUCT_CODE</td>
<td>v(10)</td>
<td></td>
</tr>
<tr>
<td>* PROMOTION_CODE</td>
<td>v(10)</td>
<td></td>
</tr>
<tr>
<td>* WHSE_LOCATION</td>
<td>v(10)</td>
<td></td>
</tr>
<tr>
<td>* VALID_FROM_DATE</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>* VALID_TO_DATE</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

Developing the Physical Data Model

The starting point for the physical data model is the dimensional model. Although similar, the major difference between these two models is that the physical model is a thorough and detailed specification of physical database characteristics such as the datatypes, length, database sizing, indexing strategy, and partitioning strategy.

Note: During the process of translating the dimensional model into a physical model, you must to determine which columns will have datatypes that differ from their representation in the source system (if the source systems are available). For example, the zip code may be stored as an integer in the source system and as CHAR within the warehouse.
Defining the Hardware

Transforming the base dimensional data model into the physical model includes some of the following:

- Defining naming and database standards
- Performing an initial sizing
- Designing tablespaces
- Defining an initial indexing strategy
- Using partitioning to split table and index data into smaller, more manageable chunks
- Determining where to place database objects on disk (RAID, striping, disk mapping)
- Using parallel processing

Defining the Hardware

The data warehouse tenets described in lesson two, namely that the architecture must be scalable, manageable, available, extensible, flexible, and integrated, are perceived to be the primary tenets in a data warehouse environment. From an Oracle perspective, the tenets of the Data Warehouse Methodology (DWM) can be extended to include tunable, reliable, robust, supportable, and recoverable.

Compromises may affect the task of balancing user needs and business requirements if budgetary constraints restrain your choices or if technical difficulties are too challenging. The architecture requirements definition must be considered at an early stage, in parallel with the user requirements. Only at this time can successful choices be made.

A good physical model is often the difference between a success or failure. The design of the physical model builds on the logical model, adding indexes, referential integrity, and physical storage characteristics.
Architectural Requirements

Strategy for Architecture Definition

Although mentioned in lesson 1, it is important to reiterate the importance of these critical data warehouse architectural requirements. You must have a definitive strategy that uses identified and proven technology. Using DWM as a foundation for this discussion, consider some of the tasks you must perform in the early stages when planning the hardware architecture and surrounding environment.

- Obtain existing plans and outlines of the current technical architecture for the environments that will supply the warehouse.
- Obtain existing capacity plans for the current environments.
- Document existing data warehouse interfaces, and document enterprise data warehouse interface requirements.
- Prepare enterprise data warehouse capacity plan.
- Prepare enterprise data warehouse technical architecture.
- Document enterprise data warehouse system operational requirements.
- Develop recovery and fallback strategy.
- Develop security and control strategy.
- Create enterprise data warehouse architecture.
- Create technical risk assessment.
Architecture Characteristics

Consider the hardware architectures first. This is an area of the plan where a number of people including the data warehousing IT team members, must be involved. This includes the current database administrators of the operational systems, who have the experience and expertise of current systems and performance and who can also provide useful input regarding the existing architectures and interfaces. You must ensure that networking staff are involved as well. It is a critical issue for processes such as ETT and user access.

The choice of hardware architecture is critical to the success of the data warehouse and its infrastructure. Warehouses require hardware architectures that are robust, available, reliable, flexible, extensible, scalable, & supportable, recoverable and parallel.

In addition, the architecture should:

- Have a very large memory (VLM) capability
- Be able to use 64-bit addressing
- Be connective and conform to open system standards

Important: Do not confuse the term database server with a file server on a local area network or any other server. For our purposes, the term database server describes the Relational Database Management System (RDBMS) or Database Management System (DBMS).
Oracle9i Database
Architectural Advantages

New and improved technologies:
• Real Application Clusters and Cache Fusion
• Self-managing in critical areas
• Flashback Query
• Data Guard and Recovery Manager

Oracle9i Database Architecture Advantages
• Applications run concurrently across nodes without any modifications, yielding a highly scalable structure for e-business. Real Application Clusters employs Cache Fusion to “virtually” combine multiple caches across nodes providing high availability of new hardware. Additionally, the database workload dynamically shifts and self tunes to accommodate the database workload and to satisfy query requests from local or other caches.
• Oracle9i Database is capable of managing its own undo (rollback) segments by allocating the undo space in a single undo tablespace. You can grow or shrink the SGA dynamically and re-size a buffer cache or shared pool.
• Flashback Query allows for human error correction, while Data Guard and Recovery Manager promote enhanced disaster recovery and automation.
Hardware Requirements

- SMP (Symmetric multiprocessing)
- Cluster and MPP (massively parallel processing)
- NUMA (nonuniform memory access)
- Hybrids using SMP and MPP

Hardware Requirements

Hardware architectures support a number of different configurations that are useful for data warehousing and are more cost effective than hardware architectures previously available.

- Symmetric multiprocessing (SMP)—Symmetric multiprocessing architectures are the oldest of the technologies and have a proven track record.
- Cluster—Cluster and massively parallel processing architectures are comparatively new but are more scalable and provide a lot of power.
- Massively parallel processing (MPP) and Nonuniform Memory Access (NUMA)—NUMA is an even more recent innovation that gives you the scalability of an MPP environment and the manageability of an SMP environment.
- Some architectures are a hybrid, using both SMP and MPP capabilities.
Evaluation Criteria

Determine the platform for your needs:

- SMP
- Clusters
- NUMA
- MPP

Scalability:

- Low
- High

Maturity:

- Low
- High

By specifying the hardware requirements early on in the development of the warehouse, you have enough lead time to acquire and test the chosen components. Determining the platform depends upon a number of factors, and the different architectures have advantages and disadvantages that you must evaluate before making a final decision.

A symmetric multiprocessing architecture may be sufficient if you have a small database, can afford a longer response time, and have problems that are not complex. Problem complexity is determined by the number of users, the type of calculations, and the types of queries that the system must handle.

The larger your database, the more complex your problems, and the shorter the required response time, the closer you are to specifying a massively parallel processing system.
Parallel Processing

- Parallel daily operations
- Shared resources
  - Memory
  - Disk
  - Nothing
- Loosely or tightly coupled

**Parallel Processing**

Hardware architectures that contain parallel processors are often categorized according to the resources they share.

- Memory—SMP machines are often described as tightly coupled.
- Disk—Clustered architectures are often described as loosely coupled.
- Nothing—MPP machines are described as loosely or tightly coupled, according to the way communication is accomplished among nodes.
- NUMA is an SMP architecture with loosely coupled memory using uniform and nonuniform memory access.
Making the Right Choice

- Requirements differ from operational systems
- Benchmark
  - Available from vendors
  - Develop your own
  - Use realistic queries
- Scalability important

Making the Right Choice
How do you know which architecture to choose? Operational environments do not map directly to the way the warehouse operates, with its unpredictable workloads and scalability requirements. The only realistic way to determine the interaction between your data warehouse database and the hardware configuration is to perform full-scale testing. Of course you may not be able to achieve this.

When benchmarking, use real user queries against volumes of data that mimic the volumes anticipated in the warehouse. If you are unhappy with vendor benchmarks, consider developing your own. This is going to add to the cost of development. However, costs are high for a warehouse implementation and you may find the amount spent on your own benchmark worthwhile in the long term.
Symmetric Multiprocessing (SMP)

- Communication by shared memory
- Disk controllers accessible to all CPUs
- Proven technology

Symmetric Multiprocessing

A symmetric multiprocessing (SMP) machine comprises a set of CPUs that share memory. It has a shared everything architecture composed of the following:

- Each CPU has full access to the shared memory through a common bus.
- Communication between the CPUs uses the shared memory.
- Disk controllers are accessible to all CPUs.

This is proven technology, particularly in the data warehousing environment.
SMP

Benefits:
• High concurrency
• Workload balancing
• Moderate scalability
• Easy administration

Limitations:
• Memory (cluster for improvements)
• Bandwidth

Note: SMP machines are often nodes in a cluster. Multiple SMP nodes can be used with certain vendors’ architectures—DEC, Pyramid, Sequent, Sun, SparcServer—where disk is shared among multiple nodes. Some warehouse sites are exploring the evolving concept of loaning excess memory or processing capacity among applications or hardware. Some SMP vendors allow you to scale to MPP without losing your SMP box. You simply add interconnect software and associated technology.
Nonuniform Memory Access (NUMA)

Loosely coupled shared memory:

Nonuniform memory access

Shared memory

Disk

CPU node 1  Shared bus  Shared memory  Disk

Shared memory  CPU node 2  Shared bus  finally back to CPU node 1

Nonuniform Memory Access

Shared memory systems are systems with loosely coupled memory. The shared memory can be accessed by using uniform memory access from CPUs or by Nonuniform Memory Access (NUMA). The Oracle Parallel Server can work with either form of memory access, but NUMA is a more costly form of access and synchronization than uniform memory access. While any CPU can access the memory, it is more costly for remote nodes.

In the example, if you follow a request through the node (red dashed and dotted line), you see that disks are not shared amongst the CPU nodes. Therefore the flow is as follows:
NUMA

• Benefits:
  – Fully scalable, incremental additions to disk, CPU, and bandwidth
  – Performs better than MPP
  – Suited for Oracle server

• Limitations:
  – The technology is new and less proven
  – You need new tools for easy system management
  – NUMA is more expensive than SMP

NUMA

Benefits:
• A fully scalable architecture that can overcome some of the scalability problems of SMP
• A very scalable parallel architecture, and therefore it is possible to add disk, CPU, and bandwidth incrementally to any level
• A system that performs better than an MPP system where there are ad hoc or mixed workloads
• Increased speed
• Hybrid of SMP with MPP

Oracle runs as an application on the operating systems. Parallel server and parallel instances are accessing different databases. The bus takes care of the requests without each database knowing that multiple processes are requesting data.
Clusters

Shared disk, loosely coupled systems have the following characteristics:

- Each node consists of one or more CPUs and associated dedicated memory.
- Memory is not shared between nodes.
- Communication occurs over a high-speed bus.
- Each node has access to all of the disks and other resources.
- An SMP machine can be a node, if the hardware supports it.

Instructor Note

Real Application Clusters, a feature within the Oracle9i database, allows for a “plug & play” deployment of new applications and hardware to an existing pool by using Cache Fusion or the creation of a single virtual cache that dynamically redistributes the workload and transparently reconfigures and balances the load across all nodes.
Clusters

- Shared disk, loosely coupled
- Dedicated memory
- High-speed bus
- Shared resources
- SMP node

Benefits:
• High availability; all data is accessible even if one node dies
• Single database concept; the concept of one database, which is an advantage over shared nothing systems such as MPP
• Incremental growth

Limitations:
• Scalability; bandwidth of the high speed bus limits the scalability of the system.
• Internode synchronization needed; Internode synchronization is required. Each node has a data cache; cache consistency must be maintained for the locking mechanisms to work effectively.
• Operating system overhead

Instructor Note
Each node runs one instance of Oracle to one single database. Clustered and MPP computers include Digital, Tandem and Exotics (Teradata), and NCR, Inc.
Massively Parallel Processing (MPP)

Massively Parallel Processing
The massively parallel processing (MPP) architecture is concerned with disk access, rather than memory access, and works well with operating systems that provide transparent disk access. You can scale the configuration up by adding more CPUs.
If a table or database is located on a disk, access depends entirely on the CPU that owns it. If the CPU fails, the data cannot be accessed, regardless of how many other CPUs are running, unless logical pointers are established to alternative CPUs.
Typically, MPP architectures have the following characteristics:
• Are very fast compared with SMP and cluster architectures
• Support a few to thousands of nodes
• Provide fast access between nodes
• Have associated nonshared memory associated with each node
• Have a low cost per node
MPP technology is comparatively new and not proven to the same extent as SMP and cluster technology.
MPP \( n \)-CUBE Arrangements

- A shared nothing architecture
- Many nodes
- Fast access
- Exclusive memory on a node
- Low cost per node
- Scalable
- \( n \)-CUBE configuration

Nodes may be organized on a grid arrangement if using \( n \)-CUBE. Multiprocessor designs provide a scalable architecture that allows to increase performance easily as your needs grow. The essential principle to a multiprocessor system is the interconnect—the mechanism that allows the processors to communicate and cooperate. In an \( n \)-CUBE system, processors are connected in a multidimensional cube called a hypercube, providing the fastest and densest communications network available. The hypercube network is organized so that connections among processors form cubes. As more processors are added, the cube grows to larger dimensions. The \( n \)-CUBE system is scalable to hundreds of processors.
MPP Benefits

- Unlimited incremental growth
- Very scalable
- Fast access
- Low cost per node
- Good for DSS

MPP Benefits

- Practically unlimited, and incremental growth
- Very scalable (given careful data placement)
- Fast access between nodes
- Low cost per node (each node is an inexpensive processor). Each node has its own devices, but, in case of failure, other nodes can access the devices of the failed node (on most systems); failure may be local to the node.
- Good for DSS and read-only databases

Instructor Note

A partial list of clustered and MPP computers includes HP MC/Clusters, IBM RS/6000 HACMP or SP2, SunSparcCluster PDB, AT&T GIS 3600, Sequent ptx/Clusters, Pyramid Clusters, nCUBE, and Reliant 100 or 150.
MPP Limitations

- Rigid partitioning
- Cache consistency
- Restricted disk access
- High memory cost per node
- High management burden
- Careful data placement

MPP Limitations
- Many database servers (not necessary with Oracle) require rigid data partitioning for parallelism and scalability.
- Cache consistency must be maintained.
- Disk access is restricted.
- The memory cost per node is high.
- The management burden is high.
- Careful data placement is required for scalability.
Windows NT

• Architecture based on the client-server model

• Benefits:
  – Includes built-in Web services
  – Scalability (up to 4 processors)
  – Ease of management and control

• Limitations:
  – Not as secure
  – Cannot execute programs remotely (RCP)
  – Lacks linear scalability beyond four processors
  – Addressing space for applications is limited to two gigabytes

Windows NT
The architecture for Windows NT is based on the client-server model. The approach divides the operating system into an executive running in kernel mode and several server processes, each running in user mode. Each server process implements a unique operating system environment.

Benefits:
• Windows NT server operating system includes built-in Web services that provide a complete, integrated intranet solution.
• Windows NT offers scalability improvements of up to 33 percent, yielding more linear scalability on machines with eight or more processors.
• Ease of management and control with user profiles and system policies enable system administrators to easily manage user desktops, including the ability to control access to the network and to desktop resources as well as support for users accessing multiple workstations.

Limitations:
• Windows NT is not as secure as other operating systems such as UNIX.
• Although Windows NT can support SMP with up to 32 processors, Windows NT has been criticized for its lack of linear scalability beyond four processors.
• Addressing space limits Windows NT applications to two gigabytes. This is insufficient for large data warehouses.
Architectural Tiers

Tiered structures:
- Modular
- Logical separation

Distributed structures:
- Two-tier
- Three-tier
- Four-tier (and more)

Architectural Tiers

Architectures can be the simple two-tier type, the more complex and most common three-tier type, or if Web applications are involved, up to a four-tier type. This enables a useful division of labor for specific tasks and processes, and can assist and complement the network setup.

- Two-Tier Architecture—A simple two-tier architecture involves a mainframe CPU, such as IBM, with source data, which is copied and extracted periodically to a smaller server, such as Windows NT. A query and analysis tool is then provided for the NT environment. This structure does not fit well into the kind of enterprisewide environments discussed so far.

- Three-Tier Architecture—A three-tier architecture uses a separate middleware layer for data access and translation. Tier 1 hosts the production applications on a mainframe or midrange system and is devoted to real-time production level data processing. Tier 2 comprises a departmental server resident with the warehouse users, for example, a UNIX workstation or NT server, which is optimized for query processing, analysis, and reporting. Tier 3 comprises the desktop and handles reporting, analysis, and graphical data presentation. PCs are connected on a LAN. The three-tier architecture is more effective than two-tier architecture because the first tier is devoted to operational processing, the second to department-level query processing and analysis, and the third to desktop data presentation.

- Four-Tier and Greater Architecture—Adds a Web-based server.
Middleware

Middleware is a term that is used to describe technologies that allow you to integrate multiple server technologies together in a seamless manner. Middleware tools are common in today’s computing environment. Oracle gateway technology is one example of middleware that is available off the shelf. In a multitier data warehousing environment with Internet access, middleware is becoming increasingly redefined and refined.
Database Server Requirements

• Robust
• Available
• Reliable
• Extensible
• Scalable
• Supportable
• Recoverable
• Parallel
Parallelism

- Database
- Query
- Load
- Index
- Sort
- Backup
- Recovery

The driving force behind the warehouse implementation is the needs of the end users who require access to the information. The database environment must handle all operational tasks and processes quickly and efficiently. Of course, parallel capabilities minimize the time that is required to perform all the major functions of the warehouse and maximize availability. As you have seen parallelism at all levels is becoming mandatory for warehouses.
Further Considerations

- Optimization strategy
- Partitioning strategy
- Summarization strategy
- Indexing techniques
- Hardware and software scalability
- Availability
- Administration

Further Considerations
Parallelism is not the only consideration; you must also consider the following:
- The optimization strategy, particularly star query techniques used with star and snowflake structures (Today’s servers enable you to optimize data access in many different ways.)
- The partitioning strategy
- Summarization strategies, to ensure that the overhead of creating summaries does not affect the load
- Indexing techniques, in particular, bitmap indexes
- Hardware and software scalability
- Availability of the warehouse
- The system administration, which must easily manage the entire infrastructure

Instructor Note
Oracle9i database provides not only high availability using Cache Fusion, but also high availability by employing Fast Restart and Recovery Manager (bringing a failed online system up quickly), Flashback Query and Log Miner (for reversing human error), and Online Redefinition (for performing routine maintenance without taking the database offline).
Server Environments

<table>
<thead>
<tr>
<th>Operational servers</th>
<th>Warehouse servers</th>
<th>Data mart servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Open DBMS</td>
<td>• Open DBMS</td>
<td>• Open DBMS</td>
</tr>
<tr>
<td>• Network, relational, hierarchical</td>
<td>• Relational</td>
<td>• Relational and multidimensional</td>
</tr>
<tr>
<td>• Mainframe proprietary DBMS</td>
<td>• General purpose and warehouse-specific DBMS</td>
<td>• General purpose and warehouse-specific DBMS</td>
</tr>
<tr>
<td>• Oracle, IMS, DB2, VSAM, Rdb, Non-Stop, SQL, RMS</td>
<td>• Oracle, Informix, Sybase, IBM DB2, NCR/AT&amp;T, Teradata, Red Brick</td>
<td>• Oracle, Oracle Express, Arbor Essbase, MS SQL Server, NT</td>
</tr>
</tbody>
</table>

Server Environments

Many different database servers and hardware architectures can be used for a warehouse solution. It is generally assumed that data warehouse database technology means relational technology.

• Operational Servers—Open, mainframe proprietary database servers (whether network database server, hierarchical database server, or relational database server), such as Oracle, IMS, DB2, DB2/PE, VSAM, Rdb, Non-Stop, SQL, or RMS.

• Warehouse Servers—Open (usually relational) database servers that may be warehouse specific or general purpose, such as Oracle, Informix, Adabas D, OpenIngres, or Red Brick.

• Data Mart Servers—Relational, multidimensional (OLAP) databases, or both; they may be warehouse specific or general purpose, such as Oracle, Oracle Express, Arbor Essbase, MS SQL Server, and NT-based environments.

B-tree indexes are useful on dimension tables and bitmap indexes on fact tables.
Parallel Processing

A large task broken into smaller tasks:
- Concurrent execution
- One or more processors

Parallel Processing
A parallel processor takes a task (usually a large task) and divides it into smaller tasks that can be executed concurrently on one or more nodes (separate processors). As a result, a large task requested by a single user completes more quickly. Before examining the individual parallel features, consider the parallel database.
Parallel Database

A parallel database takes advantage of architectures that share access to data, software, and peripheral devices by running multiple instances that share a single physical database. This type of processing has two key features:

- Increased speed—The server can perform the same task in less time.
- Improved scalability—The ability to perform a task many times larger, on a system many times larger, without any performance degradation.

These key features give you the following benefits:

- Higher performance
- Greater availability
- Greater flexibility
- Greater accessibility to online users

All of these features directly benefit the warehouse and are supported by the Oracle7, Oracle8, Oracle8i, and Oracle9i Server.
Parallel Query

SQL code split among server processes

Parallel Query
Most database servers today support parallel query. Specifically, the Oracle Server parallel query option divides the work of processing a single SQL statement among multiple query server processes. In some applications, particularly decision support systems, an individual query can use vast amounts of CPU resource and disk I/O. The server parallelizes individual queries into units of work that can be processed simultaneously. Oracle’s flexible architecture allows for a query to execute with any specified degree of parallelism. Use the parameter, PARALLEL AUTOMATIC_TUNING to TRUE. Oracle9i database has enhanced the parallelization strategies internally to take advantage of MPP configurations using Real Application Clusters.
Parallel Load

Bypass SQL processing to speed throughput

Parallel Load

Parallelism can dramatically increase the speed at which data loads. Database servers can bypass standard SQL processing (that is, data manipulation language commands, such as `INSERT`), and the data is loaded directly into the database tables.

Scalable Operations

Data Partitioning enables data management operations to be performed in parallel at the partition level and delivers the full benefits of parallel processing to data warehouses and data marts. Virtually every operation—data loading, building indexes, enforcing constraints, gathering optimizer statistics, and backup and recovery—can be done at the granularity of a partition. Further, data reorganization tasks such as moving partitions, splitting partitions or converting Oracle7 Partition Views into table partitions can all be performed using scalable parallel data flow operations.

Beginning with Oracle8, support was introduced for the execution of bulk insert, update and delete operations in parallel. These parallel data manipulation operators deliver high performance, scalability and efficient utilization of all hardware resources and enable the completion of data management tasks within ever-shrinking batch windows. Parallel insert, update and delete capabilities are very useful for bulk operations such as the creation of summary tables, purging historical data, data reorganization, and batch updates in support of “what-if” analysis.
Parallel Processing

- Index: reduces the time to create
- Sort: allocates memory in cache efficiently

Parallel Processing

Parallel Index Creating an index in parallel decreases the time required to create and reconfigure a warehouse. Many indexes exist in the warehouse database. Nearly every attribute on dimension tables and composite key value on the fact table is indexed. Indexes take up a lot of space in the warehouse, and you must consider the direct access storage device (DASD) needed for indexes, as well as fact and dimension tables.

Parallel Sort Sorting is an intensive task that requires a substantial amount of memory. If you are working in a parallel environment, sort areas are allocated more efficiently to reduce serialization and cross-instance pinging. Sort space is cached in memory (in the Oracle server this is in the System Global Area).
Parallel Processing

- Backup: runs simultaneously from any node (online and offline)
- Recovery: runs simultaneously from redo logs
- Summaries: uses the `CREATE TABLE AS SELECT` statement

Parallel Processing

**Parallel Backup** With parallel operations, backups can be performed simultaneously from any node of a parallel server.

- Online backups enable the database to be backed up while active, allowing users continuous access.
- Offline backups enable the database to be backed up while shutdown, preventing user access.

**Parallel Recovery** The goal of parallel recovery is to use I/O parallelism to reduce the elapsed time required to perform crash recovery, instance recovery, or media failure recovery. The server uses one process to read files sequentially and dispatch redo information to several recovery processes to apply the changes from the log files to the data files.

**Parallel Table Creation** You can create tables in a parallel manner using the `CREATE TABLE AS SELECT (CTAS)` statement.
Summary

In this lesson you should have learned how to do the following:

• Translate the dimensional model into a physical model
• Discuss the architectural requirements for the data warehouse
• Consider the benefits of various hardware architectures
• Describe the database server characteristics required in a warehouse environment
• Review the importance of parallelism for the data warehouse environment
Practice 4-1 Overview

This practice covers understanding the physical model, including hardware architecture and parallel query.

Practice 4-1
Solutions for this exercise are located in Appendix B.
Practice 4-1
Answer the following questions:

1. Where does the physical model reside?
   ______________________________________________
   ______________________________________________
   ______________________________________________

2. __________ is the oldest hardware architecture useful for data warehousing.

3. __________ and __________ provide a greater level of scalability over SMP for data warehousing.

4. __________ although a relatively new entrant into the data warehousing arena, provides both scalability and manageability.

5. 3-tier (or more) architecture is preferable for data warehousing because
   ______________________________________________
   ______________________________________________
   ______________________________________________
   ______________________________________________
   ______________________________________________

6. A __________ ____________ divides a large task into smaller tasks that can execute concurrently in one or more nodes.

7. __________ ____________ splits SQL code among several processors to minimize the impact of query analysis on CPU resource and disk I/O.

8. Why is a parallel database of benefit to the data warehouse?
   ______________________________________________
   ______________________________________________
   ______________________________________________
   ______________________________________________
   ______________________________________________
Practice 4-2 Overview

This practice covers enhancing a Star Dimensional Model.

Practice 4-2
Solutions for this exercise are located in Appendix B.
**Practice 4-2 (continued)**

**Additional Diagram Notations**

As explained in practice 3-3, entities need attributes to define them. The following is a list of these common notations within Oracle tools:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Mandatory</td>
<td>Value must always be specified</td>
</tr>
<tr>
<td>°</td>
<td>Optional</td>
<td>Value must not always specified</td>
</tr>
<tr>
<td>#</td>
<td>UID</td>
<td>Attribute is part of, or is a unique identifier</td>
</tr>
</tbody>
</table>
Practice 4-2 (continued)

Goal
Create a physical design (star schema) for Vintners Ltd. sales process.

Scenario
Now that all attributes and most table definitions have been applied to the star model, management would like to see a rudimentary data model. In order to do this, you have to transform the conceptual model. The star model is composed of tables, columns, and keys.

Your Assignment
To assist you in developing the star schema further, answer the following questions:

1. What column in the fact table of a star schema represents a dimension?
   __________________________________________

2. Because the sales table is the heart of the star schema, it must be associated with the other tables. To accomplish this, you must add three foreign keys to the sales table. Which three tables should participate?
   1. __________________________________________
   2. __________________________________________
   3. __________________________________________

3. Create a star schema for Vintners Ltd. sales process.
Practice 4-3 Overview

This practice covers completing a Star Dimensional Model.

Practice 4-3
Solutions for this exercise are located in Appendix B.
Practice 4-3 (continued)

Goal
Complete the star schema for Vintners Ltd. customer sales process.

Scenario
You have worked hard to develop the initial star schema for the customer sales process to present to management, but you realize that it is incomplete. A data warehouse must always contain a time dimension.

Your research indicates the following:
• Vintners Ltd. will have a normal ISO calendar on which to base the time dimension. Currently no fiscal hierarchies are required within the warehouse. The time dimension has the following hierarchy from left to right (highest to lowest level):
  Gregorian—YEAR, QUARTER, MONTH, DAY
  Fiscal—FYEAR, FQUARTER, FMONTH, FWEEK, FDAY
• You have drawn up a chart for use in developing the dimension. This chart is located on the next page.

Your Assignment
Develop and add a TIME dimension to the star schema based on your research.
<table>
<thead>
<tr>
<th>Level</th>
<th>Attribute</th>
<th>Content of Attribute</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day</td>
<td>Time ID</td>
<td>Surrogate key on lowest level for the dimension</td>
<td>Number (10)</td>
</tr>
<tr>
<td></td>
<td>SQL Date</td>
<td>Date on day level (mm/dd/yyyy)</td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Julian Day</td>
<td>Julian day number</td>
<td>Number (3)</td>
</tr>
<tr>
<td></td>
<td>Day of Year Nbr</td>
<td>Number for the day within the year</td>
<td>Number (3)</td>
</tr>
<tr>
<td></td>
<td>Day of Month Nbr</td>
<td>Number for the day within the month</td>
<td>Number (2)</td>
</tr>
<tr>
<td></td>
<td>Day of Week Nbr</td>
<td>Number for the day within the week</td>
<td>Number (1)</td>
</tr>
<tr>
<td></td>
<td>Day of Week Name</td>
<td>Description for the day of the week. Vintners Ltd. week always begins on Sunday and Sunday is always numbered 1</td>
<td>Char (9)</td>
</tr>
<tr>
<td>Month</td>
<td>Month Nbr</td>
<td>Number of the month within the year</td>
<td>Number (2)</td>
</tr>
<tr>
<td></td>
<td>Year Month Nbr</td>
<td>Number of the month combined with the year</td>
<td>Number (6)</td>
</tr>
<tr>
<td></td>
<td>Month Name</td>
<td>ISO calendar month names should be used for this field</td>
<td>Char (8)</td>
</tr>
<tr>
<td>Quarter</td>
<td>Year Quarter Nbr</td>
<td>Year combined with quarter number</td>
<td>Number (5)</td>
</tr>
<tr>
<td></td>
<td>Quarter Nbr</td>
<td>Number of the quarter</td>
<td>Number (1)</td>
</tr>
<tr>
<td>Year</td>
<td>Year Nbr</td>
<td>Year number</td>
<td>Number (4)</td>
</tr>
</tbody>
</table>
Storage Considerations for the Physical Model

ILT Schedule:

<table>
<thead>
<tr>
<th>Timing</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>90 minutes</td>
<td>Lecture</td>
</tr>
<tr>
<td>15 minutes</td>
<td>Practice</td>
</tr>
<tr>
<td>105 minutes</td>
<td>Total</td>
</tr>
</tbody>
</table>
Objectives

After completing this lesson, you should be able to do the following:

- Explain data warehouse sizing techniques and test load sampling
- Describe data warehouse indexing types and strategies
- Discuss features and storage strategies for tablespaces
- Discuss partitioning methods for tables and indexes

Lesson Overview

In this lesson, you learn about ways to size your data warehouse, and enhance query performance through partitioning and indexing. One of the main challenges within data warehousing is to recognize that fact and detail tables will grow incredibly large and to manage that growth successfully. Query performance continues to present challenges as these fact tables grow. Partitioning, indexing, and offloading data that is no longer required are essential to sustaining a healthy data warehouse. This lesson focuses on different methods that assist in balancing the warehouse, and is broken down into three primary components: sizing, partitioning, and indexing.
Database Sizing

- Sizing influences capacity planning and systems environment management.
- Sizing is required for:
  - The database
  - Other storage areas
  - Indexes
- Sizing is not a science.
- Techniques vary.
- DWM suggests when to perform sizing.

Sizing the Database and Other Storage Requirements
A major factor in capacity and space planning is the physical size of the data warehouse database. The considerations for sizing include the amount of physical disk space that is required for the data warehouse database; for example, for the tables, views, and indexes. Determine the amount of physical disk space required for:

- The architecture of the environment
- Backup and recovery tasks
- Mirroring techniques
- Temporary space and loading techniques

Sizing the database is not an exact science. Techniques vary from implementation to implementation with many possible approaches; you should identify one that meets the requirements of your implementation.
Estimating the Database Size

1. Estimate the size of each row in the fact table.
2. Determine the grain of each dimension and estimate the number of entries in the finest level.
3. Multiply the number of rows of all dimensions and multiply the result by the fact table row size.
4. Determine whether the fact table is sparse or dense and estimate the reduction or increase in size.

Estimating the Database Size
Gary Dodge recommends that although the Oracle Server Administrator’s Guide provides important information on sizing a database, additional steps are required to preclude fragmentation which could ultimately result in performance degradation. He further states that the methodology that you use involves the following exigencies:

• Examine the size of the Oracle data block because this affects database performance and storage efficiency. The data block must account for header space, PCTFREE, and growth (future inserts).
• Examine the space required for an average row by determining the average column length for each column within the row. Because most Oracle data types are stored in variable length formats, review the test or quality assurance database to use as a benchmark estimate. Use the \texttt{AVG(VSIZE(column\_name))} to assist with this evaluation. Repeat this for each table within the star design.
• Examine the space required for indexes and summary table by using the same method annotated above except estimate the number of rows to be stored in each summary by using the \texttt{GROUP BY} clause.
• Plan approximately four times this space for rollback segments, temporary tablespace, and system tablespace. A good practice is to provide three to four times the space of actual detail data for indexing, summarization, and other system requirements.

Gary Dodge, \textit{Essential Oracle8i Data Warehousing}, Copyright 2000
Validating Database Size Assumptions

After you estimate the size of the database, you can validate your assumptions by doing the following:

• Extract sample files
• Load data into the database
• Compute exact expected row lengths
• Add overhead for indexing, rollback and temporary tablespaces, aggregates, views, and a file system staging area for flat files

Validating Database Size Assumptions

Every time Oracle reads from or writes to the database, it does so using database blocks. These blocks are moved in and out of the SGA. The database block size must be a multiple of the underlying operating system block size. Although the default block size for Oracle is set to 2KB, a minimum of 8KB is recommended with 16KB preferred. In a large data warehouse, the more data that can be read in at one time, the more performance is enhanced.
# Example: Estimating the Database Size

<table>
<thead>
<tr>
<th>Description</th>
<th>Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate the size of one row of the fact table</td>
<td>52 bytes (assumed for this example)</td>
</tr>
<tr>
<td>Estimate the entries in the lowest level within each dimension</td>
<td>Channel: 3 channels</td>
</tr>
<tr>
<td></td>
<td>Customer: 63 ship_to_locations</td>
</tr>
<tr>
<td></td>
<td>Product: 36 items</td>
</tr>
<tr>
<td></td>
<td>History: 48 months</td>
</tr>
<tr>
<td>Multiply the # of entries for each dimension and multiply the result by the fact table row size</td>
<td>(3 x 63 x 36 x 48) x 52 = 16,982,784 bytes</td>
</tr>
<tr>
<td>Sparsity is low, adjust by 10%</td>
<td>16,982,784 * .10 = 1,698,278</td>
</tr>
<tr>
<td></td>
<td>16,982,784 - 1,698,278 = 15,284,506 bytes</td>
</tr>
<tr>
<td>Estimated database size</td>
<td>15.3 MB</td>
</tr>
</tbody>
</table>
Applying the Test Load Sampling

- Analyze statistically significant data samples
- Use test loads for different periods
- Reflect day-to-day operations
- Include seasonal data and worst-case scenarios:
  - Calculate the number of transactions
  - Use the average sales price approach
- Consider indexes and summaries

Test Load Sampling

A good approach to sizing is based on the analysis of a statistically significant sample of the data. Test loads can be performed on data from a day, week, month, or any other period of time. You must ensure that the sample periods reflect the true day-to-day operations of your company, and that the results include any seasonal issues or other factors, such as worst-case scenarios, that may prejudice the results. After you have determined the number of transactions based on the sample, you calculate the size.

You must also consider the following factors that can have an impact:
- Indexing, because the amount of indexing can significantly impact the size of the database
- Summary tables that can be as large as the primary fact table, depending on the number of dimensions and the number of levels of the hierarchies associated with those dimensions
Test Load Sampling Using the Server

- Load a sample of data.
- Query to determine the number of rows per block.
- Estimate based on the number of rows.

Using the Server

Oracle datatypes are variable in length, which creates a problem when estimating averages accurately. The only accurate source of average column lengths for variable-length data is existing data. Therefore, the best approach is to actually take a sample and allow the Oracle server to calculate the average values, rather than guessing. The larger the sample size, the more accurate your estimate will be.

1. Load several thousand rows into the table.
2. Execute this query to determine the average number of rows per database block:
   For Oracle8
   ```sql
   select AVG(COUNT(DBMS_ROWID.ROWID_ROW_NUMBER(ROWID)))
   from <TABLE>
   where ROWNUM <= <TOTAL-#-ROWS-MINUS-A-FEW>
   group by DBMS_ROWID.ROWID_OBJECT(ROWID),
   DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID),
   DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID);
   ```
3. Take the result from the query. Multiply it by the expected number of rows to obtain the number of database blocks in the table.
Indexes

Indexes for Querying Data Indexes are used by the server to access data as quickly as possible from any table or combination of tables. Indexes have the following characteristics:
- Index column values are stored presorted.
- Indexes are stored in a separate area of the database, which means that they can be created and dropped at any time without any effect on the underlying table.

Indexes for Data Manipulation A unique index on the primary key columns enforces uniqueness when you insert new rows into a table. Additionally, as data is deleted, added, or updated, the relevant indexes are maintained automatically. Indexes exist independently of the data in the table.

Note: When you define a primary or unique key constraint on the `CREATETABLE` statement, Oracle servers create the unique index. Foreign key constraints do not create indexes automatically.

Indexes and the Warehouse Indexes are critical to speed a query that is processed in the warehouse. Determining the correct index strategy for your warehouse is imperative. A warehouse contains many more indexes than an operational system. Analysis determines the data model, which satisfies the query requirements. Indexes are then created, often for the many different combinations of queries that may be performed. The indexes that are particularly useful in data warehousing include B*tree indexes, B*tree cluster indexes, and bitmap indexes.
Indexing Types

To develop a useful index plan, you must to understand how your relational database management system’s query optimizer and indexes work, and how warehouse requirements differ from OLTP requirements. For example, the Oracle server optimizer uses either a rule-based optimization (RBO) approach, which consistently uses indexes, or a cost-based optimization (CBO) approach, which is better at determining whether index use is beneficial. CBO is commonly used in data warehouses. The following types of indexes are available with the Oracle server:

- B*tree indexes
- Bitmap indexes
- Partitioned indexes (Oracle8 and later)
- Index-organized tables (Oracle8 and later)
- Bitmap join indexes
Indexing Types

• **B*tree index**: small number of distinct values in a particular column
• **Bitmap index**: used for star query transformations
• **Partitioned index**: partition B*tree or Bitmap indexes
• **Index-organized tables**: data is held within the index
• **Bitmap join index**: used for star query joins and materialized views

Indexing Types

• **B*tree indexes**—Used when there are a small number of distinct values in a particular column.
• **Bitmap indexes**—The warehouse environment uses CBO, which needs bitmap indexes for star query transformation. You can create an index on each foreign key in the fact table to access fact data by using this algorithm.
• **Partitioned indexes** (Oracle8 and later)—You may choose to partition B*tree or bitmap indexes in sync with your table-partitioning strategy. These are called local indexes. Indexes can be prefixed (synchronized with the tablespace), nonprefixed (related to columns not in the partition), or global (the index is partitioned differently from the table).
• **Index-organized tables** (Oracle8)—With this type of index, the data for the table is held in the index, and changes to the data result only in changes to the index. Access can be by primary or any other key that is a valid prefix of the primary key. Standard SQL is used to access these indexes.
• **Bitmap join indexes** (Oracle9i)—This index type provides improved performance for a more specific class of join queries, specifically star joins and materialized views. When the bitmap join is executed, the large dimension table is not accessed.
Indexing Strategies

Consider the following guidelines as you develop your indexing strategy:

- Build indexes whenever columns are used in a WHERE clause, even though they are not part of a primary or foreign key.
- OLTP applications usually limit the number of indexes from four to six. Data warehouse implementations are more liberal in their indexing strategy.
- Not enough indexing or not the right indexing results in slow queries; however, too many indexes may result in increased storage and loading time.
- Do not use indexes on small tables, because the gain in performance is inconsequential when weighed against the cost of maintaining the index.
- Do not use B*trees on columns that have few distinct values, because the marginal gain in searching through the levels of the B*tree is not sufficient to incur the cost of maintaining the indexes.
- In general for star schemas, use parallelism for multiple CPUs when building indexes, use the NOLOGGING parameter, and use the COMPUTE STATISTICS option to gather index and column statistics.
**B***tree Index

The **B***tree is organized like an upside-down tree. The upper blocks (branch blocks) of a **B***tree contain index data that points to lower-level index blocks. They provide a road map to get to the right block at the leaf level. The lowest level index blocks (leaf blocks) contain every indexed data value and a corresponding ROWID that is used to locate the actual row. This is similar to the way an index in a book has a page number associated with each index entry. In general, you use **B***tree indexes when you know that your typical query refers to the indexed column and retrieves a few rows. In these queries, it is faster to find the rows by looking at the index. If you are retrieving most of the rows in a table, it might be faster to scan the entire table.

**Advantages**
- **B***tree indexes provide uniform and predictable performance for retrieval operations.
- **B***tree indexes are optimal for retrieving a single or a small set of records in a query.
- **B***tree indexes stay balanced automatically.
- **B***trees provide excellent retrieval performance for a wide range of queries, including exact match and range searches.
- Inserts, updates, and deletes are efficient, maintaining key order for fast retrieval.
- **B***tree performance is good for both small and large tables, and does not degrade as the size of a table grows.

**Note:** **B***tree indexes need to be rebalanced periodically as new data is inserted.
Bitmap Indexes

• Store values as 1s and 0s
• Are used instead of B*tree indexes when:
  – Tables are large
  – Columns have low cardinality
  – Multiple columns are constrained in the same query

Bitmap Indexes

Bitmap indexing benefits data warehousing applications, which have large amounts of data and ad hoc queries but a low level of concurrent transactions. For such applications, bitmap indexing provides:

• Reduced response time for large classes of ad hoc queries
• A substantial reduction of space usage compared to other indexing techniques
• Dramatic performance gains even on very low end hardware
• Very efficient parallel data manipulation language (DML) and loads

Fully indexing a large table with a traditional B*tree index can be prohibitively expensive in terms of space, because the index can be several times larger than the data in the table. Bitmap indexes are typically only a fraction of the size of the indexed data in the table. The advantages of using bitmap indexes are greatest for low cardinality columns, that is, columns in which the number of distinct values is small compared to the number of rows in the table. If the values in a column are repeated more than a hundred times, the column is a candidate for a bitmap index. Even columns with a lower number of repetitions and thus higher cardinality, can be candidates if they tend to be involved in complex conditions in the WHERE clauses of queries. Oracle considers a column low cardinality when the number of distinct values is small (< or = 20%) compared to the number of overall rows.

Note: Bitmap indexes are available with the Oracle8i and Oracle9i Enterprise Edition. The optimizer can use B*tree as well as bitmap indexes, even for the same query.
Bitmap Index Example

**CUSTOMER table**

<table>
<thead>
<tr>
<th>CUSTOMER_NBR</th>
<th>MARITAL_STATUS</th>
<th>REGION</th>
<th>GENDER</th>
<th>INCOME_LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>single</td>
<td>east</td>
<td>male</td>
<td>bracket 1</td>
</tr>
<tr>
<td>102</td>
<td>married</td>
<td>central</td>
<td>female</td>
<td>bracket 4</td>
</tr>
<tr>
<td>103</td>
<td>married</td>
<td>west</td>
<td>female</td>
<td>bracket 2</td>
</tr>
<tr>
<td>104</td>
<td>divorced</td>
<td>west</td>
<td>male</td>
<td>bracket 4</td>
</tr>
<tr>
<td>105</td>
<td>single</td>
<td>central</td>
<td>female</td>
<td>bracket 2</td>
</tr>
<tr>
<td>106</td>
<td>married</td>
<td>central</td>
<td>female</td>
<td>bracket 3</td>
</tr>
</tbody>
</table>

```
CREATE BITMAP INDEX REGION_IDX ON CUSTOMER(REGION);
```

Sample bitmap index on the REGION column

<table>
<thead>
<tr>
<th>REGION= 'east'</th>
<th>REGION = 'central'</th>
<th>REGION = 'west'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bitmap Index Example**

The CUSTOMER table in the slide shows a portion of a company’s customer data. Because MARITAL_STATUS, REGION, GENDER, and INCOME_LEVEL are all low cardinality columns (there are only three possible values for marital status and region, two possible values for gender, and four income levels) it is appropriate to create bitmap indexes on these columns. You should not create a bitmap index on CUSTOMER_NBR because this is a high cardinality column. Instead, a unique B*tree index on this column would provide the most efficient representation and retrieval.

Each entry or bit in the bitmap corresponds to a single row of the CUSTOMER table. The value of each bit depends upon the values of the corresponding row in the table. For instance, the bitmap REGION = ‘east’ contains a 1 as its first bit because the region is “east” in the first row of the table. The bitmap REGION = ‘east’ has a 0 for its other bits because none of the other rows of the table contain “east” as their value for REGION.
Bitmap Index Example (continued)

An analyst investigating demographic trends of the company’s customers might ask, “How many of our married customers live in the central or west regions?” This corresponds to the following SQL query:

```
SELECT COUNT(*)
FROM CUSTOMER
WHERE MARITAL_STATUS = 'married' AND
REGION IN ('central', 'west');
```

Bitmap indexes can process this query with great efficiency by merely counting the number of 1s in the resulting bitmap as shown in the slide. To identify specific customers who satisfy the criteria, you use the resulting bitmap to access the table. In the example, rows 2, 3, and 5 satisfy the query and are therefore accessed from the CUSTOMER table.
Data Placement

After you have determined the size of your database down to the individual object level and have determined your index strategy, the database administrator must determine how to store these objects physically within the database. The main task is to identify which objects reside in which Oracle tablespace. Consider parallelism when planning file systems. Increased performance can be achieved with multiple file systems.
Oracle Tablespaces

- Control disk space allocation
- Assign space quotas
- Control availability of data
- Perform partial database backup or recovery
- Allocate data storage across devices

Oracle Tablespaces

Tablespaces are used to:
- Manage data and file systems
- Control disk space allocation for database data
- Assign specific space quotas for database users
- Control the availability of data by taking individual tablespaces offline
- Perform partial database backup or recovery operations
- Allocate data storage across devices to improve performance

A database administrator has full control over the tablespaces and is able to create new tablespaces, alter existing tablespaces, alter storage characteristics, make a tablespace read-only or write-able, make a tablespace temporary or permanent, and remove tablespaces when they are no longer required.

Every Oracle database has a default tablespace called SYSTEM, which contains all the necessary data dictionary tables. Other tablespaces are created as needed.
Physical Database Structure

- Provides automatic extent allocation
- Avoids dynamic extent allocation in OLTP

It is briefly worth examining the structure of the database before discussing the specifics. Data files exist in tablespaces and contain segments that hold a specific data structure. This structure contains a set of extents that are a series of contiguous data blocks.

**Note:** If the table or index is partitioned, each partition is stored in its own segment. When more extents are needed in a segment, the Oracle server obtains one more extent. The extents may or may not be contiguous on disk. Initial and subsequent extent sizes are controlled by parameters in the `STORAGE` clause of the `CREATE TABLE` statement.

When creating a table, you can specify the size of its first extent. This should always be set to minimize any dynamic allocation of extents and to avoid fragmentation. Dynamic space allocation is also an option. However, it usually leads to fragmentation, because gaps are left in the database that are not automatically reused. Gaps such as this can be eliminated by ensuring that the `PCTINCREASE` parameter is set to 0 rather than 10, which is the default. Dynamic extent allocation is an issue in OLTP systems, but only for the Data warehouse load.

- A segment exists in only one tablespace
- A segment can span data files in a tablespace
- An extent can contain data from only one data file
Avoiding Fragmentation

- Set `PCTINCREASE` value to 0
- Keep extents a uniform size
- Remember loading and indexing activities

As mentioned earlier, when you create the object, the `STORAGE` clause also enables you to specify a `PCTINCREASE` parameter, which should be set to 0 to eliminate fragmentation. `PCTINCREASE` specifies how much each allocated extent should grow after the second extent is allocated. The default is 50, which means that an extent grows by 50% for the third and subsequent allocations.

Sizing Extents: When you create extents within a tablespace, try to keep them the same size, so that any freed extents can be immediately reallocated.

Where large objects are stored in tablespaces, make the extent size one-tenth the size of the data files for that tablespace. This enables you to monitor space utilization in a simple, effortless way.

**Note:** You also need to consider the effect on extent allocation of indexes, loading tasks, and queries that require extensive sorting.
Creating Objects in a Tablespace

Using the `CREATE` statement for both tables and indexes, and `CREATE TABLE AS` for a table, you can specify exactly in which tablespace the object is to reside. This means that you have control over:

- The physical tablespace in which the object resides, for ease of backup, recovery, and maintenance
- The initial size of that object, which can be specified in kilobytes or megabytes
- The number of extents to be allocated to reduce maintenance and fragmentation
Tablespace Features

- Tablespaces can be taken offline individually.
- Tablespaces are ideally suited for logically dividing objects.
- Tablespaces can be read-only.

Tablespace Features
Design your database carefully to ensure the best use of tablespaces. In addition to the points mentioned, you should consider the following features of tablespaces:

- Tablespaces can be taken offline individually, allowing you to maintain, backup, or recover individual objects in that area.
- Tablespaces are ideally suited for dividing objects logically.
- Tablespaces can be read-only, which can eliminate the need to perform backup and recovery on a regular basis. Obviously, you always back up with a data warehouse after new data is loaded (refreshed). The read-only facility is also a good security mechanism. A tablespace is automatically created as read-write, but you can use `ALTER` to make it read-only.
Transportable Tablespace Features

• Move data from:
  – An OLTP database to a staging database
  – A staging database to the enterprise data warehouse
  – A data warehouse to a data mart

• Archive obsolete data

Transportable Tablespaces
An enterprise data warehouse contains historical detailed data about the company. Typically, data flows from one or more online transaction processing (OLTP) databases into the data warehouse on a monthly, weekly, or daily basis. The data is usually processed in a staging database before it is added to the data warehouse. The transportable tablespace feature enables you to move a subset of an Oracle database from one Oracle database to another. Moving data by transporting tablespaces can be faster than either exporting/importing or unloading/loading of the same data, because transporting a tablespace involves only copying data files and integrating the tablespace metadata. When you transport tablespaces you can also move index data, so that you do not have to rebuild the indexes after importing or loading the table data.

Some cautions should be noted when using transportable tablespaces:
• A partitioned table must be completely contained within the set of tablespaces being made transportable.
• Oracle version must be greater than 8.10
• Must be binary compatible systems
• An index cannot be part of the transportable tablespace unless the table it references is also included within the set
• You must be able to make the table being transported Read-only for a short period
• Declare TRANSPORT_TABLESPACE=Y when invoking the export utility
Changing the Tablespace Size

- Increase the number of data files
- Create a new tablespace
- Create data files that can grow dynamically

Tablespace Size
You can change the amount of space allocated to a tablespace by performing any one of the following:
- Increasing the number of data files (shown as df n.ora) in the SYSTEM tablespace
- Creating a new tablespace with its own data files (this is the principle shown in the slide)
- Creating data files that can grow dynamically
Other Factors Impacting Size

- Block size
- PCTFREE and PCTUSED
- INITRANS and MAXTRANS
- Redo logs

Size Considerations

Block Size Every time Oracle reads from or writes to the database, it does so using database blocks. These blocks are moved in and out of the SGA. The database block size must be a multiple of the underlying operating system block size. Although the default block size for Oracle is set to 2KB, a minimum of 8KB is recommend, with 16KB preferred. In a large data warehouse, the more data that can be read in at one time, the more performance is enhanced.

PCTFREE Set this parameter to 0 to maximize all of the space available and minimize the number of data blocks required. After the block is filled during load, there is no more room available to dynamically create another transaction entry. If multiple DML update commands are to be allowed against the table, set PCTFREE to 1. PCTUSED varies only from its default of 0 if you allow deletes against the table.

INITRANS and MAXTRANS These parameters default to 1 and generally should be left as such unless multiple DML update commands are to be allowed against the table. In this case set INITRANS to 2 allowing two simultaneous updates to the table at one time.

Redo Logs These are Oracle’s journal of changes that provide the basic level of recoverability for an instance. At least three redo logs should be specified with at least 100 MB to begin with. If these are sized too small, then redo switches will occur every few seconds or minutes increasing overhead and slowing down response time. When using SQL*Loader direct-path, redo logging can be largely disabled by including the UNRECOVERABLE option as the reads and writes are directly onto Oracle data files.
Other Factors Impacting Size

- Rollback segments
- Library cache size
- Primary keys

Size Considerations (continued)

Rollback Segments: Oracle advises the use of a minimum of two rollback segments in the event that a problem occurs such as corruption of a segment, allowing some processing against the system to continue. You can calculate the number of rollback segments required by dividing the number of concurrent DML statements by four. Each segment should have a minimum of 20 extents, expanding to 40 for high-use systems. The OPTIMAL parameter should be set in conjunction with MINEXTENTS. As with redo logs, if direct path operations are in use, no rollback entries will be generated.

Library Cache: This is a shared memory pool within the SGA. When a user issues SQL statements, the SQL is loaded into the library cache where it is validated syntactically, security privileges are verified, and objects that are referenced are checked. To do this, Oracle keeps a copy of Dictionary table in cache along with the library cache to preclude the disk I/O expense. After all has been verified, the SQL along with an execution plan is placed in the library cache. This space is maintained by Oracle dynamically using a least recently used (LRU) algorithm. Statistics are maintained for this swapping out of least used blocks. Use the SHARED_POOL_SIZE initialization parameter and restart the instance.

Primary Keys: Primary key constraints generally require a concatenated index over several columns on a fact table ordered in several ways for querying, thus increasing storage requirements astronomically. Because the data does not change after it is loaded, adding this constraint to force uniqueness is fundamentally useless because the values can be checked on load.
Partitioning Tables and Indexes

Large tables and indexes can be partitioned into smaller, more manageable pieces.

Table Partitioning
Another major design issue of data in the warehouse, in addition to that of granularity, is that of partitioning. Partitioning enables you to break very large tables and indexes by dividing them into smaller parts. Partitioned tables and indexes can improve availability, ease administration, and enhance query performance in your data warehouse. A partition table is a table that is divided (partitioned) into several smaller parts, based on a range of key values that you specify. You specify storage attributes, including the physical placement, for each partition.

Logical and Physical Attributes: All partitions of a table or index have the same logical attributes, although their physical attributes may be different. For example, all partitions in a table share the same column and constraint definitions, and all partitions in an index share the same index columns, but storage specifications and other physical attributes such as PCTFREE, PCTUSED, INITRANS, and MAXTRANS may vary for different partitions of the same table or index.

B*tree and bitmapped indexes can be partitioned. Global indexes are partitioned indexes that are partitioned differently than the base table.
Advantages of Partitioning

Consider using partitioning for:

- Very large databases (VLDBs)
- Reduction of down time for scheduled maintenance and reloading
- Reduction of down time for data failure
- Decision support systems (DSS) performance
- I/O performance
- Disk striping
- Partition transparency

DSS Performance

Decision support systems (DSS) queries on very large tables present special performance problems. An ad hoc query that requires a table scan can take a long time, because it must inspect every row in the table; there is no way to identify and skip subsets of irrelevant rows. The problem is particularly important for historical tables, for which many queries concentrate access on rows that were generated recently.

Partitions help solve this DSS performance problem. An ad hoc query that requires only rows that correspond to a single partition (or range of partitions) can be executed using a partition scan rather than a table scan. For example, a query that requests data generated in the month of October 2000 can scan only the rows stored in the October 2000 partition, rather than the rows generated over many years of activity. This improves response time and can also substantially reduce the temporary disk space requirement for queries that require sorts.

Partitioned tables also resolve the problem of reloading or processing large volumes of updates against existing fact tables. By creating a new larger partition sized for growth and optimal performance, the tables (and all associated indexes) can be exchanged from the older, smaller partition avoiding the bottleneck that can occur with time consuming updates.
Methods of Partitioning

- Horizontal
- Vertical
- Range
- Hash
- Composite
- List

Methods or Partitioning
A major reason for supporting partitioned objects was the dramatic increase in the size of database objects (for example, tables) and the need to:
  - Reduce down time (due to both scheduled maintenance and data failures)
  - Improve performance through partition elimination
  - Improve manageability and ease of configuration

Oracle8i introduced hash and composite range-hash partitioning, offering improvements in one or more of the following areas for tables that do not naturally submit themselves to range partitioning:
  - Ease of specification
  - Simplicity of management for support of parallelism
  - Reduction in skew in the amount of resources required to perform maintenance operations (such as export or backup) on different partitions of a table
  - Performance, by adding support for partitionwise joins and intrapartition parallel DML
  - More advantageous use of hierarchical storage management solutions

Oracle9i introduces list partitioning allowing explicit control over how rows map to partitions and to model data distributions that follow discrete values.
Horizontal Partitioning

- Partition table and index data by:
  - Time
  - Sales
  - Geography
  - Organization
  - Line of business
- Partition by time

Horizontal Partitioning

Horizontal partitioning enables you to store a very large table in smaller tables. The database administrator controls the rows that go into each unit.

The tables can be partitioned by:
- Time
- Sales
- Geography
- Organization
- Line of business

Partitioning by Time

Horizontal partitioning by time is common. You can store 12 months of data in 12 tables, one for each month. The advantage of this, when querying data, is that full table scans are reduced. A query that requires information for a specific month merely scans a single table of data.

Partitioning by time is also effective for loading and archiving tasks. You can insert a new data table into the warehouse for each month, and easily remove (drop) the oldest table. The SQL DROP command is effective and fast.
Vertical Partitioning

- Partition data by column (rare)
- Allow access to specific user groups

Vertical Partitioning
With vertical partitioning, you break tables up on a column-by-column basis.

Use vertical partitioning when:
- It improves the speed of query and update actions.
- Users require access to specific columns. It is useful if queries use a small number of columns rather than a whole row, or you want to control visibility to sensitive data, such as salary figures on a payroll (HR) system.
- Some data is changed infrequently. You can keep the infrequently changed data in a partition separate from frequently changed data. It is easier to manage data this way, and you can make some of the attributes globally read-only.
- Descriptive dimension text can be better when it is moved away from the dimension itself.

Note: Query rewrite capability is not available when you choose vertical partitioning.
Range Partitioning

Range partitioning is based on ranges of values. This guarantees that only data with a particular set of values is contained in each partition. Range partitioning is good for rolling windows of data. The advantages of range partitioning are:

- Guarantees independence, therefore gives you good performance
- Ensures high availability because unused partitions are not touched
- Provide manageability because you do not need to maintain an even balance of data among the partitions to achieve the desired performance

Creating Partitioned Tables

- Create the table by using the PARTITION clause:

  ```sql
  CREATE TABLE sales (acct_no NUMBER(5),
    acct_name CHAR(30),
    amount_of_sale NUMBER(6),
    week_no INTEGER)
  PARTITION BY RANGE (week_no)...
  (PARTITION sales1 VALUES LESS THAN (4) TABLESPACE ts0,
   PARTITION sales2 VALUES LESS THAN (8) TABLESPACE ts1,
   . . .
   PARTITION sales3 VALUES LESS THAN (52) TABLESPACE ts12);
  ```
Hash Partitioning

- Define $2n$ partitions
- Specify `STORAGE` for the whole table
- Specify tablespace for partitions

```
CREATE TABLE product(... )
    STORAGE (INITIAL 10M)
    PARTITION BY HASH(column_list)
    (PARTITION p1 TABLESPACE TS1,
     PARTITION p2 TABLESPACE TS2);
```

Hash Partitioning

Hash partitioning reduces administrative complexity by providing many of the manageability benefits of partitioning, with minimal configuration effort. When implementing hash partitioning, the administrator chooses a partitioning key and the number of partitions. The database automatically distributes the data evenly across all partitions.

Hash partitioning enables you to do the following:
- Control the physical placement of data across a “fixed” number of partitions.
- Use a hash function on the partition columns to “stripe” data into required partitions.
- Use partition numbers to the power of 2, such as 2, 4, 16, 32, and so on.
- Partition easily for predominantly performance reasons (parallel DML, partition, elimination, piece wise joins, data that does not lend itself to range partitioning).
- Support only local indexes.

Hash partitioning is a better choice than range partitioning when:
- You do not know beforehand how much data will map into a given range
- Sizes of range partitions differ substantially
- Partition pruning and partitionwise joins on a partitioning key are important
- Cardinality and uniqueness should be as high as possible especially if you are partitioning on another key
- Tables do not have a natural partition key
Composite Partitioning in Oracle8i

- Partitions data by using the range method
- Subpartitions each partition by using the hash method:

```sql
CREATE TABLE sales
  (invoice_no NUMBER,
   sale_year INT NOT NULL,
   sale_month INT NOT NULL)
PARTITION BY RANGE (sale_year,sale_month)
  (PARTITION sales_q1 VALUES LESS THAN (1998,04)
   TABLESPACE tsa,
   PARTITION sales_q2 VALUES LESS THAN (1998,07)
   TABLESPACE tsb);
```

Composite Partitioning in Oracle8i

Composite partitioning stores data by using the range method and, within each partition, subpartitions it by using the hash method. This new type of partitioning that is available only supports historical operations data at the partition level, and parallelism (parallel DML) and data placement at the subpartition level. It is ideal for both historical data and data placement.

**Example:**
The example in the slide shows a table with two partitions for sales in the first two quarters of 1998. A row with `SALE_YEAR = 1998, SALE_MONTH = 5` has partitioning key (1998, 5); therefore, it belongs in the second partition and is stored in tablespace TSB. Similarly, a row with `SALE_YEAR = 1998, SALE_MONTH = 2` has partitioning key (1998, 2); therefore, it belongs in the first partition and is stored in tablespace TSA.

**Advantages of Composite Partitioning**

Composite partitioning:
- Provides ease-of-management advantages of range partitioning
- Provides data placement and parallelism advantages of hash partitioning
- Allows you to name the subpartitions and store them in specific tablespaces
- Allows you to build local indexes on composite-partitioned tables, which are stored in the same tablespaces as the table subpartition by default
- Allows you to build range-partitioned global indexes
- Allows you to name the index subpartitions and specify their tablespaces
List Partitioning

The list partitioning method allows the distribution of data based on discrete column values. No relationship needs to exist between partitions. This method is ideal for grouping unordered or unrelated sets of data. The following usage applies:

- Currently supports HEAP tables
- Multi-column partitioning is not supported
- Specified literal value must be unique across the value list of the object
- NULL can be specified
- MAXVALUE cannot be specified
- Listed literals’ strings cannot exceed 4K

Example:
The example in the slide shows groups states together according to their geographical location. A row is mapped to a partition by checking whether the value of the partitioning column falls within the set of values that describes the partition. The physical attributes will override the defaults; however, any partition with unspecified attributes inherit their physical attribute from the table-level defaults.
Summary

In this lesson, you should have learned how to:

- Explain data warehouse sizing techniques and test load sampling
- Describe data warehouse indexing types and strategies
- Discuss features and storage strategies for tablespaces
- Discuss partitioning methods for tables and indexes
Practice 5-1 Overview

This practice covers a review of indexing and other storage considerations.

Practice 5-1

Solutions for this exercise are located in Appendix B.
Practice 5-1 (continued)

Answer the following questions:

1. When should B*tree indexes be employed?
   _______________________________________________________________________
   _______________________________________________________________________
   _______________________________________________________________________

2. What parameter do you set in an Oracle database to avoid fragmentation?

3. Why is a parallel database of benefit to the data warehouse?
   _______________________________________________________________________
   _______________________________________________________________________
   _______________________________________________________________________
   _______________________________________________________________________
   _______________________________________________________________________

4. How does multiple parallel query work?
   _______________________________________________________________________
   _______________________________________________________________________
   _______________________________________________________________________
   _______________________________________________________________________
   _______________________________________________________________________

5. ______________ help solve decision support performance problems and also resolve the problem of reloading or processing large volumes of updates against existing fact tables.
Strategies for Extracting, Transforming, and Transporting

ILT Schedule:

<table>
<thead>
<tr>
<th>Timing</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>90 minutes</td>
<td>Lecture</td>
</tr>
<tr>
<td>20 minutes</td>
<td>Practice</td>
</tr>
<tr>
<td>110 minutes</td>
<td>Total</td>
</tr>
</tbody>
</table>
Objectives

After completing this lesson, you should be able to do the following:

• Outline the extraction, transformation, and transportation processes for building a data warehouse
• Identify extraction, transformation, and transportation issues
• Define extraction, transformation, and transportation techniques

Objectives
The extraction, transformation, and transportation processes are absolutely fundamental in ensuring that the data resident in the warehouse is:
• Relevant and useful to the business users
• High quality
• Accurate
• Easy to access so that the warehouse is used efficiently and effectively by the business users

Building the ETT process is potentially one of the biggest tasks of building a warehouse; it is complex and time-consuming. In some implementations, it can take more than half of the total warehouse implementation effort. This lesson explores ETT.
Extraction, Transformation, and Transportation Processes (ETT)

- Extract source data
- Transform and cleanse data
- Index and summarize
- Load data into warehouse
- Detect changes
- Refresh data

Extraction, Transformation, and Transportation Processes

You should be aware that extraction, transformation, and transportation describes the series of processes that:
- Extract data from source systems
- Transform and clean up the data
- Index the data
- Summarize the data
- Load data into the warehouse
- Detect the changes made to source data required for the warehouse
- Restructure keys
- Maintain the metadata
- Refresh the warehouse with updated data

You can use custom programming, gateways between database systems, and internally developed tools or vendor tools to carry out the ETT processes.
The Data Staging Area

In lesson two, we briefly discussed the need to plan a staging area or ODS. Here, we will go into a bit more detail on the operational data store.

Ralph Kimball is one of the most widely recognized experts in the field of data warehousing. Kimball calls the data staging area the construction site for the warehouse. This is where much of the data transformation and cleansing takes place.

A staging area is a typical requirement of warehouse implementations. It may be an operational data store environment, a set of flat files, a series of tables in a relational database server, or proprietary data structures used by data staging tools. You can use multitier staging that reconciles data before and after the transformation process and before data is loaded into the warehouse. As many as three tiers are possible, from the operational server to the staging area and then to the warehouse server.

Instructor Note

Oracle Warehouse Builder (OWB) supports Kimball’s model.
Preferred Traditional Staging Model

Remote staging: Data staging area in its own environment, avoiding negative impact on the warehouse environment

Possible Staging Models
The model you choose depends upon operational and warehouse requirements, system availability, connectivity bandwidth, gateway access, and volume of data to be moved or transformed. The remote staging model is considered to be the preferred method. You can choose to extract the data from the operational environment and transport it into the warehouse environment for transformation processing. You can optionally execute some transformation processing during the extraction and transportation phases from operational systems to the warehouse environment. You can then execute the bulk of transformation processing in the warehouse environment's staging area.

Instructor Note
OWB in conjunction with Pure*Extract and Pure*Integrate can provide immense flexibility to the traditional approach. All three tools allow varying levels of transformation.
Extracting Data

The process of data extraction takes selected data fields that pertain to the subject area maintained by the data warehouse. The data may come from a variety of source systems, and the data may exist in a variety of formats. The extraction routines are developed to account for the variety of systems from which data is taken. These routines contain data or business rules, as well as audit trails and error correction facilities.

Source systems may be in the form of data existing in production operational systems, archives, internal files not directly associated with company operational systems, such as individual spreadsheets and workbooks, and external data from outside the company.

Extraction routines are specifically developed to account for the variety of systems from which data is taken. The routines contain data or business rules, audit trails, and error correction facilities and take into account the frequency with which data is to be extracted.

You can extract data from different source systems to the warehouse in different ways:

- Programmatically, using procedural languages such as COBOL, C, C++, or Procedural SQL
- Using a gateway to access data sources. This method is acceptable only for small amounts of data; otherwise, the network traffic becomes unacceptably high.
- In-house developed tools that store a physical definition of the source and warehouse data, create data dictionaries, generate data conversion programs, clean and transform the data, allow selective retrieval, and maintain metadata.
Examining Source Systems

• **Production**
  - Legacy systems
  - Database systems
  - Vertical applications

• **Archive**
  - Historical (for initial load)
  - Used for query analysis
  - May require transformations

**Examining Data Sources**

Production data can come from a multitude of different sources including:

- Operating system and hardware platforms
- File systems (flat files) and database systems, such as Oracle, DB2, dBase, Informix, ISAM, NonStop
- SQL, Rdb, and TurboImage
- Vertical applications, such as Oracle Financials, SAP, Hogan, and PeopleSoft

Archive data may be useful to the enterprise in supplying historical data. Historical data is needed if analysis over long periods of time is to be achieved. Archive data is not used consistently as a source for the warehouse; for example, it would not be used for regular data refreshes. However, for the initial implementation of a data warehouse (and the first-time load), archived data is an important source of historical data.

You must consider this carefully when planning the data warehouse:

- How much historical data do you have available for the data warehouse?
- How much effort is necessary to transform it into an acceptable format?

The data warehouse may need some careful and unique transformations, and clear details of the changes must be maintained in metadata.
Examining Source Systems

• Internal
  – Business plans
  – Forecasts
  – Budgets
  – Annual report

Examining Data Sources
Internal data may be information prepared by planning, sales, or marketing organizations that contains data such as budgets, forecasts, or sales quotas. The data contains figures (numbers) that are used across the enterprise for comparison purposes. The data is maintained using software packages such as spreadsheets and word processors and uploaded into the warehouse. Internal data is treated like any other source system data. It must be transformed, documented in meta data, and mapped between the source and target databases.
External data is important if you want to compare the performance of your business against others. There are many sources for external data:
• Periodicals, newspapers, and reports
• External syndicated data feeds (some warehouses rely regularly on this as a source)
• Competitive analysis information such as purchased marketing, competitive, and customer related data
• Free data from the Web
**Examining Source Systems**

- **External**
  - Dunn and Bradstreet
  - A. C. Nielsen, IRI, Walsh America
  - Barron’s
  - Wall Street Journal

---

**Examining Data Sources**

Issues to consider for external data:

- **Frequency**—No real pattern like that of internal data. Constant monitoring is required to determine when it is available.
- **Format**—Data may be different in format than internal data, and the granularity of the data may be an issue. In order to make it useful to the warehouse a certain amount of reformatting may be required. In addition, you may find that external data, particularly that available on the Web, comes with digital audio data, picture image data, and digital video data. These present an interesting challenge with regard to storage and speed of access.
- **Predictability**—External data is not predictable; it can come from any source at any time, in any format, on any medium.
Mapping

• Defines which operational attributes to use
• Defines how to transform the attributes for the warehouse
• Defines where the attributes exist in the warehouse

Mapping Data
After you have determined your business subjects for the warehouse, you should determine the required attributes from the source systems. On an attribute-by-attribute basis you must determine how the source data maps into the data warehouse, and what, if any, transformation rules to apply. This is known as mapping. Mapping information should be maintained in metadata that is server (RDBMS) resident, for ease of access, maintenance, and clarity.

Instructor Note
OWB, Pure*Extract and Pure*Integrate all provide mapping capability.
Designing Extraction Processes

- **Analysis**
  - Sources, technologies
  - Data types, quality, owners
- **Design options**
  - Manual, custom, gateway, third-party
  - Replication, full, or delta refresh
- **Design issues**
  - Batch window, volumes, data currency
  - Automation, skills needed, resources
- **Maintenance of metadata trail**

**Designing Extraction Processes**

When designing your extraction processes, consider the analysis issues, the design options available to you, and the design issues.

- Analysis issues—Existing data feeds and redo logs, Data types (EBCDIC or ASCII), Data volumes, Operational schedule in the source environment, Spare processing capacity in the source environment
- Design options—Manual data entry, custom programs, gateway technologies, replication techniques, third-party tools, full refresh or delta changes
- Design issues—Data volumes, data currency (how up-to-date the data is to be), degree of automation required
- Resources—technology skills needed, and time and money available
Importance of Data Quality

- Business user confidence
- Query and reporting accuracy
- Standardization
- Data integration

Importance of Data Quality
We have explored the fundamentals for extracting data from source systems. Now we need to consider the quality of the data that is suitable for end-user query and analysis applications. The importance of quality data in the data warehouse cannot be overemphasized. Although data anomalies are bound to exist in source systems, if they are allowed to get into the data warehouse this leads to inaccurate information, which further leads to inaccurate reports and bad business decisions.

The overall result is a lack of confidence in the system to deliver the solution and a data warehouse that either is not used or requires substantial improvement and management buy-in. Quality data is the key to a successful warehouse; it is better to have no data at all than bad data.
Benefits of Data Quality

Cleansed data is critical for:
• Standardization within the warehouse
• High quality matching on names and addresses
• Creation of accurate rules and constraints
• Prediction and analysis
• Creation of a solid infrastructure to support customer-centric business intelligence
• Reduction of project risk
• Reduction of long term costs

Benefits of Quality Data
• Target the right audience for marketing communication
• Determine that a particular customer buys related products
• Determine that a group of people form a family, each of whom is a potential customer (householding)
• Identify that an organization is part of a larger enterprise (commercial householding)
• Identify that a customer is part of another organization due to acquisition or take over
• Match customers where there are many different records for the same customer. (For example, the different components of health care, such as the hospital, the pharmacy, and the doctor have their own records, or a patient may be treated by different physicians in the same hospital.)
• Identify the age of data and its history

Note: The terms scrubbing, cleaning, cleansing, and data reengineering are used interchangeably.
Guidelines for Data Quality

- Operational data should not be used directly in the warehouse.
- Operational data must be cleaned for each increment.
- Operational data is not simply fixed by modifying applications.

Guidelines for Data Quality

Do not assume that because the data in the operational system suits you at the operational level, it is going to be appropriate, suitable, and of a sufficiently high quality for the data warehouse.

- The operational system contains no aging information.
- There are many examples of disparity in the data including many different meanings being applied to data.
- Good operational data when merged may become poor data warehouse data. For example, Item number is a unique key in the product file and is also contained in a historical promotions files. If the item number changes over time, the two files might not be able to relate to each other in the data warehouse due to constraints on the separate tables and lack of summary table information.

Do not assume it is acceptable to clean up data after the pilot run of the first increment or implementation.

- The credibility of the data warehouse or data mart suffers. Post implementation cleanups are more costly and the risk is higher than during the pilot.
- The programs needed to handle the multitude of problems are very complex and would need to be rewritten after cleanup.

Do not assume that fixing applications at the point of entry is going to satisfy quality and clean up the data for the future. It is often too time-consuming and costly to implement changes at that level. Changes cannot be implemented quickly enough to keep up with constantly changing operational requirements. The cost in time and resources in reengineering the existing legacy data may be too high.
Transformation

Transformation involves a number of tasks, the most important being to eliminate all anomalies. Cleaning also includes eliminating formatting differences, assigning datatypes, defining consistent units of measure, and determining encoded structures. Along with these tasks, another objective is to ensure that the data is presented in a subject-oriented fashion. Many potential problems can exist with source data:

- No unique key for individual records
- Anomalies within data fields, such as differences between naming and coding (datatype) conventions
- Differences in the interpreted meaning of the data by different user groups
- Spelling errors and other textual inconsistencies (this is particularly relevant in the area of customer names and addresses)
Transformation Routines

- Cleansing data
- Eliminating inconsistencies
- Adding elements
- Merging data
- Integrating data
- Transforming data before load

Transformation Routines
The transformation process comprises a number of tasks, one of which is cleaning the data, also referred to as data cleansing or scrubbing.

Transformation routines:
- Adding an element of time to the data, if it does not already exist
- Translating the formats of external and purchased data into something meaningful for the warehouse
- Merging rows or records in files
- Integrating all the data into files and formats to be loaded into the warehouse

Transformation should be performed:
- Before the data is loaded into the warehouse
- In parallel (on larger databases, there is not enough time to perform this process as a single threaded process)
- The process should be self-documenting, should generate summary statistics, and should process exceptions
Why Transform?

- In-house system development
- Multipart keys
- Multiple encoding
- Multiple local standards

Why Transform?

- In-house system development takes place over many years and often uses different software and standards for each implementation.
- Multipart keys—Many older operational systems used record key structures that had a built-in meaning. To allow for decision support reporting, these keys must be broken down into atomic values. The fact data must be at a granular level.
- Multiple encoding—Some systems may represent values in different ways. For example, some systems use M to denote “male” and F to denote “female”, while others use 1 and 0, or even NULL values.
- Multiple local standards—This is particularly relevant for values entered in different countries. For example, some countries use imperial measurements and others metric; currencies and date formats differ; currency values and character sets may vary; and numeric precision values may differ. Currency values are often stored in two formats, a local currency such as sterling, French francs, or Australian dollars, and a global currency such as U.S. dollars.
Why Transform?

- Multiple files
- Missing values
- Duplicate values
- Element names

Why Transform? (continued)

- Multiple files—The source of information may be one file for one condition, and a set of files for another. Logic (normally procedural) must be in place to detect the right source. The complexity of integrating data is greatly increased according to the number of data sources being integrated. For example, if you are integrating data from two sources, there is a single point of integration where conflicts must be sorted. Integrate from three sources, and there are three points of conflict. Four sources provide six conflict points. The problem is exponential.
- Missing values—Null, missing, and default values are always an issue. NULL values may be valid entries where NULLs are allowed; otherwise, NULLs indicate missing values.
- Duplicate values—You must eliminate duplicate values, which invariably exist. This can be time-consuming, although it is a simple task to perform.
- Element names—Individual attributes, columns, or fields may vary in their naming conventions from one source to another. These should be eliminated to ensure that one naming convention is applied to the value in the warehouse. If you are using independent data marts, then you should ensure that the ETT solution is mirrored; if you plan to use the data marts dependently in the future, then they will all refer to the same object.
Why Transform?

- Element meaning
- Input format
- Referential integrity

Why Transform? (continued)

- Element meaning—Like the name of an element, the meaning is often interpreted differently by different user groups. The variations in naming conventions typically drive this misinterpretation. Keep your model independent of naming conventions that are popular today, but subject to change.
- Input format—Input formats vary considerably. For example, one entry accepts alphanumeric data, so the format may be “123-73”. Another entry accepts numeric data only, so the format is “12373”. You may also need to convert from ASCII to EBCDIC, or even convert complex character sets such as Hebrew, Arabic, or Japanese.
- Referential integrity—If the constraints at the application or database level have in the past been less than accurate, then child and parent record relationships can suffer; orphaned records can exist. You must understand data relationships that are built into legacy systems. The biggest problem you encounter here is that they are often undocumented. You must gain the support of users and technicians to help you with analysis and documentation of the source data.
Why Transform?

Name and address:
- No unique key
- Missing data values (NULLs)
- Personal and commercial names mixed
- Different addresses for the same member
- Different names and spelling for the same member
- Many names on one line
- One name on two lines
- The data may be in a single field of no fixed format
- Each component of an address is in a specific field

Why Transform? (continued)
Name and address—One of the largest areas of concern, with regard to data quality, is how name and address information is held, and how to transform it. Name and address information has historically suffered from a lack of legacy standards. This information has been stored in many different formats, sometimes dependent upon the software or even the data processing center that is used. The solution to these problems is to perform the following tasks:
- Create atomic values
- Standardize formats
- Verify data accuracy
- Match with other records
- Identify private and commercial addresses and inhabitants
- Document in metadata

Instructor Note
Pure*Integrate provides the solution for these complex problems. Name and address cleansing is performed using the Pure*Name Pure*Address module. Cleansing can be performed on both personal and business names. Cleansing includes address parsing, validating, correcting, standardizing, augmenting, and reformatting. Cleansed names and addresses greatly improve the process of matching and merging customer data.
Integration (Match and Merge)

- Integration provides a full view of customer data.
- Integration is accomplished by using match rules and merge rules.
  - Match rules provide a mechanism to identify the same customers.
  - Merge rules select the best values for the final integrated view of customers and all their associated data.
Transformation Techniques

- Merging data
  - Operational transactions do not usually map one-to-one with warehouse data.
  - Data for the warehouse is merged to provide information for analysis.
- Adding keys to data

Transformation Techniques

- Merging data—An operational transaction does not usually have a one-to-one mapping with data in the warehouse, even if the data in the warehouse is maintained at the transaction level. For example, consider a sales transaction in a store. The logical transaction comprises a number of components such as date of sale, charge amount, number of items, discount amount, and payment method. The transaction may even be a return. A customer purchase and a customer return are very different types of sales transactions, and different business rules must apply. For each different transaction a different process occurs. A purchase depletes inventory and a return adds stock back into inventory. The result is, for the warehouse, that the data you are keeping is held for purely reporting purposes and these transactions become merged into data that is useful for that purpose. The data will not, in the end, map strictly to sales or returns.
- Adding keys to data—You are moving the data from one structure, with its keys defining relationships, into another that is totally different and must also have keys defining relationships. The transformation of this data also includes adding keys (generalized or artificial) or creating keys from existing data values.
Transformation Techniques

Time

Transformation Techniques (continued)
Time is important within the data warehouse. You have already looked at the time dimension, which is always created in the warehouse in order to provide reporting by time periods. Extracted source data probably does not contain time information, because it is not typical of time-stamp information in operational systems (unless of course they too are maintaining history, or time is a critical component). More likely the record in the operational system has a value associated with it, such as Order_date, Ship_date, or Call_date. Therefore it is important to consider how you are going to add a time element to your warehouse data.
Transformation Techniques

Adding a date stamp:

- **Fact table**
  - Add triggers
  - Recode applications
  - Compare tables

- **Dimension table**

- **Time representation**
  - Point in time
  - Time span

Transformation Techniques (continued)

Adding a date stamp—This is particularly important for two areas of the warehouse, namely fact tables and dimension tables.

- Fact tables that hold vast amounts of data used to analyze the business according to time periods. You might choose from a number of techniques: coded application or database triggers at the operational level to time-stamp data, which can then be extracted using date selection criteria. 1) Perform a comparison of tables, original and new, to identify differences. 2) Maintain a table containing copies of changed records to be loaded. You must decide which are the best techniques for you to use according to your current system implementations.

- Dimension data containing criteria by which you perform the analysis. Dimensions change also and there are many different techniques you can use to trap changes. Three of these methods were discussed in lesson 3 in the topic titled “Slowly Changing Dimensions.”

- The time may be represented as: 1) a single point-in-time date; and 2) a date range (start and end date). The time element must either be available in the data before loading into the warehouse, or added when loading the data.
Transformation Techniques

Creating summary data:
• During extraction on staging area
• After loading onto the warehouse server

Transformation Techniques (continued)
Creating summary data—Creating summary data is essential for improving performance in the data warehouse. It is classified here under transformation only because you are changing the way the data exists in the source system into something else for the data warehouse. In reality, the summary data is usually created on the warehouse server after transformation. You can summarize the data:
• At the time of extraction in batch routines. This reduces the amount of work performed by the data warehouse server, because all the effort is concentrated on the source systems. However, summarizing at this time increases: 1) the complexity and time taken to perform the extract; 2) the number of files created; 3) the number of load routines; and 4) the complexity of the scheduling process.
• After the data is loaded into the warehouse database. The process queries the fact data, summarizes it, and places it into the requisite summary fact table. This method reduces the complexity and time taken for the extract tasks. However, it places all the CPU and I/O intensive work on the warehouse server, thus increasing the time that the warehouse is unavailable to the users.
• Weigh the benefits of each method and determine your strategy according to your requirements and resources.
Transformation Techniques

Creating artificial keys:
- Use generalized or derived keys
- Maintain the uniqueness of a row
- Use an administrative process to assign the key
- Concatenate operational key with number
- Easy to maintain
- Cumbersome keys
- No clean value for retrieval

![Diagram showing key concatenation]

Transformation Techniques (continued)

Create Artificial Keys An artificial (generalized or derived) key can be used to guarantee that every row in the table is unique. The warehouse data may likely be a combination of many transformed records, of which there are no natural data keys to use as unique identifiers.

Concatenate Operational Key with a Number Combine synthetic key with the primary key and version digit or characters. For example, if a customer record key value contains six digits, such as 109908, the derived key may be 10990801. The last two digits are the sequential number generated automatically.

The advantage of this method is that it is relatively easy to maintain and set up the necessary programs to manage number allocation. The disadvantages of this method are that the keys may become long and cumbersome and there is no clean key value for retrieval of a record, unless you have another copy of the key.
Where to Transform?

Choose wisely where the transformation takes place:
• Operational platform
• Staging area
• Warehouse server

Where to Transform?
You must consider carefully when and where you perform transformation. You must perform transformation before the data is loaded into the warehouse, and in parallel; on larger databases, there is not enough time to perform this process as a single threaded process. Consider the different places and points in time where transformation may take place.
• Operational platform—This approach transforms the data on the operational platform, where the source data resides. The negative impact of this approach is that the transformation operation conflicts with the day-to-day working of the operational system. If it is chosen, the process should be executed when the operational system is idle or less utilized. The impact of this approach is so great that it is very unlikely to be used.
• Separate staging area—This approach transforms data on a separate computing environment, the staging area, where summary data may also be created. This is a common approach because it does not affect either the operational or warehouse environment. Cleaning, merging, and removal of anomalies are handled in the staging area, and summary creation can take place on the staging server or on the warehouse server.
• Warehouse server—You may consider performing transformations on the warehouse server itself. However, this may affect the effectiveness of the server for query access. It is more likely that transformation will be handled away from the warehouse server, however that will depend on the method or tools that are chosen to enable the transformation process.
When to Transform?

Choose the transformation point wisely:
• Workload
• Environment impact
• CPU use
• Disk space
• Network bandwidth
• Parallel execution
• Load window time
• User information needs

When to Transform?
The approach you choose depends upon operational requirements. You must balance many different factors to determine the best solution. Consider:
• The actual workload (time to complete) of the transformations needed to provide the data for the warehouse
• The physical impact on each of the environments you choose (this is particularly relevant if you choose to use the operational platform)
• The available CPU and disk space (for temporary and intermediate data and file store) on each environment
• The available network and bandwidth between environments, affecting transfer volumes
• Whether the environment is capable of working in a parallel manner
• The load window time constraints
• The information needs of business users (When do they need this data? How often do refreshes occur?)

Monitoring and tracking The transformations should be self-documenting, should generate summary statistics, and should be able to process exceptions.
Designing Transformation Processes

• Analysis
  – Sources and target mappings, business rules
  – Key users, metadata, grain, verify integrity of data
• Design options
  – PL/SQL, replication, custom, third-party tools
• Design issues
  – Performance
  – Size of the staging area
  – Exception handling, integrity maintenance

Designing Transformation Processes
When designing your transformation processes, consider the analysis issues, the design options available to you, and the design issues.
• Analysis issues—Source and target mappings, business rules, key users, metadata, granularity of the fact data and summaries, verification of data integrity
• Design options—PL/SQL, replication, custom 3GL programs, third-party tools
• Design issues—Performance and throughput, sizing the staging areas to hold the data to be loaded into the warehouse, exception handling, integrity maintenance
Transporting Data into the Warehouse

- Loading moves the data into the warehouse.
- Subsequent refresh moves smaller volumes.
- Business determines the cycle.

The transportation process moves data from source data stores or an intermediate staging area and loads it into the target warehouse database in the target system server. This process comprises a series of actions, such as moving the data and loading data into tables. There may also be some processing of objects after the load, often referred to as post-load processing.

**Moving and Loading Data**
To move and load the data can be a time-consuming task, depending upon the volumes of data, the hardware, the connectivity setup, and whether parallel operations are in place. The time period within which the warehouse system can perform the load is called the load window. Loading should be scheduled and prioritized. You should also ensure that the loading is automated as much as possible.

**Types of Data Load**
There is a single first-time load that moves large volumes of data when the warehouse is implemented. The first-time load is followed by regular refreshes of the warehouse with smaller volumes of data, the grain and frequency of which is determined by business user requirements.
Extract Processing Environment

Data Refresh Models First, to ensure that you understand how the warehouse data presentation differs from nonwarehouse data presentation, consider how up-to-date data is presented to users in two different decision support environments: a simple extract processing environment and a data warehouse environment.

Extract Processing Environment A snapshot of operational data is taken at regular time intervals: T1, T2, and T3. At each interval a new snapshot of the database is created and presented to the user; the old snapshot is purged.

Warehouse Environment An initial snapshot is taken and the database is loaded with data. At regular time intervals, T1, T2, and T3, a delta database or file is created and the warehouse is refreshed. A delta contains only the changes made to operational data that need to be reflected in the data warehouse.

- The warehouse fact data is refreshed according to the refresh cycle determined by user requirements analysis.
- The warehouse dimension data is updated to reflect the current state of the business, only when changes are detected in the source systems.
- The older snapshot of data is not removed, ensuring that the warehouse contains the historical data needed for analysis.
- The oldest snapshots are archived or purged only when the data is not required any longer.
First-Time Load

• Single event that populates the database with historical data
• Involves a large volume of data
• Uses distinct ETT tasks
• Involves large amounts of processing after load

First-Time Load
The first-time load (sometimes called an initial load) is a single event that occurs prior to implementation. It populates the data warehouse database with as much data as needed or available. The first-time load moves data in the same way as the regular refresh. However, the complexity of the task is made greater due to:
• Data volumes that may be very large (Your company decides to load the last five years of data, which may comprise millions of rows. The time taken to load the data may be in days rather than hours.)
• Distinct extraction and transformation tasks that are applicable only to this older data
• The task of populating all fact tables, all dimension tables, and any other ancillary tables you have created such as reference tables
• Post-processing of loaded data, with tasks that must work on the large data volumes, such as indexing and key generation
• Post-load processing on large volumes of data, such as creating summary tables

With all the issues surrounding a first-time load, it is a task not to be considered lightly. You must plan, prepare, and have recovery capabilities built in to your processing routines to ensure success.
Refresh

- Performed according to a business cycle
- Simpler task
- Less data to load than first-time load
- Less complex ETT
- Smaller amounts of postload processing

Refresh

After the first time load, the refresh is performed on a regular basis according to a cycle determined by users. The cycle may be daily, weekly, monthly, quarterly, or any other business period. The refresh is a simpler task than a first-time load for these reasons:
- There is less fact data to load. You are moving a new snapshot of data but not all fact data into the data warehouse.
- There is no dimension data to load (unless your model has changed, which would be an exception). There may be some dimensional data changes to incorporate.
- Less complex extraction and transformation processes may be needed. Additionally, because these processes are executed regularly, they can be monitored, tested, and improved for each refresh until they run as optimally as possible.
- Post-load processing time is reduced and there is less new data to work with.
Building the Transportation Process

Specification:
- Techniques and tools
- File transfer methods
- The load window
- Time window for other tasks
- First-time and refresh volumes
- Frequency of the refresh cycle
- Connectivity bandwidth

Building the Transportation Process

Specifying the Process You must identify early on in the development process how you are going to move the data from the source systems into the data warehouse. You must identify:
- The data movement techniques and tools that are available
- File transfer methods and transfer models that are available
- The time available to load the data into the warehouse—the load window
- Whether the time window is sufficient for other tasks such as backup, preventative maintenance, and recovery, given expected performance metrics
- The volumes of data involved in the first-time load and subsequent refreshes
- The frequency of the refresh cycle and the grain of the data
- Connectivity bandwidth
Building the Transportation Process

- Test the proposed technique
- Document proposed load
- Gain agreement on the process
- Monitor
- Review
- Revise

Building the Transportation Process (continued)

Testing the Process Test the proposed technique to ensure that volumes can be physically moved within the load window constraints and network capabilities.

Documenting the Process You must communicate and document the proposed load with the operations organization to ensure their agreement and commitment to this important process.

Monitoring, Reviewing, and Revising the Process Ensure that the load is constantly monitored and reviewed, and revise metrics where needed. Warehouse data volumes grow rapidly, and metrics for load and data granularity require regular revision.
Granularity

- Important design and operational issue
- Low-level grain: Expensive, high level of processing, more disk, detail
- High-level grain: Cheaper, less processing, less disk, little detail
- Space requirements
  - Storage
  - Backup
  - Recovery
  - Partitioning
  - Load

Granularity
You have seen that the grain of the data is important in the warehouse environment. The lower the level of granularity, the more data is loaded, and this affects the amount of time taken to load the data into the warehouse.

**Low-Level Grain** Low-level grain data can be expensive to build and maintain. It requires a large amount of processing power to process the details and provide answers to business queries. It takes up more disk space and can create response time problems. However the detail provides the information needed at a low level to provide sophisticated business analysis.

**High-Level Grain** High-level grain data is easier to build and maintain than low-level grain data. It requires less processing power and disk space, allows a higher number of concurrent users to access data, and performs well. However, the lack of detail and drilldown capability hinders definitive answers to business questions.

**Note:** The level of granularity affects not only the amount of direct access storage devices (DASD) required for warehouse data, but also the amount of space required for backup, recovery, and partitioning.
Transportation Techniques

Now that you have seen how to capture the data needed for the refresh, consider how to physically move the data to the warehouse server.

The following common techniques are used to transport data into the warehouse:

- **Tools**
  - SQL Loader Direct Path
  - Utilities and 3GL
  - Gateways
  - Customized copy programs
  - Replication
  - FTP
  - Manual
- **Transportable tablespaces** (to be covered in the following slides)
Transportable Tablespaces

- Copy database subsets (tablespaces) between databases
- Fast
- Bulk copy using operating systems
- Managed transfer of metadata

Moving data from a data warehouse to a data mart or from an OLTP system to a staging area for a data warehouse can be cumbersome and time consuming. Direct path loading with SQL*Loader or parallel DML makes the task faster, but the process should be simpler for data movement between identical databases. Oracle8i provides a mechanism for copying data files between identical systems and allows the same data to be accessed by both systems, which improves performance and provides operational simplicity for the transfer of data. Provided the tablespaces are read-only and provided the original tablespaces (design considerations) are worthy of transporting entirely, the most effective method is transportable tablespaces.
Transportable Tablespaces: Uses and Restrictions

- Source and target servers must:
  - Be on the same operating system
  - Run Oracle 8i or above
  - Have the same block size
  - Use the same character set

- Bitmap indexes and tables containing nested tables or VARRAYs cannot be transported.

Transportable Tablespaces (continued)
The set of tablespaces transported in each run must be self-contained. A partitioned table must be fully transported, otherwise they must be moved one at a time as follows: 1) create a table by selecting from partition at source; 2) transport the tablespace containing resulting table; 3) exchange partitions at target; and 4) drop table at source.
The tablespace that is moved can contain tables with LOBs and user-defined data types. If a table with a BFILE column is part of the tablespace that is moved, the user must copy the referenced files to the target.
Object references (REFs) may become dangling pointers after a transport operation.
The user is responsible for resolving dependencies between objects in the tablespaces that are transported and those in the target database. The PL/SQL procedure, TRANSPORT_SET_CHECK within the DBMS_TT package, can be used to verify that a set of tablespaces is self-contained. This procedure accepts two IN arguments: a comma separated list of tablespaces (VARCHAR2) and a Boolean argument specifying whether to check for referential integrity constraints. The procedure populates the TRANSPORT_SET_VIOLATIONS table indicating which objects in the tablespaces that are specified have relationships to objects outside of the specified set.
Note: Transportable tablespaces can also be used for archiving.
SQL*Loader

- SQL*Loader direct path for large loads
- SQL*Loader direct path parallel for extremely large loads

SQL*Loader Direct Path
SQL*Loader is a general-purpose utility provided by Oracle for loading databases. It takes raw data and loads it into Oracle tables using a control file to specify the format of the input data file and to map the corresponding fields into the correct columns.

SQL*Loader has two modes of operation: conventional path and direct path. Conventional path can incur some significant overhead because each row is processed through the SGA’s buffer cache and performs constraint checking, and so on.

Rather than constructing SQL INSERT statements and processing these against the SGA by filling a bind array buffer, SQL*Loader uses a private memory space where it writes preformatted database blocks directly to the tablespaces. To utilize this mechanism, set the DIRECT parameter to TRUE.

When extremely large load files are to be used, set the PARALLEL parameter to TRUE and concurrently perform direct path parallel loads. Parallel loader does not use allocated space above the high water mark, but allocates separate new temporary segments in the same table space and loads data into the temporary segments, thereby allowing multiple processes to work simultaneously. To avoid bottlenecks on redo logs, switch on the UNRECOVERABLE option of SQL*Loader. There is no need to write changes to redo logs in this environment.

Recall our discussion on partitioning in Lesson 5. The strategy of exchanging partitions and turning subsequent periodic loads into “initial” loads works very well. By creating an empty partition and swapping out a large fact table into a new partition (rather than reloading or updating), you can minimize the impact on users.
ETT Tools and Gateways

• Tools are comprehensive but costly.
• Gateways are not always the fastest method:
  – Access other databases
  – Supply dependent data marts
  – Support a distributed environment
  – Provide real-time access if needed

ETT Tools and Gateways

Tools If your IT group has decided to use a customized ETT tool, then it becomes the means by which your data is transported, as well as extracted and transformed. This is not the most common option, particularly for early implementations. Often, because of the cost, copy utilities are the logical alternative. A very efficient, MVS native migration tool that can be used to create sequential or load image files and works cooperatively with Oracle Warehouse Builder is Oracle Pure*Extract.

Gateways A gateway is a middleware component that presents a unified view of data coming from different data sources. Of note are Oracle Transparent Gateways (or Procedural Gateways), Open Database Connectivity (ODBC) tools, which present a uniform view of a database other than an Oracle database, or a file on specific file systems. Oracle gateways are a mixture of read-only, whereas other gateways are read-write.

You should consider using gateway technology in specific instances only, and not on a regular basis. For example, using gateway technology allows you to access a database that is not an Oracle database directly, without executing the usual extract programs. Performing a simple SQL SELECT to access data that is to be processed for the warehouse is faster than building a specific extract for the task.

Gateway technology also gives you the ability to develop warehouses on distributed environments, use technologies (hardware and software) that are not Oracle-specific.
Customization and Replication

- Use customized programs as a last resort.
- Replication is limited by data-transfer rates.

Customization and Replication
If you use Oracle for your warehousing environment, SQL*Loader is recommended. Use customized programs only as a last resort. Replication is rarely used in a data warehouse environment, because of the limitations of data-transfer rates. It is customary to use SQL*Loader or in-house-developed loading techniques. If replication is used, it is more likely to be used to feed data marts from a larger warehouse. **Note:** Replication is not recommended for moving large volumes of data.
Post-Processing of Loaded Data

You have now seen how to extract data to an intermediate file store or staging area, where it is then transformed into acceptable warehouse data and transported to the warehouse server. You have also seen how the ETT process is slightly different for a first-time load, which requires all data to be loaded once and for refreshing, which requires only changed data to be loaded. You now must consider the different tasks that might take place after the data is loaded. There are various terms used for these tasks. In this course the choice of terms is post-processing. Post-processing tasks are not definitive; you may or may not have to perform them, depending upon the volumes of data moved, the complexity of transformations, and the transportation mechanism. For example, it is possible to load data using SQL*Loader in a manner that excludes database trigger processing. However, at the warehouse server, you want to ensure the triggers are executed so that the integrity and validity of data are retained. Other post-processing considerations include:

• Creating indexes
  – Generally faster after the load
  – For Primary Key or Unique Constraint, incur overhead during load by maintaining the index on load
  – Drop global indexes for partitions and maintain local indexes before load when using SQL*Loader direct path
• Creating summary tables (reviewed in the next lesson)
Summary

In this lesson, you should have learned how to:

- Outline the extraction, transformation, and transportation processes for building a data warehouse
- Identify extraction, transformation, and transportation issues
- Define extraction, transformation, and transportation techniques
Practice 6-1 Overview

This practice covers understanding extraction, transformation, and transportation.

Practice 6-1
Solutions for this exercise are located in Appendix B.
Practice 6-1 (continued)
Answer the following questions:

1. The ___________ staging model is considered to be the best because the data staging area is in its own environment, avoiding negative impact on the warehouse.

2. The process of ___________ takes selected data from source fields that pertain to the subject area maintained by the data warehouse.

3. ___________ _____ is the key to a successful warehouse because data anomalies can lead to inaccurate information and bad business decisions.

4. ___________ involves the elimination of anomalies, standardization of formatting, assignment of data types, definition of measure, and determination of encoded structures.

5. Although transformation may be effected on the operational platform or at the warehouse server level, a separate ___________ area is considered to be the preferable approach because it does not affect either the operational or warehouse environment.

6. The _____________ process moves data from source data stores or an intermediate staging area and loads it into the target warehouse database in the target system server.

7. Oracle8i provides a very fast, effective method to move data from one database to another. This methodology is called ___________ ____________.
## Summary Management

### ILT Schedule:

<table>
<thead>
<tr>
<th>Timing</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>75 minutes</td>
<td>Lecture</td>
</tr>
<tr>
<td>20 minutes</td>
<td>Practice</td>
</tr>
<tr>
<td>95 minutes</td>
<td>Total</td>
</tr>
</tbody>
</table>
Objectives

After completing this lesson, you should be able to do the following:

• Discuss summary management and Oracle implementation of summaries
• Describe materialized views
• Explain the cost-based optimizer and query rewrite
• Understand star query optimization
• Identify Oracle dimensions

Lesson Aim

This lesson introduces the concepts of aggregate data and the use of materialized views. Additionally, the cost-based optimizer and dimensions are explored.
What Is a Summary?

A table that:

• Stores presummarized data
• Is based on user query requirements

What Is a Summary?

A summary (or aggregate) is a fact table record that condenses base-level fact table records. The aggregate stores computed results such as sums, counts, and averages and improves performance for queries. The need to define summaries is often linked to the overriding need to contain costs and enhance performance. With costs increasing due to storage requirements for grain data (growing algebraically over a period of five to ten years) some form of capacity planning must be realized. Scalability as a result of aggregation becomes more feasible. Summary tables (an extension of the physical design) are important in the design of the data warehouse because they:

• Improve service to end users by providing better response time to analytical queries
• Provide improved and optimized use of resources, storage, and CPU
• Enhance the analysis process that allows you to drill down from higher levels of detail, and drill up from lower levels of detail

Oracle8i replaced standard user summaries with materialized views.
Implementing Summaries

- Fact and dimension data: Stored in relational tables
- Summaries: Created as materialized views
- Relationships for rollup: Implemented using referential integrity constraints or dimensions

Implementing Summaries

Fact and dimension data Data is stored in relational tables allowing sales data (facts), for example, to be stored in the SALES table. Dimension data can be stored either in a single table or in multiple tables.

Summaries Materialized views store summaries. While creating materialized views, storage options can be specified to control their size and location. Consider using CUBE and ROLLUP operators (an extension of GROUP BY) to compute multiple aggregates over combinations of dimensions.

Dimensions Dimensions are purely metadata. They define relationships between columns in one or more tables. Dimensions are discussed in greater detail in a subsequent section.
Summary Navigation

- Effective use of summary tables requires summary table awareness.
- Methods for summary navigation:
  - Warehouse database engine
  - Proprietary summary-aware products
  - Open summary-aware middleware
  - 3GL and metadata solutions

Summary Navigation

Having developed summary tables, you are now challenged with using them appropriately. The tool or the query mechanism must be summary table-aware. In other words, the existence of the summary tables must be known to the query.

Summary (Aggregate) Navigators Summary navigators are software components that intercept the end user’s SQL and transform it so that it uses the best available summary (usually the smallest available table that can answer the user’s request). The summary navigator maintains special metadata, which describes the current profile of summary tables stored in the data warehouse. It also should maintain statistics on queries, which shows which aggregates are being used and which should be built to help slow-running queries.

Methods for Summary Navigation Summary navigators can be located in:
- The warehouse database engine—This is the best-case scenario. Because this approach makes summary redirection (or query rewrite) accessible to all applications. For example, Oracle9i contains its own built-in summary navigator.
- End-user query tools—In recent years, this approach has been the most common, one primarily due to the fact that few database engines incorporated their own navigator. However, this approach requires that all query tools maintain their own summary navigation facility and metadata layer.
- Middleware tools that facilitate the navigation to the summary tables (older systems)
- Your own summary navigation technique using 3GL code and metadata (older systems)
Managing Historical Summary Data in the Warehouse

Developing a strategy to manage summary tables in the warehouse is another major design consideration. Because use of the data is the most important factor in determining which summaries should be created, you may not be able to determine this strategy immediately. Summaries do not have to be consistently applied across the warehouse. For example, you may want to examine more recent data in greater detail than older data. Therefore, you might keep daily data for the last twelve months, along with summarized monthly data. Older data might be summarized to the month, quarter, or year.
What is Summary Management?

The summary management process begins with the creation of dimensions and hierarchies that describe the business relationships and common access patterns in the database. Fact tables representing business transactions are joined to dimensions through keys. Oracle’s summary management capability consists of the following features:

- Mechanisms to define materialized views and dimensions
- A refresh mechanism to ensure that all materialized views contain the latest data
- A query rewrite capability to transparently rewrite a query to use a materialized view
- An advisor to recommend which materialized views to create, retain, and drop, thereby aiding warehouse administrators.
Summary Management

• Summaries are created using materialized views and dimensions.
• Summary Advisor provides advice on materialized view creation, retention, and deletion.

Summary Management in Oracle8i

In Oracle8i and Oracle9i, materialized views and dimensions can be used to implement summaries.

Materialized views A materialized view is an instantiation of a SQL statement that stores both the definition of a view plus the rows resulting from the execution of the view. Like a view, it uses a query as the basis, but the query is executed at the time the view is created and the results are stored in a table.

When a query can be satisfied with data in a materialized view, the server transforms the query to reference the view rather than the base tables. The process of modifying a query to use the view is called a query rewrite.

Dimensions The database provides a schema object type, called a dimension. Dimensions provide for the creation of hierarchical relationships between columns in one or more tables for rollup purposes. If summaries are created with dimensions, query rewrite options are greatly enhanced.

Summary Advisor Summary Advisor is a facility that can be used by a database administrator to study the effectiveness of summaries based on how they are used. It compares performance benefit to storage costs, and advises on creating, retaining, or deleting summaries.
Materialized Views

- User view stores definitions, plus the rows resulting from execution of the view.
- May be indexed and partitioned
- Server transforms the query to reference the view, not the base tables.
- Joins are precalculated; therefore aggregations do not need to be reexecuted.

Materialized Views

A disadvantage of user views is that they must be resolved at execution time by going to the detailed underlying data in the base tables. Materialized views resolve this issue by physically storing the data that corresponds to the view’s defined query. When base table changes occur, these changes are automatically propagated through the materialized view. Generally, summary tables collapse detail across multiple dimensions and numerous summaries are required. Over time these summaries grow to be quite large and require rebuilding.

```
CREATE MATERIALIZED VIEW All_Sales_Rentals_MV
STORAGE (INITIAL 128K NEXT 128K PCTINCREASE 0)
TABLESPACE ALL_RENTAL_DATA06
NOLOGGING
PARALLEL
BUILD DEFERRED
REFRESH FAST ON DEMAND
ENABLE QUERY REWRITE ...
```
Why Use Materialized Views?

Use materialized views if:
• Retrieval time is high
• Aggregation time is high
• Compression ratio is low

Why Use Materialized Views?
• Retrieval time is high—sheer dimension size or a complex dimensional structure
• Aggregation time is high—with a simple fact table, dynamic aggregation of data to create summary information may be sufficient. However, a complex fact table with complicated calculations would suggest pre-aggregation to avoid complex SQL statements at query time.
• Compression ratio is low—the compression ratio for a dimension is determined by dividing the number of rows returned in a query by the number of rows actually scanned.
Guidelines for Using Materialized Views

• Define a single materialized view including all measures.
• Include \( \text{COUNT}(x) \) when using the aggregating measure \( \text{AVG}(x) \).

Guidelines for Using Materialized Views

• Define a single materialized view that includes all measures instead of defining multiple materialized views on the same tables with the same \groupby columns but with different measures.
• Include \( \text{COUNT}(x) \) when using the aggregating measure \( \text{AVG}(x) \) to support incremental refresh. Similarly, if \( \text{VARIANCE}(x) \) or \( \text{STDDEV}(x) \) is present, then always include \( \text{COUNT}(x) \) and \( \text{SUM}(x) \) to support incremental refresh.
Creating a Materialized View Summary

Materialized views can be defined with the same storage parameters as any other table and placed in the tablespace of your choice. You can also index and partition the materialized view to improve the performance of queries executed against them.

Example:
The example in the slide creates a summary and populates it with the result of the query. The performance and storage costs of maintaining the materialized view must be compared to the costs of re-executing the original query whenever it is needed.

```sql
CREATE MATERIALIZED VIEW sales_sumry
    TABLESPACE sum_data
    STORAGE(INITIAL 200K NEXT 200K
            PCTINCREASE 0)
    PARALLEL(...)
    BUILD IMMEDIATE
    REFRESH FAST
    ENABLE QUERY REWRITE
AS
    SELECT p.brand, c.city_name, t.month,
           SUM(s.amt) AS tot_sales
    . . .
    GROUP BY p.brand, c.city_name, t.month;
```
Query Rewrite

```sql
SELECT p.brand, c.city_name, t.month, SUM(s.amt)
FROM sales s, city c, timetab t, product p
WHERE s.city_code = c.city_code
AND s.state_code = c.state_code
AND s.sdate = t.sdate
AND s.prod_code = p.prod_code
GROUP BY p.brand, c.city_name, t.month
HAVING SUM(s.amt) > 5000000;
```

```
SELECT brand, city_name, month, tot_sales
FROM sales_sumry
WHERE tot_sales > 5000000;
```

Query Rewrite

Accessing a materialized view can be significantly faster than accessing the underlying base tables, so the cost-based optimizer will rewrite a query to access the view when the query allows it. Query rewrite is the primary benefit enabled by materialized views.

- The query rewrite activity is transparent to applications. In this respect, its use is similar to the use of an index.
- Users do not need explicit privileges on materialized views to use them. Queries executed by any user with privileges on the underlying tables can be rewritten to access the materialized view.
- A materialized view can be enabled or disabled. A materialized view that is enabled is available for query rewrites.

Example:
In the example, the optimizer is able to perform a query rewrite and use the summary created earlier to satisfy the query instead of the base SALES table. If the SALES table consists of several million rows and the materialized view contains a few thousand rows, the query will execute much faster.
Oracle Optimization Methods

- Rule-based approach
- Cost-based approach:
  - Recommended by Oracle for data marts
  - Requires table statistics

Oracle Optimization Methods
Beginning with Oracle8i, Oracle provides two approaches to execution plans for executing a SQL statement:

**Rule-based approach**—When using the rule-based approach, the optimizer looks at the different access paths available and uses a preconfigured set of rules to rank these access paths according to the execution speed of each path. Then it chooses the access path with the highest ranking. This approach does not take into account the size of the table or index, or other issues, such as how the data is distributed. Rule-based optimization is not usually used in data mart environments.

**Cost-based approach**—The optimizer considers available access paths and factors based on statistics for the tables or indexes accessed by the SQL statement to determine which path is the most efficient. These statistics are stored in data dictionary tables and quantify the data distribution and storage characteristics of tables and indexes. Using these statistics, the optimizer calculates how much I/O, CPU time, and memory are required to execute a SQL statement for a particular access path, and it assigns costs to each possible execution plan based on these calculations. The cost-based approach also considers hints or optimization suggestions placed in the SQL statement.

- Oracle recommends cost-based optimization for all data mart applications.
- Confirm that the `init.ora` `OPTIMIZER_MODE` parameter is set to `FIRST_ROWS`. If statistics are available for at least one table in the SQL statement, the optimizer uses the cost-based approach.
- Cost-based optimization requires statistics that are compiled using the `ANALYZE` command.
**Cost-Based Optimizer**

- Generates a set of potential execution plans
- Estimates the cost of each plan based on data size and storage capacity
- Compares the costs and selects the plan with the lowest cost

Cost-Based Optimizer

Whether provided within a tool or issued directly, SQL is used to access and manipulate the data in the database.

- SQL statements consist of reserved words. There are two kinds of SQL statements: data manipulation language (DML) and data definition language (DDL).
- Data can be accessed through different paths. Approaches to the data include full-table scans, index scans, and joins such as sort-merge joins, nested-loop joins, and hash joins.
- Oracle optimizes SQL statements by:
  - Validating the SQL statement
  - Validating user access
  - Choosing the optimization approach
  - Selecting an access path to obtain the data
  - Selecting join orders and then selecting an operation to perform the joins
  - Generating an explain plan, which shows the order of execution and types of joins that are used, and generates statistics
  - Optimizing based on previous processing
- The DBA can provide hints to suggest ways to optimize a SQL statement. The use of hints usually forces a rule-based approach, and does not allow the cost-based approach to occur.
Star Query

A star query, also called a star join, is a join between a fact table and a number of dimension tables. This is a typical style of query against a star schema which uses the cost-based optimizer. These table types are relational in structure, but use the less normalized model that is prevalent in warehouse database structures. This structure is different from the usual two-dimensional joins that have been considered, and another optimization technique is required to ensure the best performance of any star query. The technique starts by querying the dimension tables. The dimensional tables are those that form the query constraints (a `WHERE` clause). The dimension tables are never joined (even though the query may require data from many dimensions). The resulting data is retained, and the query then drives to the fact table—the largest table of all. A `SELECT` for a star join might appear as follows:

```sql
select *
from FACT, D1, D2, D3
where FACT.COL1 = D1.COL1
and FACT.COL2 = D2.COL1
and FACT.COL3 = D3.COL3
and . . . (further query restrictions)
```

Note that the dimension tables D1, D2, and D3 are not explicitly joined.
Star Query Optimization

Star query optimization depends on the star model and the cost-based optimizer. The optimizer has information about the indexes and join methods, as well as more detailed physical data on the rows and blocks and distribution of data.

If the optimizer assesses that the star join is the appropriate join method, then use the most basic approach. The dimensions (1) are used to create a Cartesian product (2), which is computed against smaller reference tables (3); the result is joined to the fact table (4) to produce the query result (5).

If the dimension tables are very large, then the star query may not be applicable because building the Cartesian product between those dimensions is resource intensive.

The star query can be made more efficient through the use of join indexes. A join index can be created across a single fact table and all its dimensions, eliminating the need to generate a Cartesian product of the dimensions during query execution. The shortcomings of join indexes relate to the relationship of the queried dimensions and the order of the dimension values in the index. You obtain the highest efficiency when the referenced dimensions are first in the join index.
Star Join Optimization Implementation

- Create bitmap indexes on the foreign keys in the fact table
- Create other indexes according to query requirements

Performance Solutions: Star Join Transformation

A star transformation technique is inherent in the Oracle server. Further improvements in performance over prior releases is evidenced in Oracle9i.

Implementation
- Create bitmap indexes on the foreign keys in the fact table.
- Create other indexes to support specific queries as needed.

Execution of a Query
1. Database retrieves appropriate rows from the fact table.
2. Using the bitmap indexes one dimensional constraint at a time, the database builds one combined bitmap.
3. The combined bitmap is used to retrieve the rows from the fact table.
4. These rows are joined to the dimension tables.

Additional Capabilities
- Support for outside-in queries, where constraints are first put on the facts, then joined to the dimensions
- Support for snowflake models
- Complete parallelization
- Support for bitmap join indexes

The STAR_TRANSFORMATION_ENABLED setting in the init.ora must be set to TRUE to affect the star query transformation.
Oracle9i Bitmap Join Index

- Improved performance for join queries
- Useful for star queries using materialized views

Oracle9i Bitmap Join Index

Bitmap join indexes are best understood by examining a simple example.

Example:
Suppose that a data warehouse contained a star schema with a fact table named SALES and a dimension table named CUSTOMER. Using bitmap join indexes, the following join index could be created on the SALES and CUSTOMER tables:

```sql
CREATE BITMAP INDEX cust_sales_bj_idx
ON Sales(Customer.state)
FROM Sales, Customer
WHERE Sales.cust_id = Customer.cust_id;
```

This join index could be used to evaluate the following query:

```sql
SELECT SUM(Sales.dollar_amount
FROM Sales, Customer
WHERE Sales.cust_id= Customer.cust_id
AND Customer.state = 'California';
```

The CUSTOMER table in this SELECT will not be accessed as the join index will be used instead. If the CUSTOMER table is a large dimension table, then the bitmap join indexes can by not requiring access to the CUSTOMER table.
Dimensions

- Data dictionary structures that define hierarchies based on existing columns
- Dimensions are optional, but highly recommended:
  - Enable additional query rewrites without the use of constraints
  - Help document hierarchies
  - Can be used by online analytical processing (OLAP) tools

Dimensions
Dimensions are purely metadata. They are data dictionary structures that define hierarchies based on columns in existing database tables. Although they are optional, they are highly recommended, for the following reasons:
- They enable additional rewrite possibilities without the use of constraints. Implementation of constraints may not be desirable in a data warehouse for performance reasons.
- They help document dimensions and hierarchies explicitly.
- They can be used by online analytical processing (OLAP) tools.
### Dimensions and Hierarchies

**Dimensions** Dimensions describe analytic business entities such as products, departments, and time in a hierarchical, categorized manner. A dimension can consist of one or more hierarchies. In the example shown, the time dimension consists of a calendar hierarchy.

**Hierarchies** A hierarchy consists of multiple levels. Each value at a lower level in the hierarchy is the child of one and only one parent at the next higher level. A hierarchy consists of a 1:n relationship between levels, with the parent level representing a level of aggregation of the child level. In the example, the calendar hierarchy consists of sales date, month, quarter, and year. The arrows indicate the direction of traversing a hierarchy to roll up data at one level to get aggregate information at the next level. For example, rolling up daily data yields monthly data, rolling up monthly data yields quarterly data, and so on.

**Level Keys and Attributes** A level key is used to identify one level in a hierarchy. The use of surrogate keys to identify hierarchical elements during the dimensional design phase further leverages the performance advantage provided by level keys. There may be additional attributes for a level, which can be determined given the level key. Attributes can be used as aliases for a level.

In the example, MONTH_KEY (defined as two digits) is the level key that identifies a month, and MONTH_DESC is an attribute that can be used as an alias for a month.
Dimension Example

Dimensions, and the hierarchical relationships established between dimensions, can be based on columns in a single table (or columns from several tables in the case of normalized or snowflake schemas).

In the example, the dimension **TIME_DIM** is based on the **TIME** table and has four levels:

- The highest level in the hierarchy consists of the **YEAR_KEY** column.
- The next level is derived from the **QUARTER_KEY** column.
- The third level has the **MONTH_KEY** column as the key and **MONTH_DESC** as an attribute.
- The lowest level is based on the **SALES_DATE** column.
Defining Dimensions and Hierarchies

The system privilege, `CREATE DIMENSION`, is required to create a dimension in one’s own schema based on tables that are within the same schema. Another new privilege, `CREATE ANY DIMENSION`, allows a user to create dimensions in any schema. In the example shown, the `TIME_DIM` dimension is based on the `TIME` table.

```sql
CREATE DIMENSION time_dim
  LEVEL sdate IS time.sales_date
  LEVEL mon IS time.month_key
  LEVEL qtr IS time.quarter_key
  LEVEL yr IS time.year_key
HIERARCHY calendar_rollup (  
  sdate CHILD OF  
  mon CHILD OF  
  qtr CHILD OF yr )
ATTRIBUTE mon
  DETERMINES month_desc;
```
Dimensions with Multiple Hierarchies

The previous example showed a single hierarchy within the TIME dimension, but it is possible to have multiple hierarchies. For example, the pair of hierarchies shown above can be created within a single dimension. The statement to do this is as follows:

```sql
CREATE DIMENSION time_dim
  LEVEL dt IS time.sales_date
  LEVEL wk IS time.week_key
  LEVEL mon IS time.month_key
  LEVEL qtr IS time.quarter_key
  LEVEL yr IS time.year_key
HIERARCHY cal (  
  dt CHILD OF  
  mon CHILD OF  
  qtr CHILD OF yr)  
HIERARCHY week (  
  dt CHILD OF  
  wk child of yr);
```
Rewrites Using Dimensions

The example in this slide shows a rewrite that is enabled by the `TIME_DIM` dimension. The relationship between month and year is inferred from the definition of the dimension and is used to roll up the sales summary data to obtain yearly sales.

Oracle9i supports the ANSI join syntax with complete support for one-sided and full outer joins.

```sql
SELECT v.year, s.brand, s.city_name, SUM(s.tot_sales)
FROM sales_summary s,
(SELECT distinct t.month, t.year FROM time t) v
WHERE s.month = v.month
GROUP BY v.year, s.brand, s.city_name;
```

```sql
SELECT t.year, p.brand, c.city_name, SUM(s.amt)
FROM sales s, city c, time t, product p
WHERE s.sales_date = t.sdate
AND s.city_name = c.city_name AND
  s.state_code = c.state_code
AND s.prod_code = p.prod_code
GROUP BY t.year, p.brand, c.city_name;
```

Rewrites Using Dimensions
Summary Advisor

The DBMS_OLAP package contains several procedures and functions to manage summaries in a data warehouse. The summary advisory functions within the package use two major sources of information: workload statistics and data dictionary information.

- Workload statistics—These can be collected either using Oracle Trace in the Enterprise Manager tuning pack or through third-party tools. An event, Materialized View Usage, is available with Oracle Trace to collect this information.
- Data dictionary—The data dictionary information used by the advisory functions includes summary and dimension data.

Information that can be obtained from the Summary Advisor includes:

- Summary usage—The number of times a rewrite was made to use a summary, the space used by a summary, a cost-benefit ratio for each summary, and so on
- Summary recommendations—Creation, retention, and dropping of summaries
- Space requirements based on queries for possible summaries
Summary

In this lesson, you should have learned how to:

• Discuss summary management and Oracle implementation of summaries
• Describe materialized views
• Explain the cost-based optimizer and query rewrite
• Understand star query optimization
• Identify Oracle dimensions
Practice 7-1 Overview

This practice covers examining materialized views.

Practice 7-1
Solutions for this exercise are located in Appendix B.
Practice 7-1 (continued)

Answer the following questions:

1. What is the fundamental difference between a summary and a materialized view?

2. The process of transforming a materialized view to reference the view rather than the base tables is called __________ __________.

3. a.) What are the two allowable Oracle execution plans for executing SQL statements?

b) Which approach is recommended by Oracle for data marts?

4. To enable Oracle’s use of bitmap indexes to perform star query transformations, the ______________ parameter should be set to TRUE in the init.ora file.

5. ____________ are data dictionary structures that define hierarchies based on columns in existing database tables and describe analytic business entities.
Analytical Capabilities

ILT Schedule:  
<table>
<thead>
<tr>
<th>Timing</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>45 minutes</td>
<td>Lecture</td>
</tr>
<tr>
<td>20 minutes</td>
<td>Practice</td>
</tr>
<tr>
<td>65 minutes</td>
<td>Total</td>
</tr>
</tbody>
</table>
Objectives

After completing this lesson, you should be able to do the following:

• Define Business Intelligence (BI)
• Understand the categories of BI tools and their use within the data warehouse
• Describe the benefits and concerns for deploying the data warehouse on the Web

Lesson Aim

This lesson discusses supporting end-user access. Specifically, this lesson introduces the concept of Business Intelligence and online analytical processing.
What Is Business Intelligence?

“Business Intelligence is the process of gathering meaningful information about the subject matter being researched.”

— Jonathan Wu

What Is Business Intelligence?

Jonathan Wu, co-founder and managing director of BASE Consulting Group, Inc., states “Business Intelligence is the process of gathering meaningful information about the subject matter being researched.” (DM Review Online, February 2000)

- From an information systems standpoint, BI provides users with online analytical processing or data analysis capabilities to predict trends, evaluate business questions, and so on.
- From a business analyst viewpoint, it is the process of gathering high-quality, meaningful information about a subject, which enables the analyst to draw conclusions.

BI applications empower the user by:

- Alleviating IT requests
- Providing the end user direct access to information through structured views eliminating the need for the individual user to understand the complexities of a relational database and SQL language
- Sharing accurate information across business units
- Improving the decision-making process

Data warehousing is the cornerstone for BI, providing vast analytic and trend capabilities based on actual data.
Evolution to BI

Over the past ten years technology has evolved from home-grown applications for EIS to packaged applications that allow the paradigm of data exploration, analysis, and mining to shift from IS to the individual end user.

- EIS applications were generally developed by the IS team and written in 3GL, 4GL, C++, or some other structured programming language. These were predefined, somewhat restrictive queries that were delivered in tabular or chart form. Generally, information provided was limited to sales totals, units produced, and so on.
- DSS applications were the first generation of packaged software that provided dynamically generated SQL enabling users to extract data from relational databases. This data was relevant to their business needs and focus. Primarily used by analysts, pertinent, exploratory questions could be asked, such as “What products are selling the best” or “Who are our customers?”
- BI, the next generation of DSS, provides the capability to format reports easily. Additionally, multiple sources and multiple subject matters can be used simultaneously to provide an accurate assessment of the business. What-if scenarios can be run against the multidimensional models and Web-enabled applications provide decentralization of the warehouse. BI is enabling a rapid evolution in Customer Relationship Management (CRM), supply chain analysis, sales force automation, technology forecasting, and so on.
Categories of Business Intelligence Tools

- Reporting tools and query tools for data access
- Data mining
- OLAP

Reporting tools and query tools—allow users to produce sophisticated reports based on warehouse data and generate intuitive ad hoc queries. Oracle Reports is a powerful, enterprise-scalable reporting solution used to create and deliver sophisticated, Web-based reports. Oracle Discoverer is an intuitive ad-hoc query, reporting, and analysis tools that empowers people to make better decisions. Oracle Reports is tightly integrated with Oracle Discoverer, allowing Discoverer end users to report the results of their slice and dice, ad-hoc queries.

Data mining tools—identify patterns and relationships in the data that are often useful for building models that aid decision making or predict behavior. Data mining uses technologies such as neural networks, rule induction, and clustering to discover relationships in data and make predictions that are hidden, not apparent, or too complex to be extracted using statistical techniques. Oracle Darwin is powerful, easy-to-use, enterprise data mining software that finds meaningful patterns hidden within corporate data. It enables 1:1 marketing by segmenting customers and predicting their behavior.

Online analytical reporting (OLAP) tools—provide a multidimensional view of data, allowing users to easily navigate through multiple dimensions (such as customer, organization, and time) and hierarchies within dimensions (such as year, quarter, and month). The different types of tools in this category are multidimensional OLAP (MOLAP), relational OLAP (ROLAP), and hybrid OLAP (HOLAP). Oracle Express provides sophisticated online analytical processing (OLAP) analysis through its advanced calculation engine and multidimensional data cache.
Query Tools

Query scheduling:
- Manages information usage
- Directs queries
- Executes queries
- Sets job queue priorities

Query monitoring:
- Tracks resource-intensive queries
- Detects unused queries
- Catches queries that use summary data inefficiently
- Catches queries that perform regular summary calculations at the time of query execution
- Detects illegal access

Business Intelligence Query Scheduling and Monitoring
The tools that you use must provide the flexibility to answer a user’s immediate and future needs. After the warehouse is operational, queries are submitted to the warehouse server. You must create a process that:
- Manages the use of information in the data warehouse
- Directs queries to the appropriate data source, using metadata
- Schedules the execution of a query
- Sets job queue priorities

You must check on warehouse query activity. The query management program or tool must:
- Track resource-intensive queries, which require analysis to identify why they are so resource intensive, followed by tuning to improve performance
- Detect queries that are never used and remove them. Ensure that users are advised of this kind of change.
- Catch queries that use summary data inefficiently; the summary strategy may need revision.
- Catch queries that perform regular summary calculations at the time of query execution.

You may decide to include another summary table in the data warehouse with the presummarized data to provide immediate access, which improves overall speed of access.
- Detect illegal access. A user may need access to currently denied data.
Multidimensional Query Techniques

These techniques are standard in modern query tools that present data in a multidimensional manner. The following defines some of the common multidimensional query techniques.

- **Slicing** means limiting the view of data to a selection of the data to a selection of the consultant, region, or cost center. An example of a slice of data can be a view of the data for a regional manager across all products and time periods.
- **Dicing** is slicing in multiple directions. You are making the selection along more than one dimension. In dicing, you can refine the selection by adding or removing data from the cube.
- **Drilling** is opening up a subset of data that corresponds with a particular value of a dimension. It is a term used to describe the action of moving down to further levels of data detail or up to higher levels of summary data.
- **Drilling down** is a mechanism that enables the user to examine the detail for a summary value. The user can examine where rackets were sold, to what companies, and how many items an individual purchased.
Multidimensional Query Techniques (continued)

- Drilling up is querying detail records and navigating up to higher level summary records.
- Drilling across is querying from one fact table to another in a single report.
- Pivoting data is changing the axes along which you orient your data. It also refers to the ability to change the organization of rows and columns in a tabular report. This enables the user to view the data along different dimensions without requerying the database itself.

OLAP has other associated query techniques, some of which are vendor dependent. For example top/bottom analysis selects the top or bottom ranges of data based on criteria to perform exception reporting.
Data Mining Tools

- Enable proactive business decisions
- Refer to a process rather than a technology
- Assist with BI decisions by:
  - Preventing customer attrition
  - Stimulating cross-selling
  - Stimulating acquisition of new customers
  - Detecting fraud
  - Providing accurate customer profiles

Data Mining Tools
- The purpose of data mining is to enable proactive business decisions. Data mining tools empower the user to search for patterns of information in data. Data mining is far less user-directed and relies upon specialized algorithms, such as fuzzy logic, neural networks, genetic algorithms, naïve bayes, and induction, that correlate information from the data warehouse and assist in trend analysis.
- Data mining also refers to a process rather than a technology, the goal of that process being to explore large amount of data to discover new trends, relationships, and categories in that data.
- Data mining is referred to as knowledge discovery, data surfing, or data harvesting.
- Potential uses of data mining are customer profiling, market segmentation, buying pattern affinities, database marketing, credit scoring, and risk analysis.
- Oracle9i Enterprise Edition embeds Oracle9i data mining functionality into the database.
OLAP Tools

Provide the ability to perform:
- Intensive data analysis by seamlessly drilling or pivoting
- Clickstream analysis if Web-enabled
- Complex calculations

OLAP Tools
OLAP tools provide the ability to perform advanced analysis beyond SQL capabilities. These tools are:
- Used for interactive analysis
- Fed by multiple systems (both internal and external)
- Usually contain a TIME dimension
- Frequently contain summarized data
OLAP Storage

According to Nigel Pendse, March 19, 2001, The OLAP Report, *OLAP Architectures*, “There is no single ideal way of storing or processing multidimensional data.” Generally, three configurations are available to store active OLAP data. These are:

- Relationship database—data is stored in an RDBMS, generally in the form of a star schema, snowflake, and so on as discussed in an earlier lesson.
- Multidimensional database—data is stored on a server. Generally aggregate and precalculated tables are stored on the server as well to enhance performance.
- Client-based files—data is stored generally in extract form on client machines. They can be distributed in advance or created on demand possibly through a Web portal.
OLAP Models

- Relational (ROLAP)
- Multidimensional (MOLAP)
- Hybrid (HOLAP)

OLAP Models
ROLAP—a two-dimensional table where queries are posed and run without the assistance of cubes, providing greater flexibility for drilling down, across, and pivoting results. Each row in the table holds data that pertains to a specific thing or a portion of that thing. Each column of the table contains data regarding an attribute.
MOLAP—a cube (the equivalent of a table in a relational database) stores a precalculated data set which contains all possible answers to a given range of questions. Each cube has several dimensions (equivalent to index fields in relational tables). The cube acts like an array in a conventional programming language. Logically, the space for the entire cube is pre-allocated. To find or insert data, you use the dimension values to calculate the position. Summarized tables are quite common in MOLAP configurations.
HOLAP—a hybrid of the MOLAP and ROLAP, combines the capabilities of both by utilizing both precalculated cubes and relations data sources.

Instructor Note
Oracle Express, Personal Express, and Discoverer provide these storage options.
Oracle9i OLAP

- OLAP-Ready relational database: All data and metadata are stored and managed from within the database.
- JAVA OLAP API: Means of expressing complex multidimensional queries
- Extended capabilities

Oracle9i OLAP
OLAP-Ready relational database—All data and metadata are stored and managed from within the database.
JAVA OLAP API—Is internet ready and provides multidimensional cursors, encapsulation, abstractions, and inheritance
Extended Capabilities—Supports a complete set of OLAP calculation functions and an extensive set of SQL capabilities for new types of analytic functions and aggregation. For analytic functions, Oracle9i builds upon the functions introduced in Oracle8i (ranking, moving-window aggregates, period-over-period comparisons, ratio-to-report, and statistical functions.) Oracle9i additionally provides:
- Inverse percentiles—Allows a query to find data that corresponds to a specified percentile value. For example, you can find the median value of a data set by querying PERCENTILE_DISC(0.5)
- Hypothetical rank and distributions—Allows a query to find what rank or percentile value a hypothetical data value would have if it were added to an existing data set.
- Histograms—Creates a width-balanced histogram of the data. For each row, this function returns a number representing the bucket number.
- FIRST/LAST aggregates—Allows comparisons between any element in a group and the first or last element. For example, this function can compute the difference between the current balance and the balance on the first day of the month.
Oracle9i OLAP SQL Aggregation

• Oracle8i introduced ROLLUP and CUBE
• Oracle9i introduces:
  – GROUPING SETS
  – CONCATENATED GROUPING SETS

Oracle9i OLAP SQL Aggregation

Oracle8i CUBE and ROLLUP extended the GROUP BY capability by enabling a single SQL query to calculate multiple levels of aggregation. Oracle9i extends this capability further with the introduction of GROUPING SETS, a feature that allows a SQL query to specify the exact levels of aggregation by the chosen subject. GROUPING SETS are specified by providing a column specification list. This example calculates aggregates over exactly 3 groupings (year, region, product), (year, product), and (region, product).

Example:
```
SELECT year, region, product, sum(sales)
FROM sales_table
GROUP BY GROUPING SETS ((year, region, product),
                          (year, product), and
                          (region, product));
```

Oracle9i also extends this capability further with CONCATENATED GROUPING SETS. These yield the cross-product of groupings from each grouping set.

Example:
```
...GROUP BY GROUPING SETS (month, year),
           GROUPING SETS (region, country)...
```
Choosing Servers

- **Relational**
- **Multidimensional**
- **Client multidimensional**

Choosing Servers
Relational—calculations are in SQL, inputs are generally to a multidimensional engine. Response time can be poor and scalability low. Consider RAM resident cache to improve query capability.
  - Warehouse stores atomic data
  - Application layer generates SQL for the three-dimensional view
  - Presentation layer provides the multidimensional view
Multidimensional—many calculations are stored in precalculated cubes. Response time is usually good, the database and access tools are typically optimized, and memory is not typically an issue. Extensive and comprehensive libraries of complex functions specifically for analysis and strong modeling and forecasting capabilities are available.
  - Application layer stores data in a multidimensional structure
  - Presentation layer provides the multidimensional view
Client multidimensional—with powerful desktops accessing Web application servers, thin clients are providing a popular alternative. An openness among systems provides the configuration with total flexibility. Different users can run different tools that access the data warehouse.
Web-Enabling the Data Warehouse

- Provides information and competitive advantage
- Shares ownership of data
- Collects information
- Makes deployment easier
- Supports an Intelligent Webhouse
- Improved ROI

Web-Enabling the Data Warehouse

A Web-enabled data warehouse is a means of providing access and query availability to your data warehouse by using a standard Web browser. It allows users to perform ad hoc queries against the database using their choice of Web browsers.

- Provides information and competitive advantage—Information demands on a data warehouse are pervasive throughout the organization and additional requirements may necessitate collection and dissemination of information with partners and customers.
- Shares ownership of data—Ensures that business units continue to advocate data warehouse proposals and share collaborative tasks more easily.
- Collects information—BI necessitates the need to collect as much information as possible from both external and internal sources. CRM and other critical initiatives flourish when a robust e-commerce customer intelligence framework is established to provide a 360° view.
- Makes deployment easier—Connection to a database through a BI application can be performed on any client system that has a Web browser.
- Supports an Intelligent Webhouse—Leverages clickstream analysis to support both reactive (e-traffic, e-marketing, and e-commerce) and proactive (recommendations and personalization) analysis.
- Improved return on investment (ROI)—Increasing the use of data warehouse spreads its value among more users and shortens the time for data warehouse ROI.
Challenges of Web-Enabling a Data Warehouse

• Security—The loss of data warehouse data to hostile parties can have extremely serious legal, financial, and competitive impacts on an organization. Make sure that your solution has strong encryption, authorization, and authentication services.
• Impact assessment on the infrastructure—You must assess the impact a Web-enabled warehouse will have on your IT organization and infrastructure. This includes changes in utilization patterns and the number of active clients; the need to learn new skills, such as integrating a warehouse database with a Web server; networks, servers, failover, and recovery procedures, development and testing tools, and training programmers as well as operator issues.
• Integration—If deployed on a stand-alone server, Assess how information provided to internal and external customers will be kept up-to-date and mission critical for purchasing or commitment and so on. You also must determine what information should be made available and to whom.
• Clickstream analysis—You must review the wealth of data that you are collecting and carefully craft dimensions to provide a 360° view of the customer. Date and Time, Visitor, Page Object, and Session Type dimensions are but a few that are necessary to focus the river of data and statistics flowing from the Internet. (Ralph Kimball, January 20, Volume 3, Number 2, The Special Dimensions of the Clickstream)
Security Issues in Deploying the Web-Enabled Data Warehouse

- Subject area sponsors:
  - Review and authorize access requests
  - Identify enhancements
- Transparent security or unobtrusive security
- Should be easy to implement, maintain, and manage
- Authentication

Security
- View-based security techniques, with a permissions table that identifies users’ clearance codes. The codes themselves match to clearance codes held with the data in the warehouse.
- Caching techniques that allow only queries available to users of a certain code to actually access the cached data.
- Password abstraction, which allows you to specify a password for access that is then converted behind the scenes, where access to the database is then made available.
- All security measures should be well documented and easy to maintain. Develop a security plan initially, and then maintain and adhere to this plan.

Simple Security
- Roles—limit use by departments
- Restrict access by password
- Audit sessions using the AUDIT_TRAIL parameter
- Allow read-only access to data warehouse and limited update access to the staging area

Advanced Security
- Fine grain row-level security for INSERT, UPDATE, SELECT, or DELETE
- Net8 Advanced Security can be enabled (authentication by the network, by the middleware, and by the architecture)
- Consider an LDAP server for authentication
Summary

In this lesson, you should have learned how to do the following:

• Define Business Intelligence (BI)
• Understand the categories of BI tools and their use within the data warehouse
• Describe the benefits and concerns for deploying the data warehouse on the Web
Course Summary

In this course, you should have learned how to do the following:

• Define the process of designing a data warehouse database model
• Describe the role of metadata in data warehouse design and strategies used to define and maintain metadata
• Explain the central concepts of dimensional data models
• Analyze and transform business requirements into a business model
Course Summary

• Use entity relationship diagrams to transform the business model into a dimensional model
• Transform the dimensional model into a physical data design
• Evaluate summaries and understand the value of materialized views
• Understand multidimensional query concepts
Practice 8-1 Overview

This practice covers the following topics:
• Understanding Business Intelligence
• Identifying Web deployment considerations

Practice 8-1
Solutions for this exercise are located in Appendix B.
Practice 8-1 (continued)
Answer the following questions:

1. In your own words, explain the concept “business intelligence”.
   _____________________________________________
   _____________________________________________
   _____________________________________________
   _____________________________________________
   _____________________________________________
   _____________________________________________

2. ___________ ___________ tools are used to identify patterns and relationships in data.

3. Name five techniques for querying data in an OLAP environment.
   1. _____________________________________________
   2. _____________________________________________
   3. _____________________________________________
   4. _____________________________________________
   5. _____________________________________________

4. What are three storage options for OLAP models?
   1. _____________________________________________
   2. _____________________________________________
   3. _____________________________________________

5. Why might you Web-enable a data warehouse?
   _____________________________________________
   _____________________________________________
   _____________________________________________
   _____________________________________________
   _____________________________________________

6. Name four challenges to Web-enabling a data warehouse.
   1. _____________________________________________
   2. _____________________________________________
   3. _____________________________________________
   4. _____________________________________________
Case Study

Case Study for the Merger of Vintners Ltd. and eVine&Dine.com
**Business Case**

The business case is one that combines two worlds of business, namely a merger between a dot com wine vendor and a traditional wine importer. To realize the combined potential of both organizations and glean insight into potential revenue sources, management has determined that a BI&W solution must be achieved. The table below shows the two partners in a summarized form:

<table>
<thead>
<tr>
<th>Overview</th>
<th>eVine&amp;Dine.com</th>
<th>Vintners Ltd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• New economy</td>
<td>• Old economy</td>
</tr>
<tr>
<td></td>
<td>• Internet/Web store front</td>
<td>• Brick-and-mortar store front</td>
</tr>
<tr>
<td></td>
<td>• B2B (modeled after Amazon.com)</td>
<td>• Face-to-face customers</td>
</tr>
<tr>
<td>History</td>
<td>• Garage startup</td>
<td>• Traditional individual family owned businesses</td>
</tr>
<tr>
<td></td>
<td>• Venture capital (VC) funding — launched the business to the internet</td>
<td>• Built a consortium of families to share different labels and increase variety</td>
</tr>
<tr>
<td>Business Drivers</td>
<td>• Expand market share</td>
<td>• Expand market share (that is, get their labels out—worldwide exposure)</td>
</tr>
<tr>
<td></td>
<td>• Worldwide presence</td>
<td>• Niche player</td>
</tr>
<tr>
<td></td>
<td>• 24x7</td>
<td>• 8x5 (8 a.m. to 5 p.m. — want to shorten hours to focus on distribution)</td>
</tr>
<tr>
<td></td>
<td>• Want to reach out to all levels of customers (current mid - low)</td>
<td>• Want to expand their customer base (current mid – high)</td>
</tr>
<tr>
<td>Current Production</td>
<td>• Specialize in Californian wines</td>
<td>• Specialize in European wines</td>
</tr>
<tr>
<td>Offerings, Accessories,</td>
<td>• Gourmet foods (sauce, dressings, oil, cheese, pasta, bread {sourdough}),</td>
<td>• Limited accessories, cheeses</td>
</tr>
<tr>
<td>&amp; Other</td>
<td>• Buckets, corks, glass wares</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Posters, labels, Cookbooks, cooking utensils, and so on</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Wine clubs (memberships)</td>
<td></td>
</tr>
</tbody>
</table>
**eVine&Dine.com background**
eVine&Dine.com, a Web storefront that specializes in Californian wines and wine accessories, opened its window on the internet to sell its products in October, 1999. Because the growth rate in the highly competitive wine market is a fraction of what it was a decade ago, eVine&Dine.com will need more than just an attractive Web site to be successful online.

Fulfillment of orders has become difficult as eVine does not have an inventory base in a warehouse. Analysis of the types of products that are selling and their customer base is impossible with the Web traffic they are experiencing given the IS tools they currently possess. Their strategy is to identify and retain repeat customers, especially those who buy high margin items such as Tiffany wine glasses, lead crystal decanters, and other wine accessories.

**Vintners Ltd. background**
Vintners Ltd. is a European company importing and distributing wines throughout the United Kingdom and the European mainland. Vintners is an old world store (brick and mortar) set up with its main office in the United Kingdom. Their expertise focuses on an in-depth knowledge of the European market and trends. The recent trend to purchase foreign wines, especially those from California and Australia, has been noted.

With the direct sales channel slowing considerably for wine products, the decision was made to add an internet channel for sales to stimulate profit.

**Integration**
Vintners Ltd. name will be retained for the merged companies. The goals for the newly formed company are:

- Consolidate sales and customer bases
- Re-establish themselves as the premier vendor for wine and upscale associated products in the global community
- Expand distribution channels
- Expand market share throughout the global community
- Integrate clickstream data with other sources of data, such as data from the order fulfillment, CRM, and OLTP customer support and purchasing systems
- Integrate both the online and offline business data for comparative performance analysis
Case Study Characteristics

As the Chief Information Officer at Vintners Ltd., Sandra Chambers needs to focus on:

- Integrating clickstream data with other sources of data such as the order fulfillment, customer relationship management (CRM) system, customer support, and purchasing systems.
- Integrating both the online and offline business data for comparative performance analysis.
Case Study Characteristics

As the VP of Sales, Gary Smith wants to find out:

Customer order information
- What products do we sell to what customers?
- What are the trends we see in this buying behavior?
- What are the high margin products and customers?
- What are the high volume products and customers?

Purchasing information
- What products do we buy from which suppliers?
- How many suppliers do we use for each product?
- Do we buy on fixed rates?
As the Data Warehouse Database Administrator, Peter Lee is the technical expert in charge of making sure the data warehouse is running smoothly. Because of the merger between eVine&Dine and Vintners Ltd., Peter has to:

- Consolidate information from multiple sources
- Work with management to look at business intelligence in order to have a tighter grip on Vintners Ltd.’s customer and product base
Information Requirements

The Sales and Marketing organization has been struggling with a lack of timely information about what is selling, who is buying, and how they are buying. In a meeting with CIO Sandra Chambers, Sales VP Gary Smith stated, “By the time I get the information, it is no longer useful! I am only able to get information at the end of each month, and it doesn’t have the details I need to do my job.” When Gary asked Sandra to be more specific about what she needed, Sandra identified the following information requirements:

- Provide sales data for specific customers
- Provide sales detail for mail order, phone, and Web sales on a weekly and monthly basis and compare them to past time periods—when, how, and what is being sold by each channel
- Provide margin information on products to understand dollar contribution for each sale

Management Team Recommendations

After numerous interviews with the executive management team, they have identified specific business processes as primary candidates for the initial warehouse increment—customer orders and sales:

- Build a data warehouse that will handle all facets of the business, beginning incrementally with customer and products
- Consolidate sales and customer bases allowing management to analyze sales trends and profile customer and customer product requirements
The Entity Relationship Diagrams (ERD) on this slide and the next were developed from reviewing all sources available in the current IS environments for both organizations. These sources included, but were not limited to, RDBMS, VSAM files, flat files, OLTP, and so on.

Please note that this class will not try to develop a complete data warehouse design for these ERDs because of classroom time constraints.
## Legacy System Attribute Tables

### Customer Table:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Format</th>
<th>Length (integer,decimal)</th>
<th>Key?</th>
<th>O(ptional) or M(andatory)</th>
<th>G(enerated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cust_ID</td>
<td>Number</td>
<td>6,0</td>
<td>PK</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>Cust_First_Name</td>
<td>Varchar2</td>
<td>20 - (not null)</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Last_Name</td>
<td>Varchar2</td>
<td>20 - (not null)</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Street_Address</td>
<td>Varchar2</td>
<td>40</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_City</td>
<td>Varchar2</td>
<td>30</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_State_Region</td>
<td>Varchar2</td>
<td>10</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Country_Name</td>
<td>Varchar2</td>
<td>40</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Email</td>
<td>Varchar2</td>
<td>25</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Phone_Number</td>
<td>Varchar2</td>
<td>20</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Credit_Rating</td>
<td>Char</td>
<td>4</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Credit_Limit</td>
<td>Number</td>
<td>10,2</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Currency_Code</td>
<td>Char</td>
<td>3</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Promotion_Category</td>
<td>Char</td>
<td>2</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cust_Last_Purchase_Date</td>
<td>Date</td>
<td>(not null)</td>
<td>M</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Legacy System Attribute Tables

#### Order Table:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Format</th>
<th>Length (integer,decimal)</th>
<th>Key?</th>
<th>O(ptional) or M(andatory) G(enered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order_ID</td>
<td>Number</td>
<td>12,0</td>
<td>PK</td>
<td>G</td>
</tr>
<tr>
<td>Order_Date</td>
<td>Date</td>
<td>(not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Order_Promised_Date</td>
<td>Date</td>
<td>(not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Order_Shipped_Date</td>
<td>Date</td>
<td>(not null)</td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>Order_Status</td>
<td>Number</td>
<td>2,0 - (not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Order_Total</td>
<td>Number</td>
<td>8,2</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Order_Discount</td>
<td>Number</td>
<td>4,2</td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>Order_Promotion_Category</td>
<td>Char</td>
<td>2</td>
<td></td>
<td>O</td>
</tr>
</tbody>
</table>

#### Order Item Table:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Format</th>
<th>Length (integer,decimal)</th>
<th>Key?</th>
<th>O(ptional) or M(mandatory) G(enerated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order_Item_ID</td>
<td>Number</td>
<td>12,0</td>
<td>PK</td>
<td>G</td>
</tr>
<tr>
<td>Order_Item_Category</td>
<td>Number</td>
<td>8,0 - (not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Order_Item_Quantity_Ord</td>
<td>Number</td>
<td>8,0 - (not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Order_Item_Quantity_Shp</td>
<td>Number</td>
<td>8,0 - (not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Order_Item_Status</td>
<td>Number</td>
<td>2,0 - (not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Order_Item_Price</td>
<td>Number</td>
<td>8,2</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Order_Item_Discount</td>
<td>Number</td>
<td>4,2</td>
<td></td>
<td>O</td>
</tr>
</tbody>
</table>
## Legacy System Attribute Tables

### Product Table:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Format</th>
<th>Length (integer, decimal)</th>
<th>Key?</th>
<th>Optional or Mandatory Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prod_ID</td>
<td>Number</td>
<td>12,0</td>
<td>PK</td>
<td>G</td>
</tr>
<tr>
<td>Prod_Name</td>
<td>Varchar2</td>
<td>50 - (not null)</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Prod_Description</td>
<td>Varchar2</td>
<td>50 - (not null)</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Prod_Qty_Onhand</td>
<td>Number</td>
<td>8,0</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Prod_Shipment_Due_Date</td>
<td>Date</td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Warehouse_ID</td>
<td>Varchar2</td>
<td>4 - (not null)</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Prod_Category</td>
<td>Char</td>
<td>2</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Promotion_Category</td>
<td>Char</td>
<td>2</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Status</td>
<td>Varchar2</td>
<td>2 - (not null)</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Weight_Class</td>
<td>Number</td>
<td>1,0</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Unit_Of_Measure</td>
<td>Varchar2</td>
<td>20</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Package_Size</td>
<td>Varchar2</td>
<td>30</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_List_Price</td>
<td>Number</td>
<td>8,2</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Cost</td>
<td>Number</td>
<td>8,2</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Min_Price</td>
<td>Number</td>
<td>8,2</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Channel_1</td>
<td>Varchar2</td>
<td>30</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Channel_2</td>
<td>Varchar2</td>
<td>30</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Catalog_URL</td>
<td>Varchar2</td>
<td>50</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Supplier_ID</td>
<td>Number</td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Supplier_Name</td>
<td>Varchar2</td>
<td>50</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Wine_Type</td>
<td>Varchar2</td>
<td>50</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Wine_Size</td>
<td>Varchar2</td>
<td>20</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Wine_Rating_Number</td>
<td>Number</td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Wine_Rating_Date</td>
<td>Date</td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Appellation</td>
<td>Varchar2</td>
<td>30</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Prod_Winery_Name</td>
<td>Varchar2</td>
<td>30</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>
## Legacy System Attribute Tables

### Warehouse Table:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Format</th>
<th>Length (integer,decimal)</th>
<th>Key?</th>
<th>O(optional) or M(andatory) G(enerated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whse_ID</td>
<td>Number</td>
<td>4</td>
<td>PK</td>
<td>G</td>
</tr>
<tr>
<td>Whse_Name</td>
<td>Varchar2</td>
<td>50 - (not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Whse_Country_Name</td>
<td>Varchar2</td>
<td>30 - (not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Whse_Mgr_Name</td>
<td>Varchar2</td>
<td>40 - (not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Whse_Phone_Number</td>
<td>Varchar2</td>
<td>20 - (not null)</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Whse_Storefront_Mgr_Name</td>
<td>Varchar2</td>
<td>40 - (not null)</td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>Whse_Storefront_Name</td>
<td>Varchar2</td>
<td>50 - (not null)</td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>Whse_Storefront_Phone_Number</td>
<td>Varchar2</td>
<td>20 - (not null)</td>
<td></td>
<td>O</td>
</tr>
</tbody>
</table>

### Sales Table:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Format</th>
<th>Length (integer,decimal)</th>
<th>Key?</th>
<th>O(optional) or M(andatory) G(enerated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales_ID</td>
<td>Number</td>
<td>10</td>
<td>PK</td>
<td>G</td>
</tr>
<tr>
<td>Amount_Sold</td>
<td>Number</td>
<td>14,2</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Quantity_Sold</td>
<td>Number</td>
<td>12,0</td>
<td></td>
<td>M</td>
</tr>
</tbody>
</table>
Practice 1-1

1. What is perceived to be the single biggest cause for data warehouse database design failure?
   - The primary reason for data warehouse database design failure centers on encompassing too many business processes. When the number of business processes are not limited, the scope of the project is much too large and the implementation time much too long.

2. What are the two basic data forms of a data warehouse star schema?
   - Fact data
   - Dimension data

3. What would Bill Inmon consider to be the four most important characteristics of a data warehouse?
   - Subject oriented
   - Integrated
   - Nonvolatile
   - Time-variant

4. What are the chief differences between an OLTP database and a data warehouse?
   - Data warehouses are designed for optimal query access not data manipulation.
   - Data warehouses are generally developed in an incremental fashion rather than long drawn out IS operational development cycles.
   - Data warehouses are nonvolatile, that is, they change very slowly not on a daily basis.
   - Data warehouses provide a plethora of historical data not current transactional data.

5. What are four components of a decision support system?
   - The two storage components are operational data stores (ODS) and data warehouses (DW).
   - The two analytic components are data mining (DM) and online analytical processing (OLAP).

6. What is the Common Warehouse Model and why is it important?
   - CWM is Oracle’s standard for data warehousing metadata enabling tighter integration among Oracle products. CWM concentrates on the needs of data warehousing and decision support and is able to provide a rich model for warehousing that simplifies the development of warehouse tools and applications. The CWMI standard will enable the interchange of warehouse metadata among data management and analysis tools, and among warehouse metadata repositories.
Practice 1-1 (continued)

7. Define scalability.
   – Scalability is the ability of a data warehouse to grow over a long period of time.

8. A **data mart** is a simpler form of a data warehouse and is designed for a single line of business or functional area.

9. a) What are the seven phases in the spiral approach to data warehouse design?
   – Strategy
   – Definition
   – Analysis
   – Design
   – Build & Document
   – Populate & Test
   – Discovery & Evolution

   b) Why is this approach preferable to the waterfall approach?
   – The waterfall methodology is not iterative by design. If business needs change, the design cannot. Requirements have been defined, documented, and frozen. This is unrealistic in today’s rapidly evolving marketplace.
Practice 2-1

1. What are the three tasks involved in business modeling?
   – Strategic analysis
   – Business modeling
   – Metadata documentation

2. Why is strategic analysis essential?
   – Strategic analysis allows you to identify achievable deliverables and identify the business processes that are most important from the standpoint of decision making.

3. Many resources are available within your organization to aid your analysis and provide necessary information. Name at least four.
   – Company annual report
   – Marketing literature
   – Documentation from earlier failed data warehouse attempts
   – Business plans

4. Why is an ODS not a data warehouse?
   – An ODS violates one of the four principle aspects of a data warehouse, that is., nonvolatile.

5. Business **measures** are the success metrics of a business process, while business **dimensions** provide the metadata definitions or analytic parameters for the processes.

6. **Granularity** is the level of detail or the atomic level of the data warehouse.

7. An important activity while identifying business definitions and rules is **documenting** metadata which provides a clear, coherent understanding of the metadata usage and definition.

8. **Dimensional** modeling is a top down approach while **entity relationship** modeling is a bottom up approach to design.

9. What four entities (organizational units) are most important for the development of the customer sales process model?
   – Customer
   – Order
   – Warehouse
   – Products

Data Warehouse Database Design B-4
Practice 2-1 (continued)


Data Warehouse Database Design B-5
Practice 3-1

1. What are the four essential data types found in a data warehouse?
   - Dimension
   - Fact (including derived fact data)
   - Summary
   - Metadata

2. What are the two table types that comprise a star dimensional model?
   - Dimension
   - Fact

3. Fact tables are joined to dimension tables through foreign keys that reference primary keys within the dimension.

4. Explain the difference between additive, semi-additive, and non-additive fact tables.
   - Additive data in a fact table can be added across all dimensions; semi-additive can be added across some, but not all; non-additive cannot be added between records.

5. What are the three ways to handle slowly changing dimensions?
   - Overwrite the existing dimensional information.
   - Add a new record each time the dimension changes.
   - Add critical fields such as a time attribute to the dimension table to preserve the changed information, provide historical documentation, and maintain the ability to query on all changed records.

6. Explain why surrogate key use is important in a data warehouse.
   - Surrogate keys are generated providing uniqueness to each record. These keys remain independent from the original source data natural keys that could cause duplicate key violations. Additionally, surrogate keys tend to be smaller in size than natural keys reducing response time for queries.
Practice 3-1 (continued)

7. What is a bracketed dimension?
   – A bracketed dimension is a grouping of values for attributes that provides a range for enhanced performance of queries.

8. **Granularity** determines the lowest level of detail in a data warehouse and is considered to be the single most important design consideration.

9. A **star model** contains a fact table with one level of related dimensions.
Practice 3-2
1. Customer Sales Star Model
Practice 3-3

1. What attribute should be added to each table to ensure uniqueness in the database?
   - Surrogate key
Practice 3-3 (continued)

2. Customer Sales Star Model

CUSTOMER
* CUST_ID
* CUST_FIRST_NAME
* CUST_LAST_NAME
  o CUST_STREET_ADDRESS
  o CUST_CITY
  o CUST_STATE_REGION
  o CUST_COUNTRY_NAME
  o CUST_EMAIL
  o CUST_PHONE_NUMBER
* CUST_CREDIT_RATING
* CUST_CREDIT_LIMIT
* CUST_CURRENCY_CODE
  o CUST_PROMOTION_CATEGORY
* CUST_LAST_PURCHASE_DATE

ORDER
# ORDER_ID
* ORDER_DATE
* ORDER_PROMISED_DATE
* ORDER_SHIPPED_DATE
* ORDER_STATUS
* ORDER_TOTAL
  o ORDER_DISCOUNT
  o ORDER_PROMOTION_CATEGORY

SALES
# SALES_ID
* AMOUNT_SOLD
* QUANTITY_SOLD

WAREHOUSE
# WHSE_ID
* WHSE_NAME
* WHSE_COUNTRY_NAME
* WHSE_MGR_NAME
* WHSE_PHONE_NUMBER
  o WHSE_STOREFRONT_MGR_NAME
  o WHSE_STOREFRONT_NAME
  o WHSE_STOREFRONT_PHONE_NUMBER

houses

OWNER OF

ORDER

owned by

CUSTOMER

houses by

SALES

include

ORDER

PRODUCT

contain

SALES

included in

PRODUCT

contain in

PRODUCT

Data Warehouse Database Design B-10
Practice 4-1

1. Where does the physical model reside?
   - The physical model resides on the relational database (RDBMS).

2. **SMP** is the oldest hardware architecture useful for data warehousing.

3. **Cluster** and **MPP** provide a greater level of scalability over SMP for data warehousing.

4. **NUMA**, although a relatively new entrant into the data warehousing arena, provides both scalability and manageability.

5. 3-tier (or more) architecture is preferable for data warehousing because the first tier is devoted to operational processing, the second tier to department-level queries and analysis, and the third tier to desktop presentation.
   
   **Note:** If a web-based server is employed, 4-tier or greater should be used.

6. A **parallel processor** divides a large task into smaller tasks that can execute concurrently in one or more nodes.

7. **Parallel query** splits SQL code among several processors to minimize the impact of query analysis on CPU resource and disk I/O.

8. Why is a parallel database of benefit to the data warehouse?
   - A parallel database provides increased speed and improved scalability by running multiple instances against a single physical database using multiple processors.
Practice 4-2

1. What column in the fact table of a star schema represents a dimension?
   – Foreign Key

2. Because the sales table is the heart of the star schema, it must be associated with the other tables. To do this, you must add three foreign keys to the sales table. Which three tables should participate?
   – Products
   – Orders
   – Customers
Practice 4-2 (continued)

3.
Practice 4-3

<table>
<thead>
<tr>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales_Sales_ID</td>
</tr>
<tr>
<td>SQL_DATE</td>
</tr>
<tr>
<td>Julian_Day</td>
</tr>
<tr>
<td>Day_of_Year_NBR</td>
</tr>
<tr>
<td>Day_of_Month_NBR</td>
</tr>
<tr>
<td>Day_of_Week_NBR</td>
</tr>
<tr>
<td>Day_of_Week_Name</td>
</tr>
<tr>
<td>Month_NBR</td>
</tr>
<tr>
<td>Year_Month_NBR</td>
</tr>
<tr>
<td>Month_Name</td>
</tr>
<tr>
<td>Year_Quarter_NBR</td>
</tr>
<tr>
<td>Quarter_NBR</td>
</tr>
<tr>
<td>Year_NBR</td>
</tr>
</tbody>
</table>
Practice 4-3 (continued)

1.
Practice 5-1

1. When should B*tree indexes be employed?
   - B*tree indexes should be employed when the anticipated number of distinct values in a column is small.

2. What parameter do you set in an Oracle database to avoid fragmentation?
   - PCTINCREASE

3. Why is a parallel database of benefit to the data warehouse?
   - A parallel database provides increased speed and improved scalability by running multiple instances against a single physical database using multiple processors.

4. How does multiple parallel query work?
   - The server parallelizes individual queries into units of work that can be processed simultaneously. These units of work are single SQL statements.

5. **Partitions** help solve decision support performance problems and also resolve the problem of reloading or processing large volumes of updates against existing fact tables.
Practice 6-1

1. The **remote** staging model is considered to be the best because the data staging area is in its own environment, avoiding negative impact on the warehouse.

2. The process of **extraction** takes selected data from source fields that pertain to the subject area maintained by the data warehouse.

3. **Quality data** is the key to a successful warehouse because data anomalies can lead to inaccurate information and bad business decisions.

4. **Transformation** involves the elimination of anomalies, standardization of formatting, assignment of data types, definition of measure, and determination of encoded structures.

5. Although transformation may be effected on the operational platform or at the warehouse server level, a separate **staging** area is considered to be the preferable approach because it does not affect either the operational or warehouse environment.

6. The **transportation** process moves data from source data stores or an intermediate staging area and loads it into the target warehouse database in the target system server.

7. Oracle8i provides a very fast, effective method to move data from one database to another. This methodology is called **transportable tablespace**.
Practice 7-1

1. What is the fundamental difference between a summary and a materialized view?
   – A summary view needs to be resolved at execution time by going to the detailed underlying data in the base tables. A materialized view resolves this issue by physically storing the data that corresponds to the view’s defined query.

2. The process of transforming a materialized view to reference the view rather than the base tables is called **query rewrite**.

3. a.) What are the two allowable Oracle execution plans for executing SQL statements?
   – Rule-based approach
   – Cost-based approach

   b) Which approach is recommended by Oracle for data marts?
   – Cost-based is recommended for data marts

4. To enable Oracle’s use of bitmap indexes to perform star query transformations, the **STAR-TRANSFORMATION_ENABLED** parameter should be set to **TRUE** in the init.ora file.

5. **Dimensions** are data dictionary structures that define hierarchies based on columns in existing database tables and describe analytic business entities.
Practice 8-1
1. In your own words, explain the concept “business intelligence.”
   - Business intelligence is a set of concepts, methods and processes used to improve business decision making. It empowers companies to direct business process improvement and monitor time, cost, quality, and control. The process is cross functional, in line with current management thinking, and not presented in IT terms.
2. **Data mining** tools are used to identify patterns and relationships in data.
3. Name five techniques for querying data in an OLAP environment.
   - Drilling up
   - Drilling down
   - Drilling across
   - Slicing
   - Dicing
4. What are three storage options for OLAP models?
   - MOLAP
   - ROLAP
   - HOLAP
5. Why might you Web-enable a data warehouse?
   - Provide information and competitive advantage
   - Share ownership of data
   - Collect information to provide a 360° view
   - Make deployment easier as connection to a database through a BI application can be performed on any client system that has a Web browser
   - Support an Intelligent Webhouse to leverage clickstream analysis
   - Improve return on investment (ROI)
6. Name four challenges to Web-enabling a data warehouse.
   - Security
   - Impact assessment on the infrastructure
   - Integration
   - Clickstream analysis
Additive Measurements in a fact table that can be added across all the dimensions.

Aggregated data Pre-calculated and pre-stored summary data that is held in tables in a data warehouse in a summarized form. Aggregations are used for two primary reasons: to save storage space and to improve the performance of business intelligence tools such as queries.

Aggregated facts See Summary tables
Aggregation See Aggregated data

Atomic Data Data elements that represent the lowest level of detail. The data value cannot be further reduced. For example, in a daily sales report, the individual items sold would be atomic data, whereas rollups such as invoice and summary totals from invoices are aggregate data.

Attribute Any detail that serves to qualify, identify, classify, quantify, or express the state of an entity.

B

B*Tree Index A specialized form of index that processes a structure hierarchically to find the values of interest.

Bitmap Index A specialized form of index where the structure contains a large number of rows but only a small number of distinct values.

Bracketed Dimensions Grouping a dimension to lower the cardinality and make the data more meaningful, for example, grouping income by tens of thousands or putting mileage into ranges.

Bus A cable or circuit used to transfer data or electrical signals among devices.

C

Cardinality The number of rows in a table.

Cell A single point in a cube.

Cleansing The process of transforming the operational and external source data into a defined and standardized format using packaged software applications or programs, prior to moving that data into the warehouse. Also known as data scrubbing. See Data cleansing and Data scrubbing.

Clickstream Data Data regarding Web browsing. Web servers capture a large amount of data in the process of receiving requests for Web pages. This data includes page served, time, source of the request, type of browser making the request, and so on.

Client A software application that is used to extract or download an application, data, or service from a host system.
Client-Server  A distributed technology approach where the processing is divided by function. The server performs shared functions—managing communications, providing database services, and so on. The client performs individual user functions—providing customized interfaces, performing screen-to-screen navigation, offering help functions, and so on.

Client-Server Architecture  A networked environment where a smaller system such as a PC interacts with a larger, faster system. This allows the processing to be performed on the larger system which frees the user's PC. The larger system is able to connect and disconnect from the clients in order to process the data more efficiently.

Client-Server Processing  A form of cooperative processing in which the end-user interaction is through a programmable workstation (desktop) that must execute a part of the application logic over and above display formatting and terminal emulation.

Cluster  A means of sorting and storing related data from different tables in the database on cluster keys. Advantageous in an environment where related data is commonly queried together.

Column  A means of implementing an item of data within a table.

CWM (Common Warehouse Method)  An integration approach for data warehousing, incorporating both technical and business metadata into a single model that concentrates on the needs of data warehousing and decision support.

Composite Key  A composite key consists of a number of columns. In the case of a concatenated primary key, combinations of values between the columns must be unique. These keys are sometimes referred to as concatenated or segmented keys, also known as compound or concatenated key.

Conformed Dimension  A dimension that is used in more than one cube and shares the same grain and detail. The use of conformed dimensions and shared measures is the primary way a set of data marts can be united into one consolidated data warehouse.

Constellation Model  A warehouse model that comprises a collection of star models.

Constraint  1) The part of the WHERE clause in a SQL SELECT statement that identifies the column or field value that qualifies the query. 2) Any external, management, or other factor that restricts a business or systems development in terms of resources, availability, dependencies, timescales and so on. Also known as business rules.

Cost-based Optimizer  A statistical mechanism that analyzes where and how to retrieve data from the Oracle7, Oracle8, and Oracle8i servers to ensure fast access to data.

Cube (Also Known As Multidimensional Cube or Hyper-Cube)  The fundamental structure for data in a multidimensional (OLAP) system. See Hyper-Cube.
**Data Acquisition**  The process of extracting, transforming, and transporting data from the source systems and external data sources to the data warehouse database objects. The term is synonymous with ETT, and is widely used within the Data Warehouse Method.

**Data Aggregation**  The process of redefining data into a summarization based on rules or criteria.

**Data Cleansing**  Removing errors and inconsistencies from data that is being imported into a data warehouse. See Data scrubbing.

**Data Integrity**  The quality of the data residing in database objects. Constraints on database tables enforce integrity rules.

**Data Mart**  A database that has the same characteristics as a data warehouse, but is usually smaller and focused on the data for one division or one workgroup within an enterprise.

**Data Mining**  A technique of finding hidden patterns and relationships in the data.

**Data Model**  A representation of the specific information requirements of a business area. See entity relationship diagram (ERD).

**Data Scrubbing**  Removing errors and inconsistencies from data that is being imported into a data warehouse. See Data cleansing.

**Data Transformation**  The modification of data as it is moved into the data warehouse.

**Data Warehouse**  A database where data is collected for the purpose of being analyzed. The defining characteristic of a data warehouse is its purpose. Bill Inmon coined the term data warehouse in 1990. His definition is: “A data warehouse is a subject oriented, integrated, non volatile, time variant collection of data designed to support management DSS needs.”

**Data Warehouse Method (DWM)**  An Oracle-structured method for full life-cycle custom development data warehouse projects.

**DDL (Data Definition Language)**  SQL statements that create, modify, and remove database objects such as tables, indexes, and users. Common DDL statements are `CREATE`, `ALTER`, and `DROP`.

**Decentralized Warehouse**  A remote data source that users can query or access through a central gateway that provides a logical view of corporate data in terms that users can understand. The gateway parses and distributes queries in real time to remote data sources and returns result sets back to users.

**Decision Support**  The act of using data and tools within an organization to support managerial decisions. Usually decision support involves the analysis of many units of data in a heuristic fashion, for example, learning through the discovery process.

**Degree of a Relationship**  Indicates how many entity instances can exist at one end of the relationship for each entity instance at the other end within an ERD.
**Denormalization** A database design function that restructures a database by introducing derived data, replicated data, and repeating data. The technique is often used to enhance performance within decision support and data warehouse environments.

**Denormalized Data** The data within a denormalized database model.

**Dependent Data Mart** A data mart that is sourced directly from an existing data warehouse.

**Derived Column** A value derived by an algorithm from the values of other columns.

**Derived Data** Data that is the result of a computational step applied to reference or event data. Derived data is the result either of relating two or more elements of a single transaction (such as an aggregation), or of relating one or more elements of a transaction to an external algorithm or rule.

**Dimension (Dimension data)** The data by which a user queries the business measurables. The dimensions are stored in tables and are joined to the facts by a key value.

**Dimensions** In Oracle8i are data dictionary structures that define relationships between columns in existing tables.

**Dimension Table** In a star schema, a table that contains the data for one of the cube's dimensions. It has a primary key which is used to connect it to the fact table and one field for each level of each hierarchy contained in the dimension. The data values in these fields become the members of each of the dimension's levels.

**Dimensional Model** A model that supports a top-down design methodology. For each business process, it determines relevant facts and dimensions.

**Discrete** Usually used with reference to dimensions attributes. Data, usually text, that takes on a fixed set of values that rarely change.

**DML (Data Manipulation Language)** SQL statements that query and amend the database data. Common DML statements are `SELECT`, `INSERT`, `UPDATE`, and `DELETE`.

**Drill Across** A technique that queries data from two or more fact tables in a single report.

**Drill Down** 1) An analytical technique that queries data from a summary row and navigates through a hierarchy of data to reach the detail-level rows. 2) The ability to move between levels of the hierarchy when viewing data with an OLAP browser. Changing the view of the data to a greater level of detail.

**Drill Up** 1) An analytical technique that navigates from a detail to headers rows of data. Used to view summarized data. 2) The ability to move between levels of the hierarchy when viewing data with an OLAP browser. Changing the view of the data to a higher level of aggregation.

**DSS (Decision Support System)** An application used to provide summary or consolidated data to users for analysis, planning, and performing what-if analysis by using specialized tools that are usually driven by a graphic user interface (GUI), for example, designed to assist an organization in making decisions.
**Enterprise** A complete business consisting of functions, divisions, or other components used to accomplish specific objectives and defined goals.

**Enterprise Data** Data that is defined for use across a corporate environment.

**Enterprise Manager** An Oracle product that uses a graphical user interface in systems and databases for enterprisewide systems management.

**Enterprise Modeling** The development of a common consistent view and understanding of data elements and their relationships across the enterprise.

**Entity** 1) A database object such as a table or view. 2) A thing of significance about which information needs to be known or held as used on an ERD.

**EIS (Executive Information Systems)** Tools programmed to provide canned reports or briefing books to top-level executives. They offer strong reporting and drilldown capabilities. These tools allow ad-hoc querying against a multidimensional database, and most offer analytical applications along functional lines such as sales or financial analysis.

**ERD (Entity Relationship Diagramming)** A process that visually identifies the relationships between data elements.

**ERM (Entity Relationship Model)** A type of data model. Part of the business model that consists of many entity relationship diagrams. A process that visually identifies the relationships between data elements.

**ETL (Extract, Transform, and Load, also known as ETT)** The process of obtaining data from one data store (Extract), modifying it (Transform), and inserting it into a different data store (Load/Transportation).

**Extensible** The ability to add new functionality to existing services without major software rewrites or without redefining the basic architecture.

**Fact Data (Facts)** The measurements within the core data warehouse on which all OLAP queries depend. Generally, facts are additive measures characterized by composite keys which join to keys in the dimension tables.

**Fact Table** In a star schema, the central table which contains individual facts being stored in the database.

**Factless Fact Table** A fact table that does not contain numeric additive values, but is composed exclusively of keys. There are two types of factless fact tables: event-tracking and coverage. Event-tracking records and tracks events, such as college student class attendance, while coverage factless tables support the dimensional model when the primary fact table is sparse, for example, a sales promotion factless table.

**Foreign Key** This key column in a table references a primary key for another table, establishing relationships between tables.

**Galaxy** Constellations that do not share dimensions.

**Gateway** A software product that allows SQL-based applications to access relational and nonrelational data sources.

**Generalized Key** A dimension table primary key that is created by modifying an existing key. Generalized keys are also used with slowly changing dimensions and summary data, a data warehouse, or data mart.
Global Index  A partitioned index that is partitioned differently than its base table.

Grain  The lowest level of detail in a data warehouse or data mart. Also known as granularity.

H

Hash  Data allocated in an algorithmically randomized fashion to evenly distribute data and smooth access patterns.

HOLAP (Hybrid OLAP)  A combined use of Relational OLAP (ROLAP) and Multidimensional OLAP (MOLAP). In HOLAP, the source data is usually stored using a ROLAP strategy and aggregations are stored using a MOLAP strategy. This combination usually results in the least amount of storage space and the fastest cube processing.

Householding  In the financial services sector, assigning a customer account or individual to a collection of accounts, individuals, or locations for marketing purposes.

Hyper-Cube (Also Known As Cube and Multidimensional Cube)  A cube with more than three dimensions. A cube is an object with three dimensions, whereas hyper-cube is a cube-like structure with more than three dimensions. In the world of OLAP, hyper-cubes are almost always referred to as cubes.

I

Independent Data Mart  A data mart that is sourced directly from operational systems.

Index  An area of the database storage that is dedicated to holding key data values to allow direct access to a database row.

Inheritance  See Transference

Integrity Rules  The laws that govern the operations allowed on the data structures of a database.

L

LRU (Least Recently Used) Algorithm  A method used by Oracle’s SGA to swap out candidate blocks from the SGA shared pool area that have not exhibited much activity.

Logical Model  The phase of database design that is concerned with identifying the relationships among the tables.

M

Mapping  The process of matching data from source systems to the structures in the data warehouse.

Materialized View  A predefined user view which contains the data that corresponds to the defined query. Data changes to the base tables are automatically propagated into the materialized view.

MDX (Multidimensional Expressions)  The querying language for OLAP cubes. MDX has some similarities to SQL.

Measure  A numeric value stored in a fact table and in an OLAP cube. Sales Count, Sales Price, Cost, Discount, and Profit can all be measures in an OLAP cube.

Member  One of the data points for a level of a hierarchy of a dimension. Some of the members of the Month level of the Time dimension are January, February, March, and April.

Metadata  Data that describes the structures in the data warehouse.

Metalayer  An architectural component of the warehouse that resides between the warehouse data and the user, and contains metadata.

Metric  See Dimension
**Modality (Also Known As Optionality)**
The participation of an entity in a relationship.

**MOLAP (Multidimensional Online Analytical Programming)** OLAP that stores data and aggregations in a multidimensional database structure.

**MPP (Massively Parallel Processor)** A shared nothing architecture that takes a number of nodes and enables them to communicate rapidly.

**Multidimensional Cube** See Cube or Hyper-Cube.

**Multidimensional Database (MDBS and MDBMS or MDD)** A powerful database that allows users to analyze large amounts of data. An MDBS captures and presents data as arrays that can be arranged in multiple dimensions.

**Node** The smallest hardware component of an architecture that is capable of independently running its own operating system and database instance.

**Nonadditive** A fact that cannot be logically added between records. May be numeric and must be combined in a computation with other facts before being added across records.

**Normalization** The process of organizing data in accordance with the rules of a relational database.

**Normalized Data** Data that has been separated into groups linked by defining normal relationships, where all redundancy in the data and repeating groups of data are removed. The usual normalization level is called third normal form, represented as 3NF.

**NUMA (Nonuniform Memory Access)** A method of accessing shared memory on systems which have memory that is loosely coupled. Oracle Parallel Server can work with this access method.

**OLAP (Online Analytical Processing)**
The use of computers to analyze an organization's data. OLAP is the most widely used term for multidimensional analysis software. The term Online Analytical Processing was developed to distinguish data warehousing activities from Online Transaction Processing—the use of computers to run the ongoing operation of a business.

**OLTP (Online Transaction Processing)**
OLTP describes the requirements for a system that is used in an operational environment.

**ODS (Operational Data Store)** An integrated database of operational data. Its sources include legacy systems and it contains current or near-term data. An ODS may contain 30 to 60 days of information, whereas a data warehouse typically contains years of data.

**Optionality (Also Known As Modality)**
Indicates the minimum of entity instances that are possible at one end of the relationship for each entity instance at the other end within an ERD.

**Oracle Method** The methodology used by Oracle for corporate system implementation. Incorporates the Data Warehouse Method and project management software.
Scale, Scalable, or Scalability  The ability of a computer system or a database to operate efficiently with larger quantities of data. Scalability is often discussed in situations when multiple processors are joined together. The system scales well (or is scalable) if doubling the number of processors also doubles the speed at which the system performs its tasks. The extra work involved in coordinating larger systems usually prevents them from being fully scalable, so that going from one to two processors increases the total speed by less than a factor of two.

Schema  1) A logical representation or model of a database structure.  2) The logical organization of data in a database.

Semiadditive  A numeric fact that can be added along some dimensions in a fact table, but not others.

Server  Software that handles the functions required for concurrent, shared access to a database. The server receives and processes SQL and PL/SQL statements originating from client applications. The computer that runs the server must be optimized for its duties. The Oracle server was previously called the relational database management system.

SGA (System Global Area)  A large area of memory allocated to a database instance for caching. The SGA provides a shared area that all foreground and background processes can access.

Slice and Dice  1) The ability to move between different combinations of dimensions when viewing data with an OLAP browser.  2) A mechanism whereby a query can analyze information along any dimension of the multidimensional model equally.
**Slowly Changing Dimensions**  A dimension that has levels or attributes that change over time.

**Smart Key**  Smart keys are values that have embedded meaning.

**SMP (Symmetrical Multiprocessing)**  The shared everything approach of parallel computing.

**Snowflake Model**  A star model (or schema) whose dimensions have been normalized.

**Snowflaking**  Normalization applied to the dimension tables of a star schema.

**Source Data**  The data that is used as the basis of warehouse data, for example, from a database, flat files, or magazine articles. Also called data source.

**Sparse (Also Known As Denseness or Density)**  The degree to which the cells of a cube are filled with data. One of the primary challenges of storing multidimensional data is the degree of density that is often encountered. When many dimensions are considered with a fine grain of detail, most of the cells will be empty.

**SQL (Structured Query Language)**  The standard language for accessing relational databases.

**Staging Area**  A file, operational data store, or series of relational database server tables that contain the data to be moved to the warehouse.

**Star Model**  Star dimensional modeling is a logical design technique that seeks to present the data in a standard framework that is intuitive and provides high performance. Every dimensional model is composed of one table called the fact table, and a set of smaller tables called dimension tables. This characteristic, denormalized, star-like structure is often called a star model.

**Star Query**  Optimization technique that enables the dimensions and fact tables in the star model to be accessed efficiently, and data to be returned to the user efficiently. It ensures that the dimension data is visited first, and the fact data last and only once.

**Subject Area**  A vertical portion of the business, such as sales and marketing, that is developed as an iteration of the enterprisewide data warehouse.

**Summary Data**  Data that is aggregated and stored in a summary fact table and made available to the user for direct and easy access.

**Summary (Summarization) Tables**  Tables used to store summarized or aggregated data.

**Surrogate Key**  A simple integer that is generated by the data warehouse.

**System**  Composed of one or more nodes that together provide the hardware infrastructure for implementing the Oracle data warehouse.
**T**

**Tactical Data Warehouse Development**
The process of selecting a portion of an enterprise and implementing a data warehouse. The process includes constructing a data model for the area, determining the data warehouse architecture, constructing the physical model, and populating the warehouse database. It also includes creating or buying the applications to access the data warehouse, prototyping the tactical warehouses (access definitions, views, and so on), and incorporating end-user feedback.

**Terabyte** One trillion bytes.

**Transference (Also Known As Inheritance)** Within the star model, redundant data is posted from one object to another for performance considerations.

**Transformation** The process of defining data based on predefined rules, using specific formulas and techniques. Also called data transformation.

**Transparent Gateway** Middleware that enables viewing of data that resides in a non-Oracle database from Oracle applications.

**Transportation** The movement of data to the warehouse server.

**U**

**Uniform Resource Locator (URL)** Text used to identify and address an item in a computer network.

**Universe** Galaxies that do not share dimensions.

---

**V**

**VLM (Very Large Memory)** Computers with 64-bit memory structures.

**VLDB (Very Large Database)** Measured in gigabytes and terabytes.

**Warehouse Manager** The mechanism that maintains the data in the warehouse database.

**Warehouse Technology Initiative (WTI)** An Oracle program that invites other vendors to offer products and services that are complementary to those offered by Oracle, particularly in the area of products and services related to data warehousing.