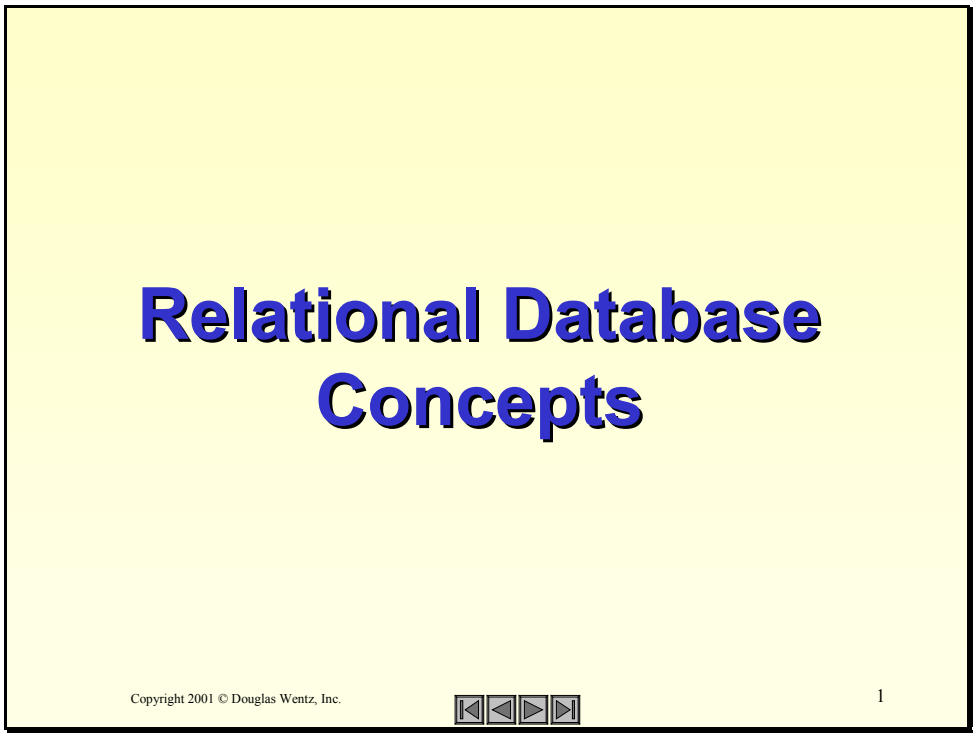


Module Overview

This module introduces the basic concepts of Relational Databases, the Relational Model and an introduction to SQL*Plus. In this module the participant will learn the basic implementation Of Relational Database concepts. This module is not intended to provide the participant a complete training course in understanding of the Relational Model although a very brief introduction is provided as a refresher for the course participant.

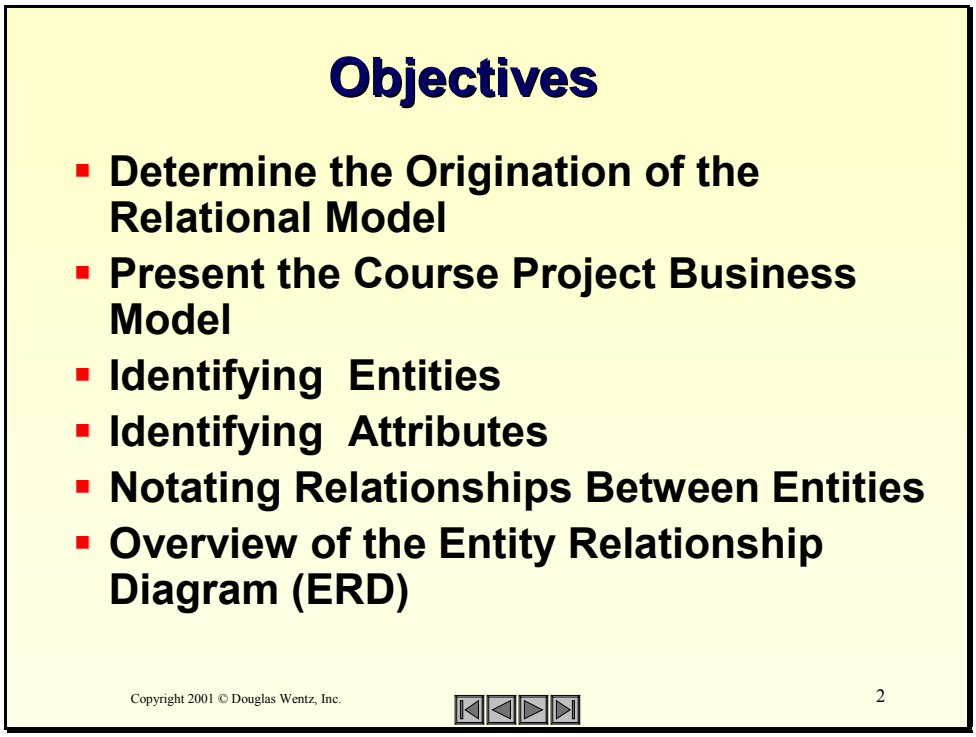
Completion Time

Estimated time to complete the course material is 2 hours and 30 minutes for the lab.

Location of Presentations , Labs, & Examples

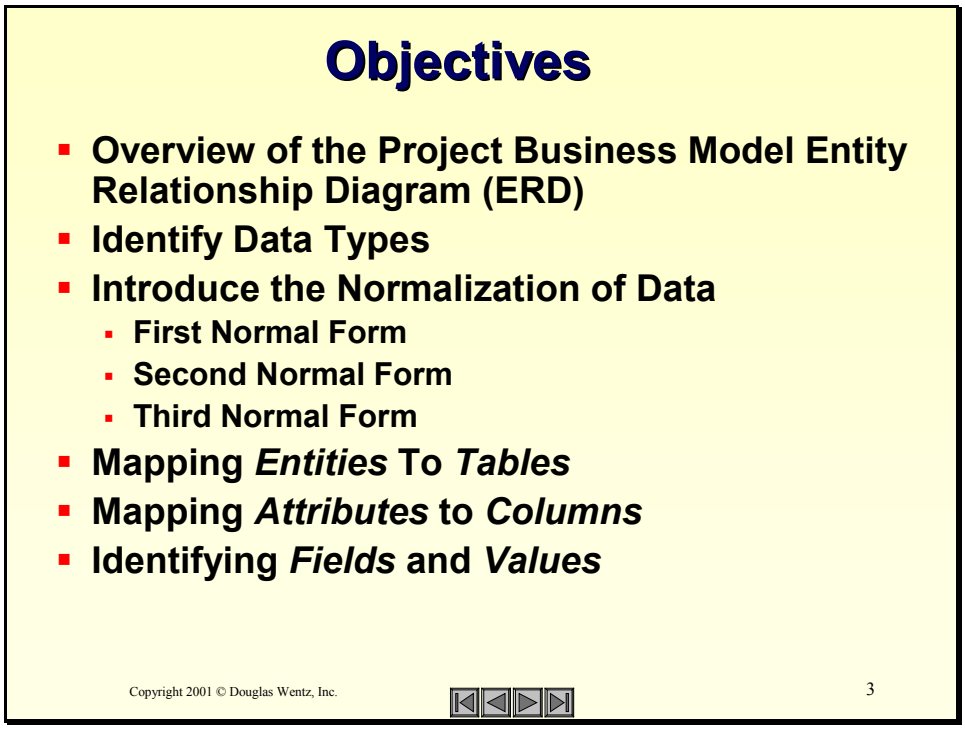
All presentations, labs, and examples are located on the *Oracle Database Administration Certified Professional* training CD in the directories and file names as follows.

DBAOCP\IntroOracle9iSQL\Labs\	Lab scripts
DBAOCP\IntroOracle9iSQL\PPTS\	PowerPoint presentations
DBAOCP\IntroOracle9iSQL\Examples\	Presentation examples
DBAOCP\IntroOracle9iSQL\Docs\	This documentation



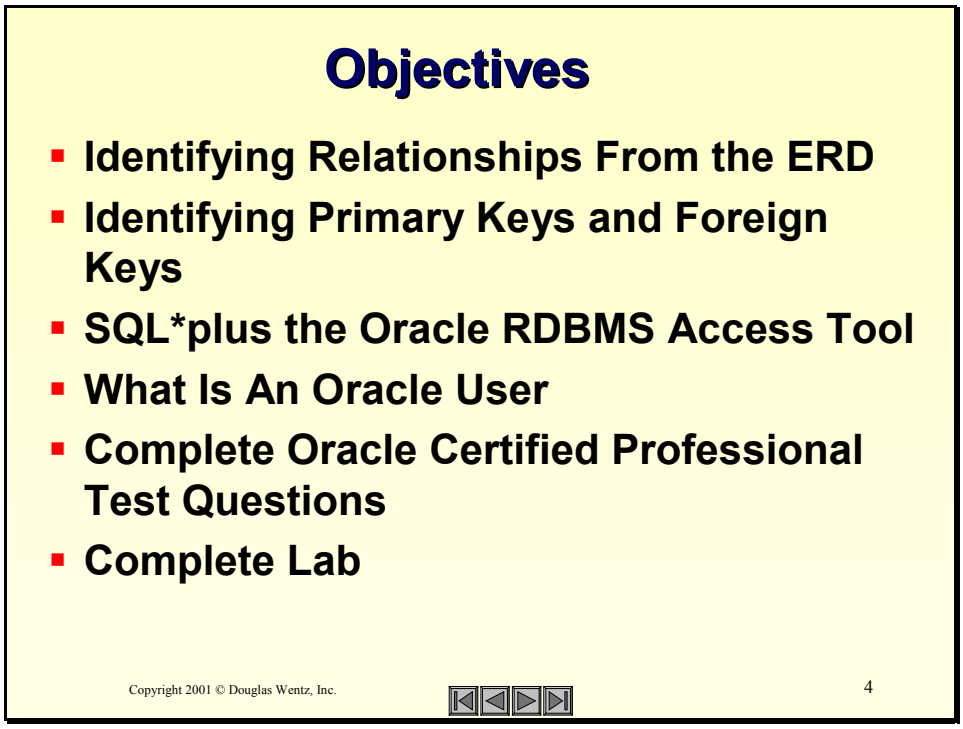
Objectives

- Determine the origination of the Relational Model.
- Present the course project business model.
- Identify Entities
- Identify Attributes
- Notating relationships between entities
- Overview of the Entity Relationship Diagram (ERD)



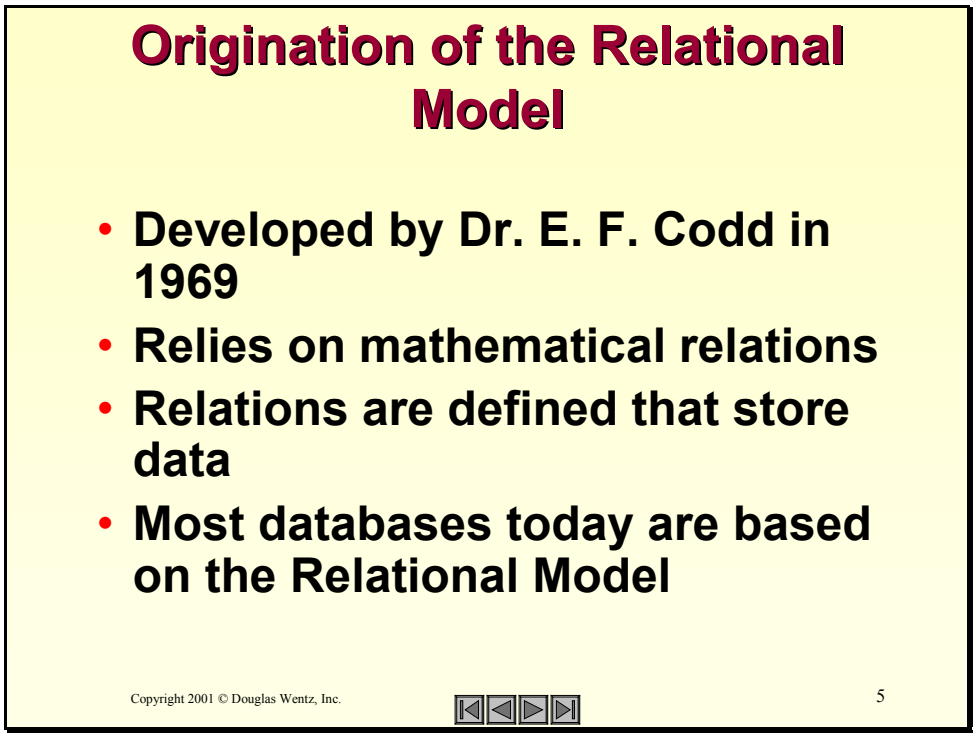
Objectives

- Present an overview of the project business model Entity Relationship Diagram (ERD).
- Identify data types supported by Oracle.
- Introduce the normalization of data.
    - First normal form.
    - Second normal form.
    - Third normal form.
- Mapping of entities to tables.
- Mapping of attributes to columns.
- Identifying fields and values.

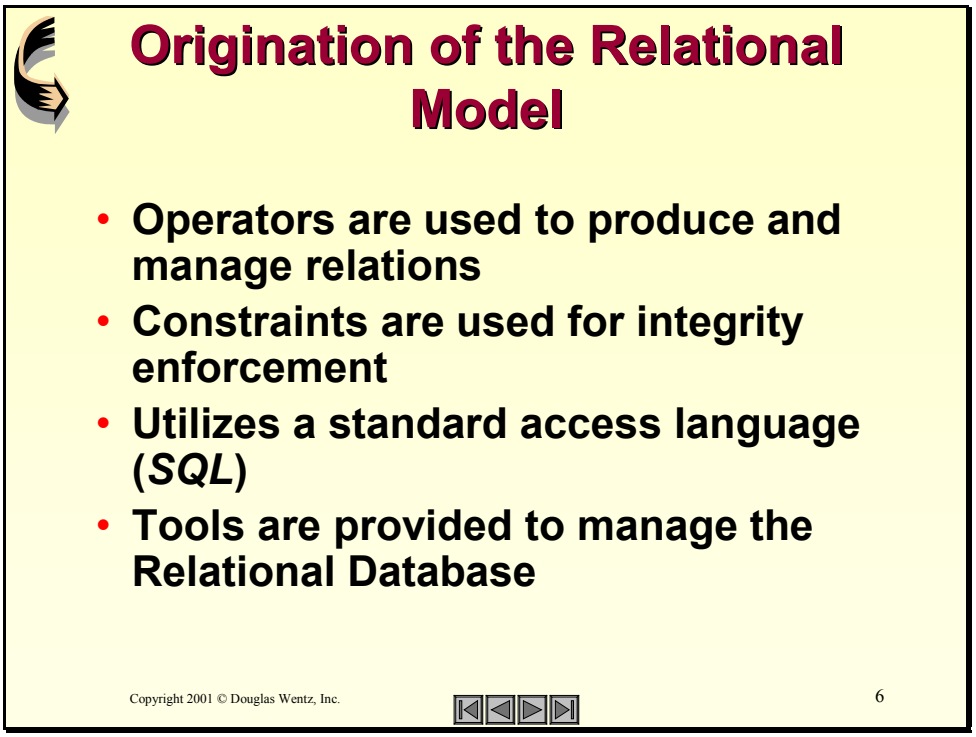


- Identify the relationship from the ERD.
- Identifying primary keys and foreign keys.
- Introduce SQL*Plus the Oracle RDBMS access tool.
- Define what is an Oracle User.
- Complete the Oracle Certified Professional test questions.
- Complete the lab.

Objectives

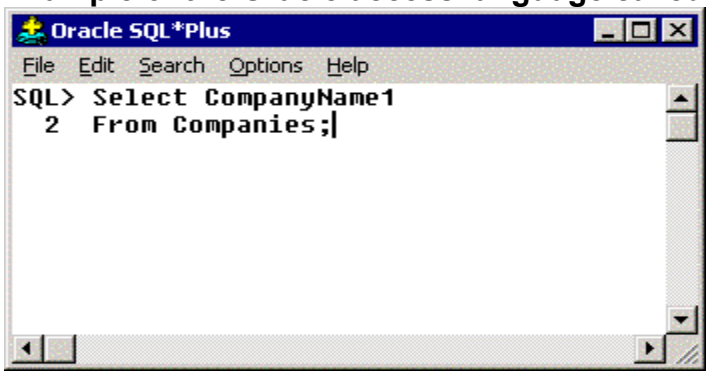


- The **Relational Model** was proposed by Dr. E. F. Codd in 1969 in a paper Dr. E. F. Codd wrote
- The paper was entitled "*Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks*"
- The **Relational Model** was an enormous advancement over other database models.
- Its distributed capabilities have fueled the client/server revolution today.
- The **Relational Model** relies on solid mathematical relations.
- Relations define the relationships between the entities.
- The **Relational Model** can be used in both conceptual and logical database design.
- Most databases today are based on the **Relational Model** including Oracle's RDBMS.

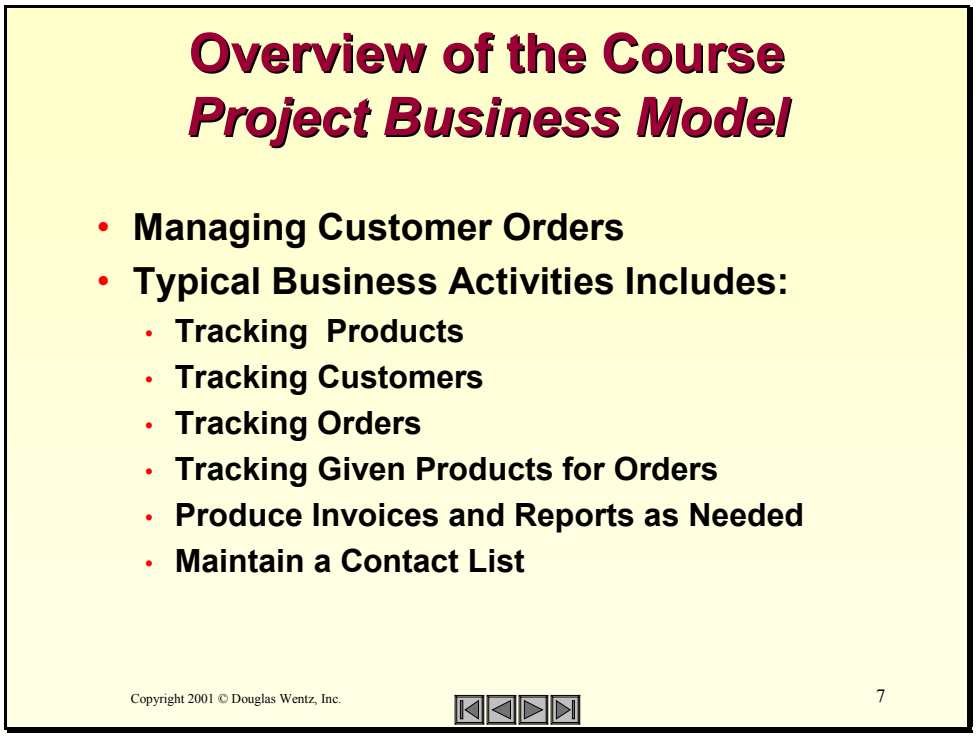


- Operators are used to produce and manage relations
- Constraints are used for integrity enforcement. Constraints are usually enforced by the use of primary keys and foreign keys.
- Relational databases use a standard access language called SQL. SQL is pronounced 'SEQUEL' and is a standard access language for many relational databases from different vendors.
- Tools and utilities must be provided to manage the relational model. Oracle provides tools including the tool SQL*Plus.

Example of the Oracle access language called SQL*Plus



Note: Many other tools are provided by Oracle and third party vendors.



All labs throughout this class and subsequent classes will utilize the *Project Business Model*. The *Project Business Model* is based on a company that develops, markets and sells courseware. Actually, the company is just like the company this class material was purchased from. The *Project Business Model* must meet the following requirements.

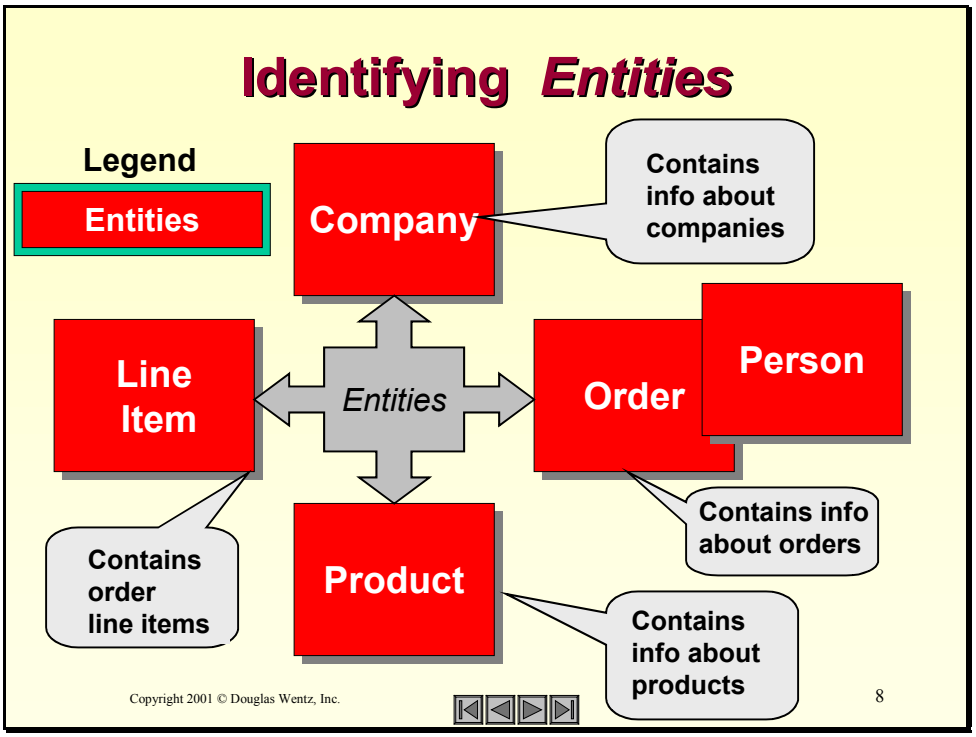
- The hypothetical company sells training courseware. This courseware is just like the courseware being used in this class.
- The courseware is broken down into classes just like the class you are taking now. Classes are further broken down into modules. For instance this module ***Relational Database Concepts*** is part of the ***Introduction to Oracle 9i SQL*** class. The **Introduction to Oracle 9i SQL** class is part of the ***Oracle 9i Database Administrator Certified Professional*** program.
- Customers must be tracked and retained. This includes their company name, contact persons, phone numbers, and finally address. Some customers may receive special discounts.
- Whenever orders are placed the company name, classes, modules requested, quantity and pricing must be obtained from the customer.
- Shipping information must be documented including order date, date requested, and ship date. Customers may pay by cash, credit card or by check. A customer may in fact request just a quote.
- Inventory stock levels must be maintained and documented.

- Number of courseware classes sold must be maintained and documented.
- Order and customer information must be retained indefinitely. This is required for statistical analysis.
- Invoices and reports must be produced at regular intervals and on demand.
- Contact persons at each company must be maintained.

Example of the Program, Class, and Module structure of courseware

Program	Class	Module
Oracle 9i Database Administration	Introduction to Oracle 9i SQL	Relational Database Concepts
		Oracle Architecture Overview
		SQL Query Fundamentals
	Oracle 9i Database Administration I	
	Oracle 9i Database Administration II	
	Oracle 9i Performance Tuning	

Note: It is imperative that the participant in this course thoroughly understands the *Project Business Model*. Each concept presented in this course and utilized subsequent classes utilize examples based on the *Project Business Model*. Each lab in this course is dependent upon the understanding of ***the Project Business Model***.

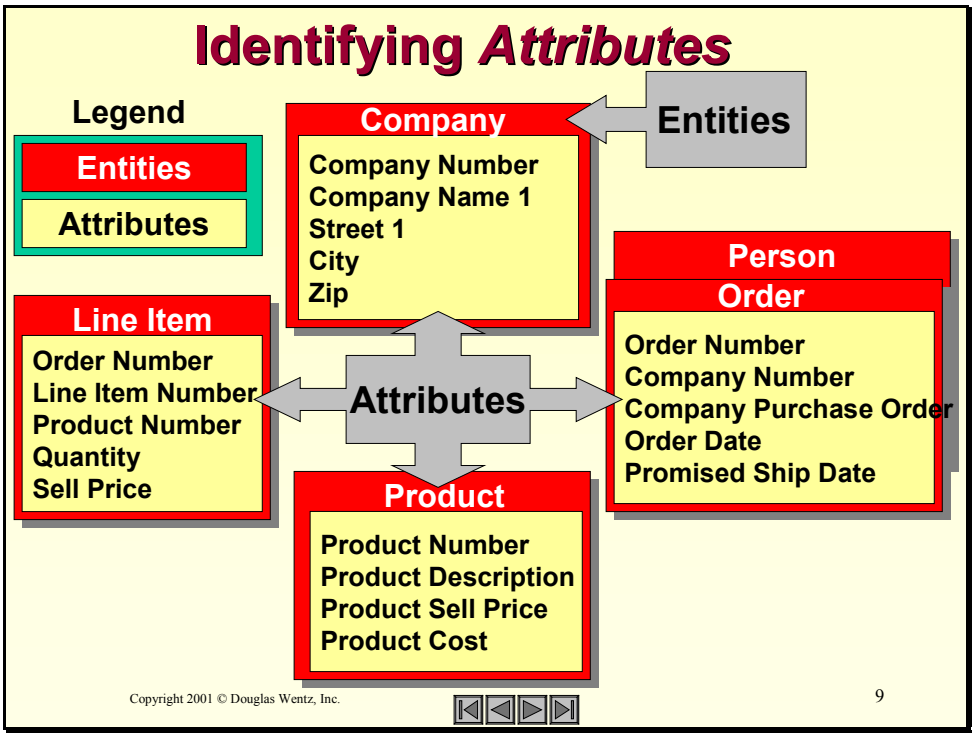


- An **Entity** is a person, place, or thing. An **Entity** can also be an object or an event, which can be distinctly identified and is of interest.
- An **Entity** is usually a noun. A noun is generally a person, place, or thing of interest.
- An **Entity** is anything in which the business has an interest.
- An **Entity** is an information container.
- An **Entity** should be singular in name.
- Real world examples for many *Business Models* would be;

Person	Address	Organization	Account
Inventory	Store	Customer	Account Holder

- In our *Business Project* example we want to store information about a;

Customer	Product	Person
Order	Line Item	

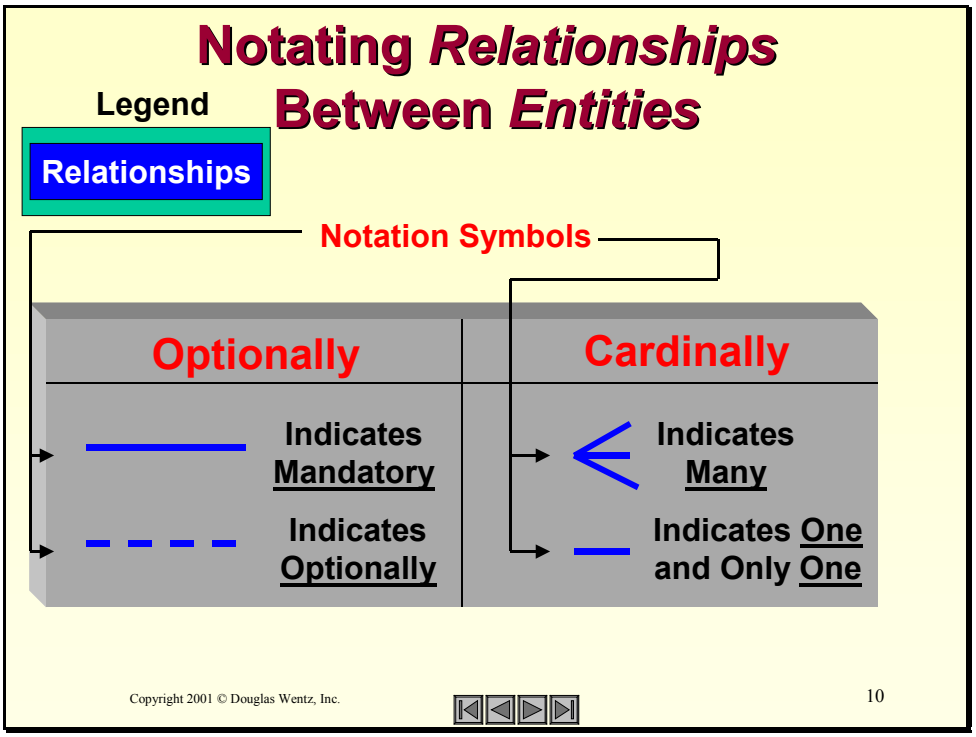


- An **Attribute** describes the quality, characteristics or property of an **Entity**.
- An **Entity** is compromised of many **Attributes**. An **Entity** must have at least 2 **Attributes**.
- An **Attribute** is an information container just like the **Entity** is an information container. **Attributes** contain detailed information about the **Entity**.
- In our *Project Business Model* example we want to store information about a Company.
- Typical real world Attributes for the Company **Entity** would be;

  Company Number Street State
  Company Name City Zip

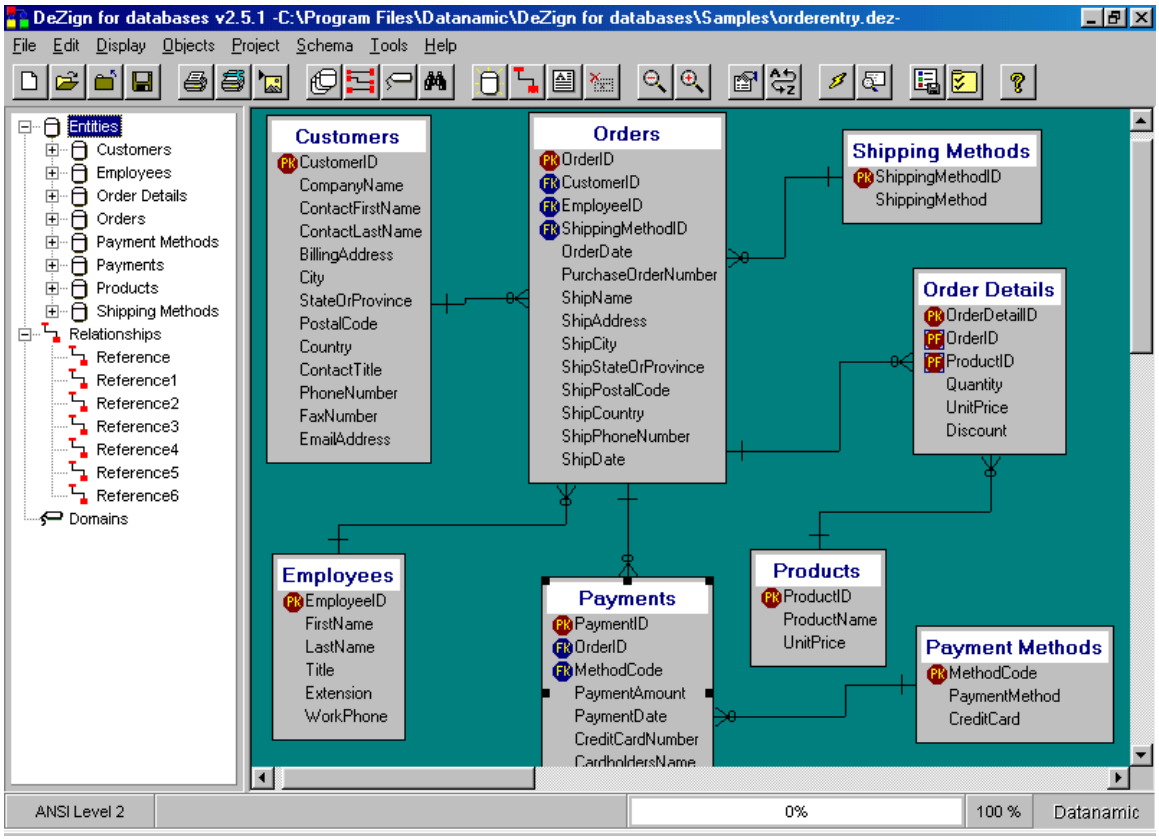
- Typical real world **Attributes** for the Product **Entity** would be;

  Product Number Product Description
  Product Sell Price Product Cost

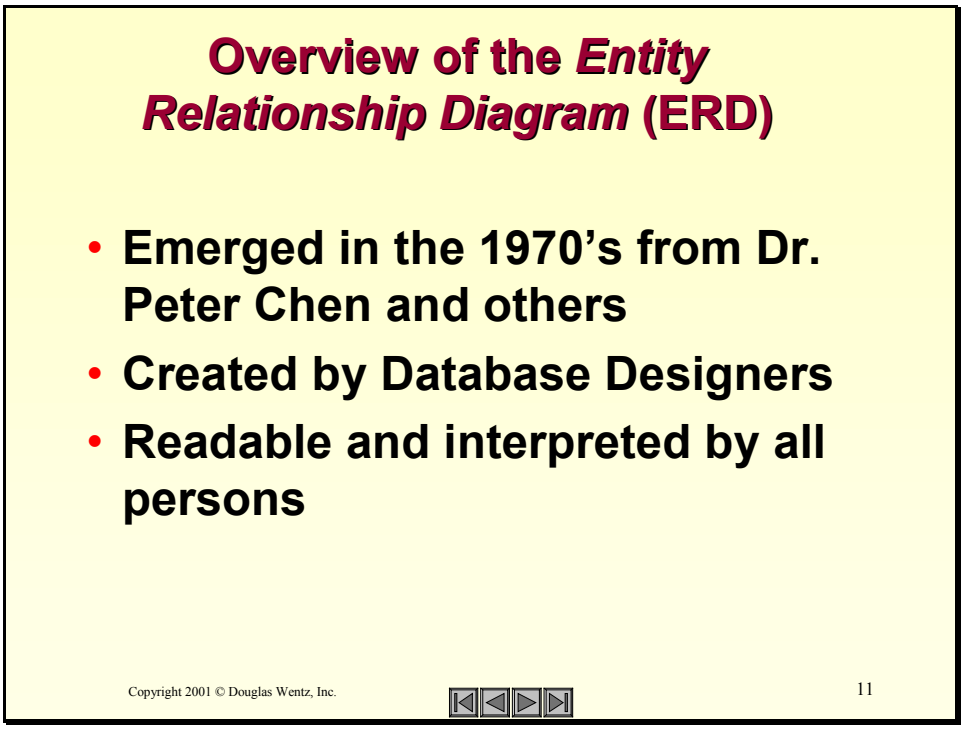


- A relationship is an association among the **Entities**. Relationships illustrate how two or more **Entities** share information in the database structure.
- In our *Project Business Model* example a relationship exists between the **Entity** Customer and the **Entity** Order. An Order must have a Customer. A customer places orders.
- If the relationship is required it is referred to as being **Mandatory**. A **Mandatory** relationship is shown as a solid blue line in the illustration above. An Order must have a Customer assigned to it.
- If the relationship is not required it is referred to as being **Optional**. An **Optional** relationship is shown as a dashed blue line in the illustration above. A Customer does not have to place an order although we would like them to.
- If many instances of the relationship exists it is referred to as being **Many**. The **Many** instances is illustrated above with the three solid lines making a crows feet. A Customer can have many Orders.
- If one and only one of the relationships exist between two entities it is referred to as being **One**. The **One** instance is illustrated above with the single line. An Order must be for one on only one Customer.

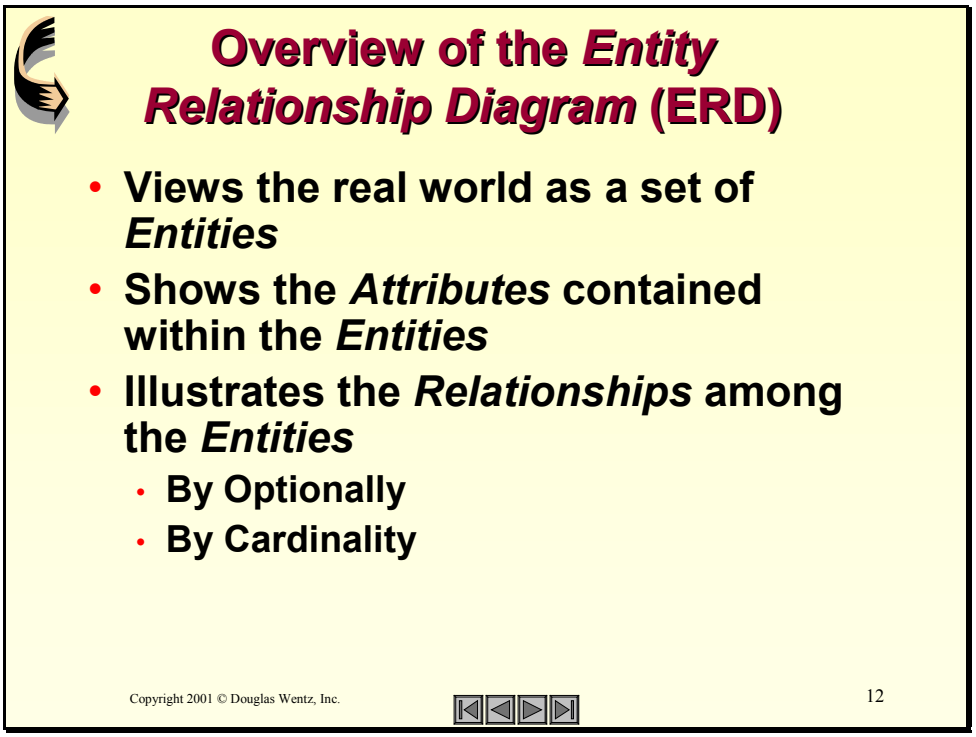
Note: This is only one of several methods of notating relationships among entities. Another method is illustrated below.



This Software design product is provided by DeZign for Databases.



- The **Entity Relationship Diagram** (ERD) emerged in the 1970's from work by Dr. Peter Chen and others.
- **Entity Relationships Diagrams** (ERD) are generally created by Database Designers. ERD's represent data that an organization needs and not their business processes. ERD's graphically illustrates a collection of Entities and **Attributes** in a real business world sense. The components of an ERD are;
    - **Entities**
    - **Attributes**
    - **Optionally**
    - **Cardinality**
- **Entity Relationship Diagrams** are readable by all persons including data modelers, managers and even end users. Anyone, including end users should be able to read an **Entity Relationship Diagram** (ERD). **Entity Relationship Diagrams** are generally used by application developers and DBA's in creating tables, constraints on tables, and to provide an understanding of the data model for developers.



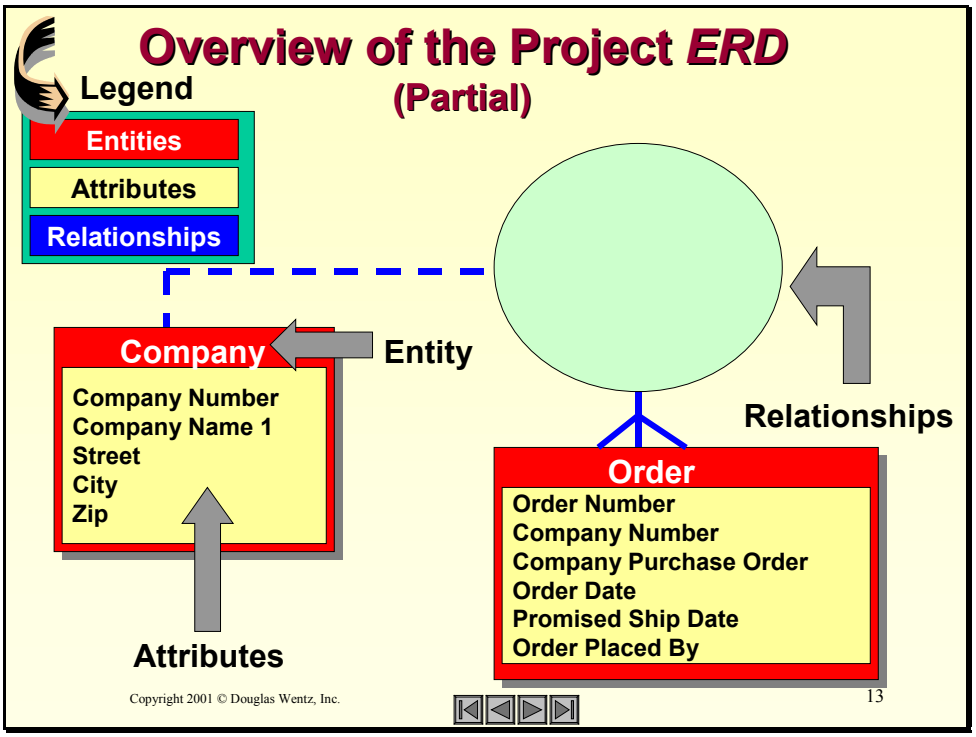
- ERD's view the real world as a set of **Entities**. Remember **Entities** are nouns and contain some set of information about the business. In our *Project Business Model* the **Entities** are;

  Customer Product Person
  Order Line Item

- **Attributes** are contained within the **Entities**. An **Attribute** describes the quality, characteristics or property of an **Entity**. Remember the **Attributes** of the Company **Entity** include;

  Company Number Company Name
  Street City
  Zip State

- ERD's also illustrate the relationship among the **Entities**. Remember an Order must be assigned to a Company however a Company may not have placed any Orders. This is referred to as the **optionally** of a relationship.



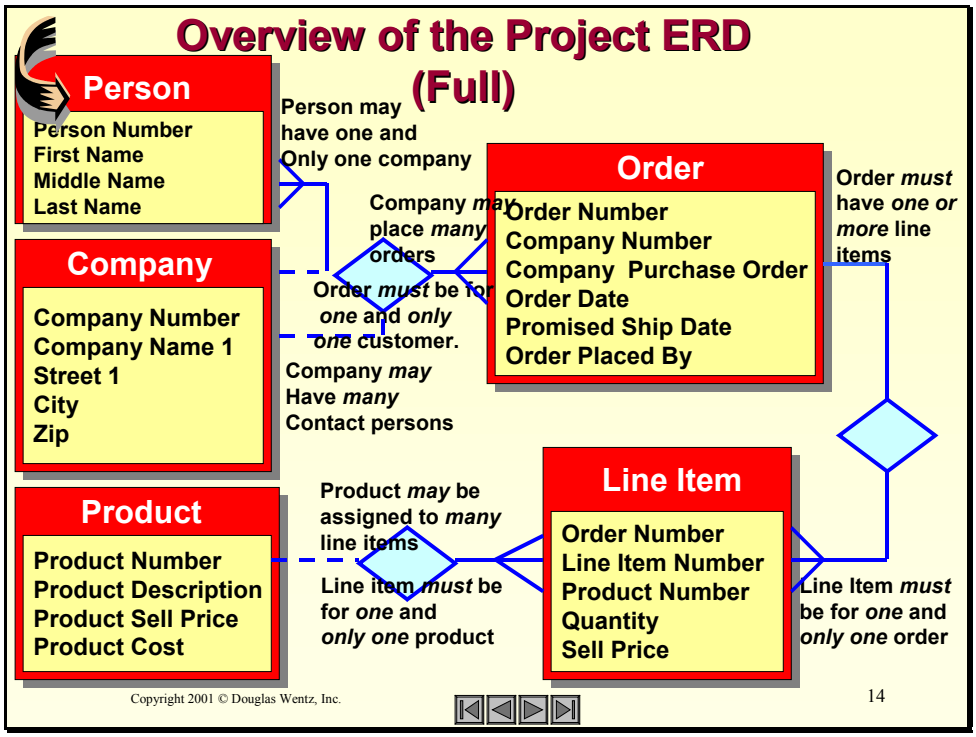
- From the above illustration we can determine the following facts about the **Entity** Company and the Entity Order. This is only a partial ERD diagram of our *Project Business Model*.
    - The dotted line from the **Entity** Company toward the **Entity** Order indicates that a Company may have an Order. This is the optionally of the relationship.
    - The solid line from the Order **Entity** toward the Company **Entity** indicates that an Order must be assigned to a Company. This again is the optionally of the relationship.
    - The crow's feet indicate a many relationship. This essentially means that a Company may have many Orders. This is the cardinality of the relationship
    - The Company **Entity** has the following attributes about it;

      Company Number Company Name Street
      Zip City State

r

    - The Order **Entity** has the following attributes about it;

      Order Number Company Number Promised Ship Date
      Company Purchase Order Order Date Order Placed By



- This is a full ERD of the *Project Business Model*. The following facts can be determined for the ERD.

  1. The *Project Business Model* contains five **Entities** including;

     Company Product Person
     Product Line Item

  2. The Company **Entity** contains the following **Attributes**;

     Company Number Company Name 1
     Contact Street 1 Contact City
     Contact Zip

  3. The Product **Entity** contains the following **Attributes**;

     Product Number Product Description
     Product Sell Price Product Cost

4. The Order **Entity** contains the following **Attributes**;

   Order Number
   Company Number
   Company Purchase Order
   Order Date
   Promised Ship Date
   Order Placed By

5. The Line Items **Entity** contains the following **Attributes**;

   Order Id
   Line Item Number
   Product Id
   Quantity
   Price

6. The Person **Entity** contains the following **Attributes**;

   Person Number
   Middle Name
   First Name
   Last Name

7. The dotted line from the Company **Entity** to the Order **Entity** and the crow's feet indicates that a Company may place many orders. We would not limit a Company to just one Order.

8. The solid line from the Order **Entity** to the Company **Entity** indicates that an order must be for one and only one Company. An order cannot be split between two or more Companies. This would not make good business sense.

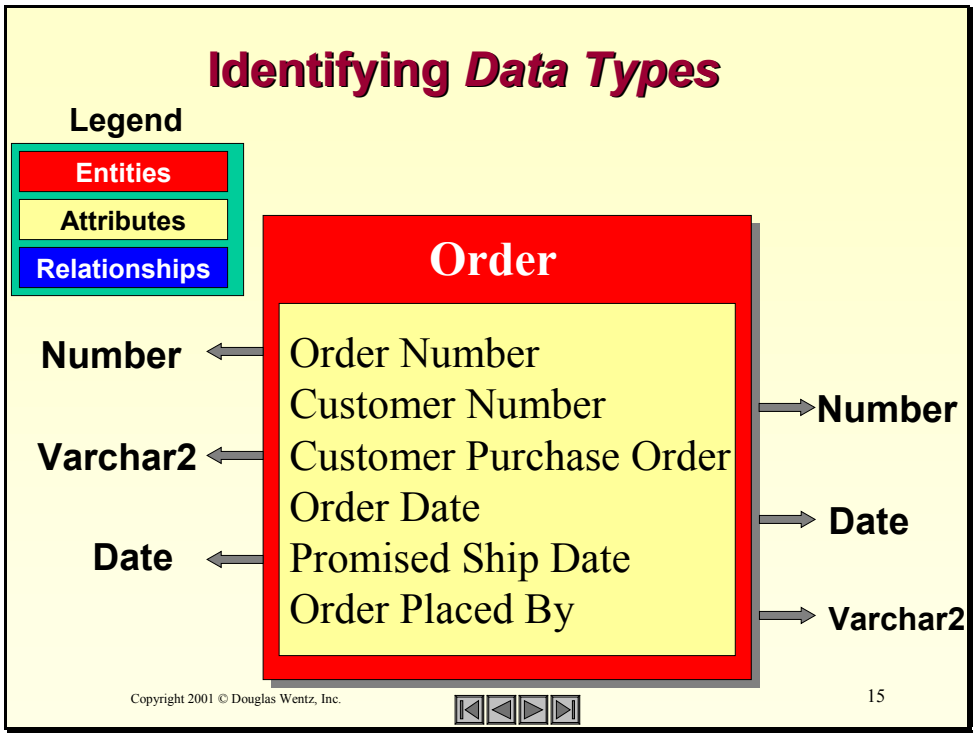
9. *The* solid line from the Order **Entity** to the Line Item **Entity** and the crows feet indicates that an order my have many line items. A Company may order more than one item.

10. *T*he solid line form the Line Item **Entity** to the Order **Entity** indicates that a line item must be for one and only one order. A line item cannot be for more than one order. This would not make good business sense.

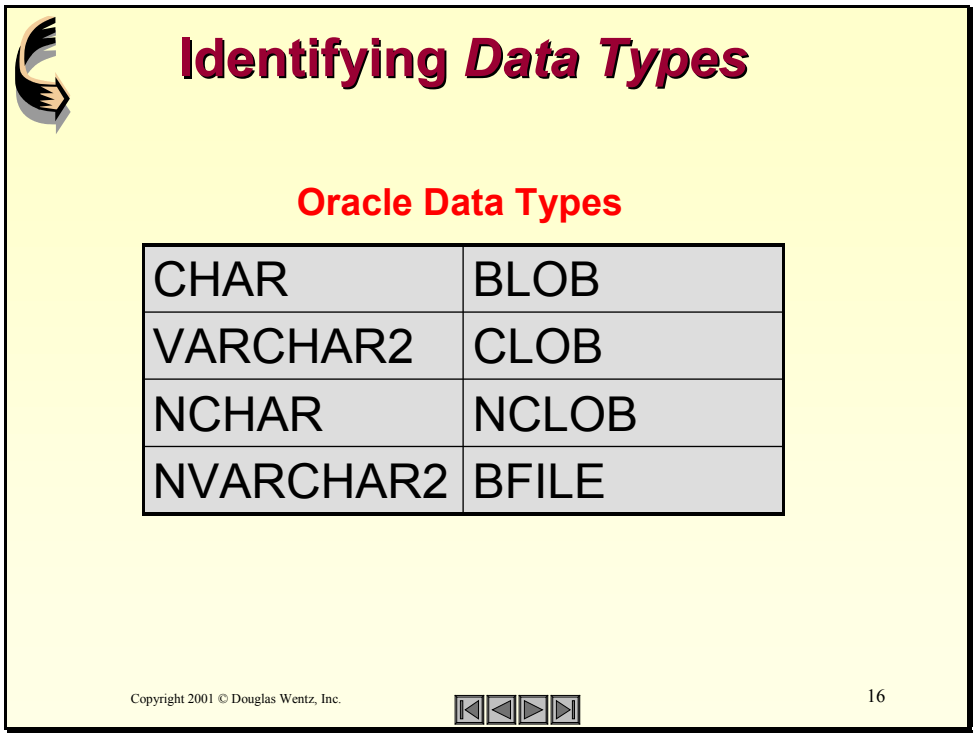
11. The dotted line from the Product **Entity** to the Line Item **Entity** and the crow's feet indicates that a Product may be for many orders. We would not limit a product to just one line item.

12. The solid line for the Line Item **Entity** to the Product **Entity** indicates that a line item must be for one and only one product.

Note: Not all the attributes for the entities are shown, only the most important attributes are shown.

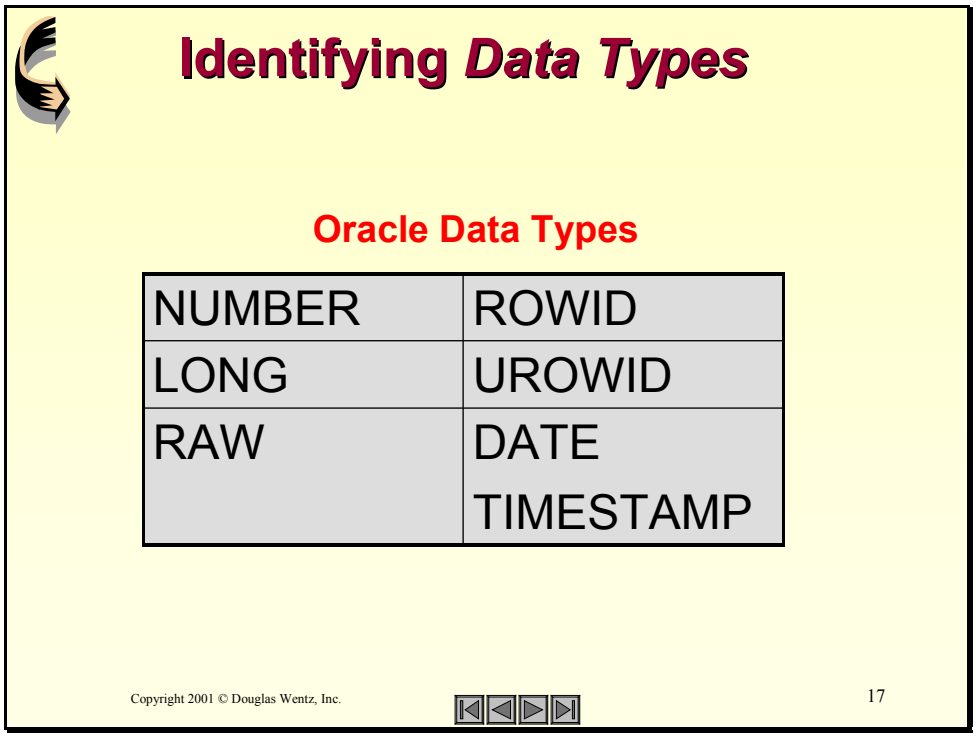


- Attributes must be assigned to a data type. A data type determines what format should data be stored in the Oracle database. There are 4 basic Oracle Built-in Data types.
    - **Character** – Stores characters and numbers.
    - **Numbers** - Stores only numbers and math can be applied.
    - **Dates** – Stores only dates and times.
    - **Binary** – Stores pictures, documents, and video files.
- The assignment of data types to attributes helps significantly in the validation of data by Oracle.
- Oracle provides the capability of users defining their own data types. The creation of user-defined data types will be covered in the *Oracle's Object Extensions* module within this class.
- Additionally Oracle provides SQL-based interfaces for defining new data types when the built-in data types are not sufficient. These can be implemented in C/C++, Java, or PL/SQL. This is beyond the scope of this class.



Oracle Data Types

CHAR	Fixed length character string. The maximum size is 2000 bytes. Will have trailing blanks.
VARCHAR2	Variable length character string. The maximum size is 4000 bytes. Will not have trailing blanks
NCHAR	Fixed length character string. Generally used with National Character Sets. The maximum size is 2000 bytes. Will have trailing blanks.
NVARCHAR2	Variable length character string. Generally used with National Character Sets. The maximum size is 4000 bytes. Will not have trailing blanks.
BLOB	Unstructured binary data up to 4 gigabytes.
CLOB	Single byte character data up to 4 gigabytes.
NCLOB	Single byte or fixed length multibyte National Character Set up to 4 gigabytes.
BFILE	Binary data stored in an external file.



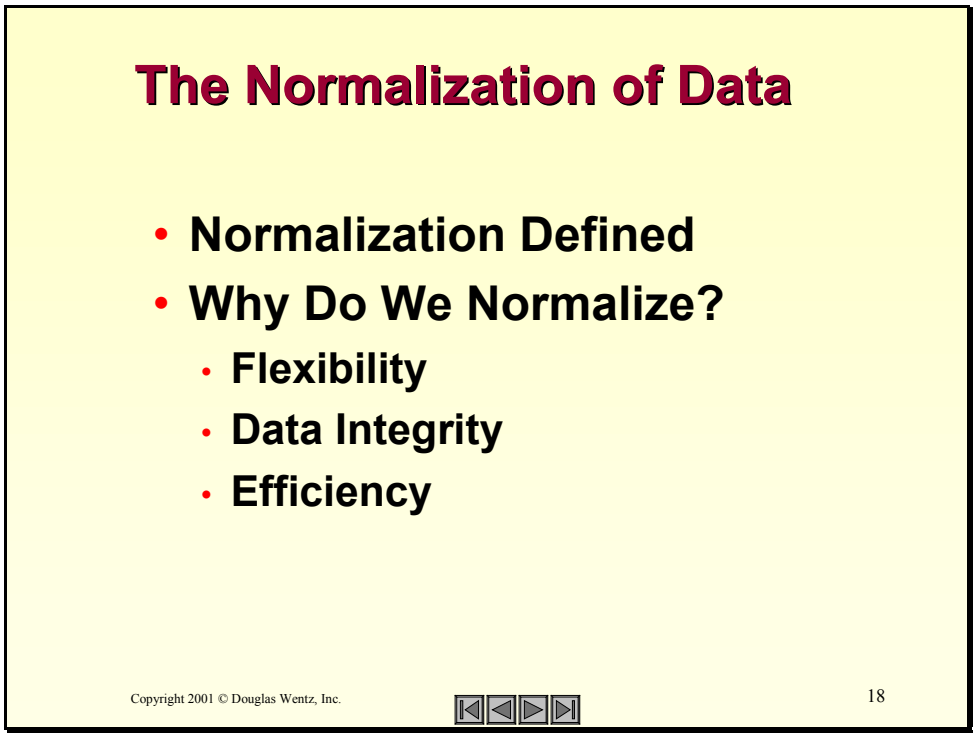
Oracle Data Types

NUMBER	Variable length numeric data. Maximum length is 21 bytes.
LONG	Variable length character data. Maximum length is 2 gigabytes. Provide for backward compatibility
RAW	Variable length raw binary data. Maximum length is 2000 bytes. Provided for backward compatibility.
Long Raw	Variable length raw binary data. Maximum length is 3 gigabytes. Provided for backward compatibility
DATE - TIMESTAMP	Fixed length date and time. Uses 7 bytes. Timestamp is new.

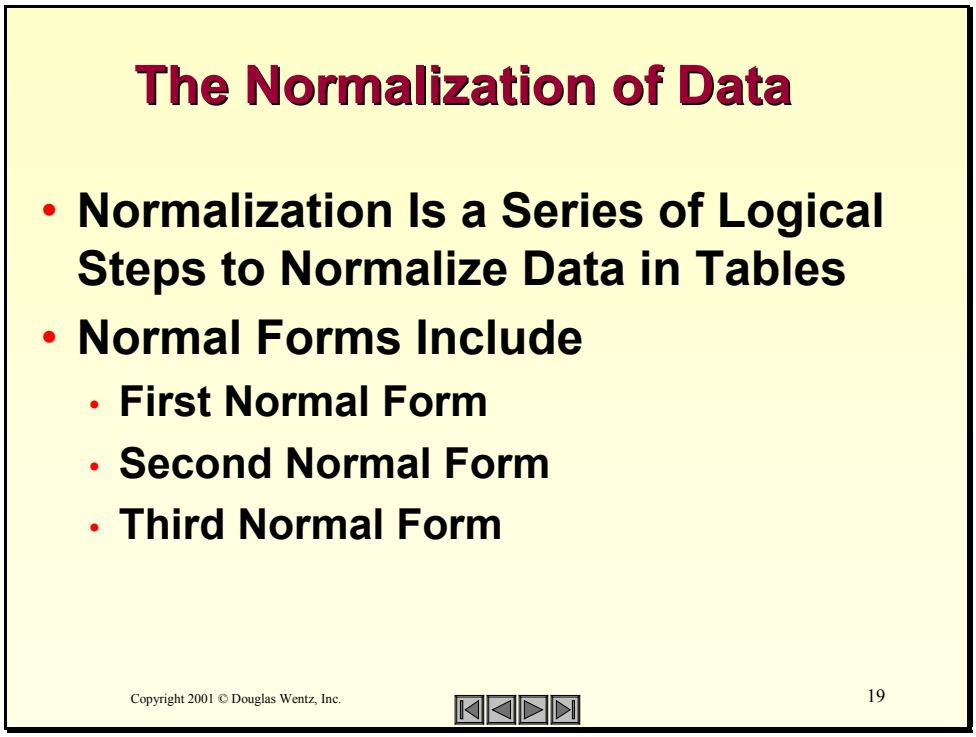
Note: The LONG, RAW, and LONG RAW data type will not be supported at some point by Oracle. Try not to use these data types. These data types are being presented in this material for backward comp ability

Note: The CHAR and VARCHAR2 data types are and will always be fully supported. At this time, the VARCHAR data type automatically corresponds to the VARCHAR2 data type and is reserved for future use.

Note: The ROWID and UROWID will be presented latter in this class.



- Normalization refers to the process of creating an efficient, reliable, flexible, and appropriate relational structure for the storing of information in the database.
- Normalized data must be in a relational structure.
- Relational databases are normalized to minimize duplication of data.
- Normalization usually involves dividing data into two or more tables and defining the relationships between the tables.
- We normalize data to obtain the following results;
    - **Flexibility** supports many ways to look at the data.
    - **Data Integrity** prevents data anomalies whenever performing deletions, insertions, and update to the data.
    - **Efficiency** prevents redundant data and saves space in the database.



- Normalization is a series of logical steps to normalize data in tables. The goal of normalization is to create a set of relational tables that are free of redundant data and can be consistently and correctly modified.
- Normalization of data involves the process of going through three steps.
- The steps of normalization is as follows;
    - **First Normal Form (1NF)**
    - **Second Normal Form (2NF)**
    - **Third Normal Form (3NF)**
- Generally all tables in a relational database should be in **Third Normal Form**.
- In some instances especially for database performance reasons data may not be in **Third Normal Form**.

The Normalization of Data
First Normal Form (1NF)

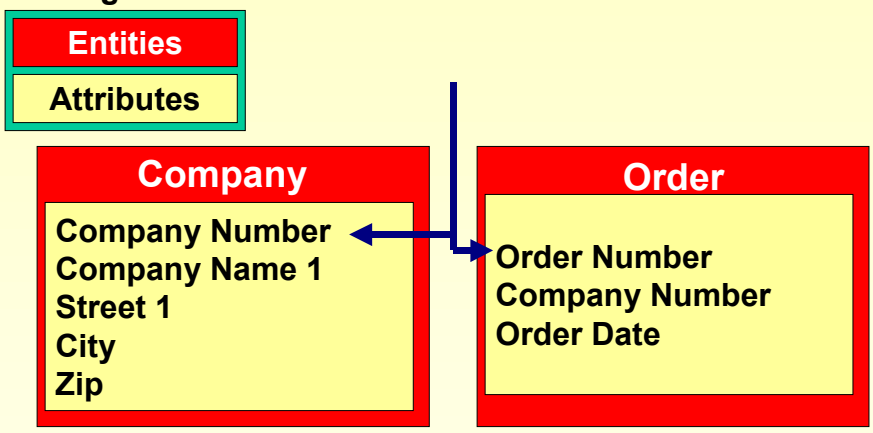
Order Date	Company	Product 1	Product 2
10-MAR-02	Relational Technologies	Introduction Oracle 9i PL/SQL	Oracle 9i Database Administration
12-MAR-02	United Training Services	Oracle 9i Backup and Recovery	Oracle 9i Database Administration

- **First Normal Form requires no repeating items in columns**

Copyright 2001 © Douglas Wentz, Inc. 20

- **First Normal Form** requires that there are no repeating items in the columns.

The Normalization of Data
First Normal Form (1NF)



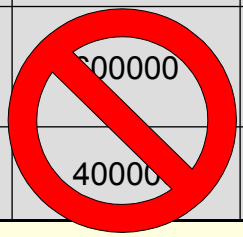
Solution is to make a separate table for each set of attributes with a

Copyright 2001 © Douglas Wentz, Inc. 21

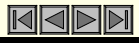
- Solution is to make a separate table for each set of attributes.

The Normalization of Data
Second Normal Form (2NF)

Order Number	Product Number	Company Number	Order Date
1	500000	135	10-MAR-02
1	400000	135	10-MAR-02



- In 1NF and every non-key column is fully dependent on the (entire) Primary key

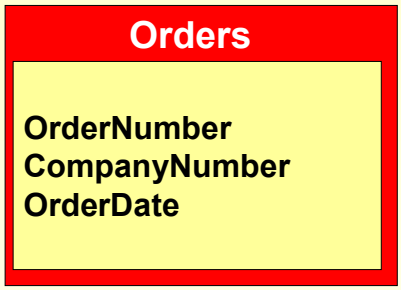


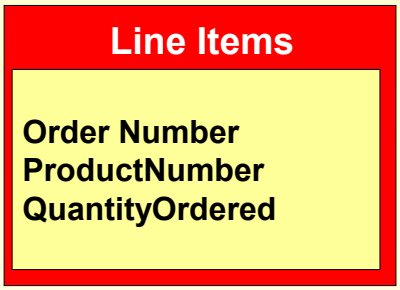
22

- Must be in **1NF** and every non-key column is fully dependent on the (entire) **Primary Key**.

The Normalization of Data
Second Normal Form (2NF)

Legend





Solution is to move the products ordered to a separate table.

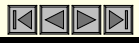
23

- Solution is to move the products ordered to a separate table.

The Normalization of Data
Third Normal Form (3NF)

Order Number	Product Number	Quantity Ordered	Price	Total
1	600000	10	99.00	990.00
1	400000	10	99.00	999.00

- **In 2NF and every non-key columns is mutually independent.**



24

- Must be in **2NF** and every non-key columns is mutually independent. Really means no calculations.

The Normalization of Data
Third Normal Form (3NF)

Legend

Entities

Attributes

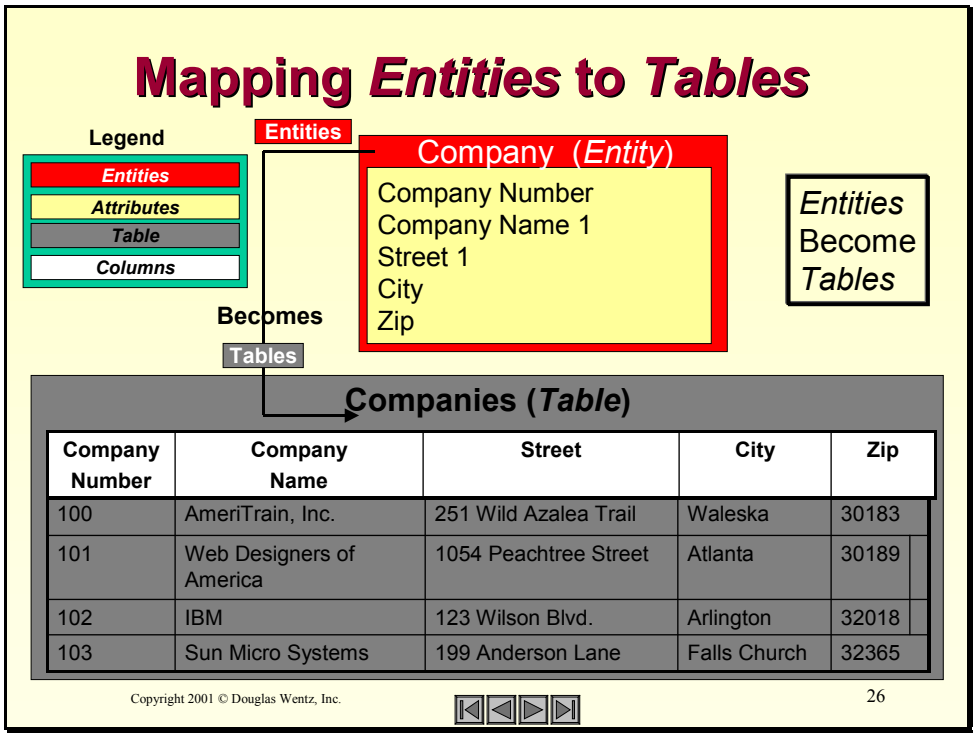
Line Items

Order Number
ProductNumber
QuantityOrdered
Price

Solution is to put calculations in queries and forms.

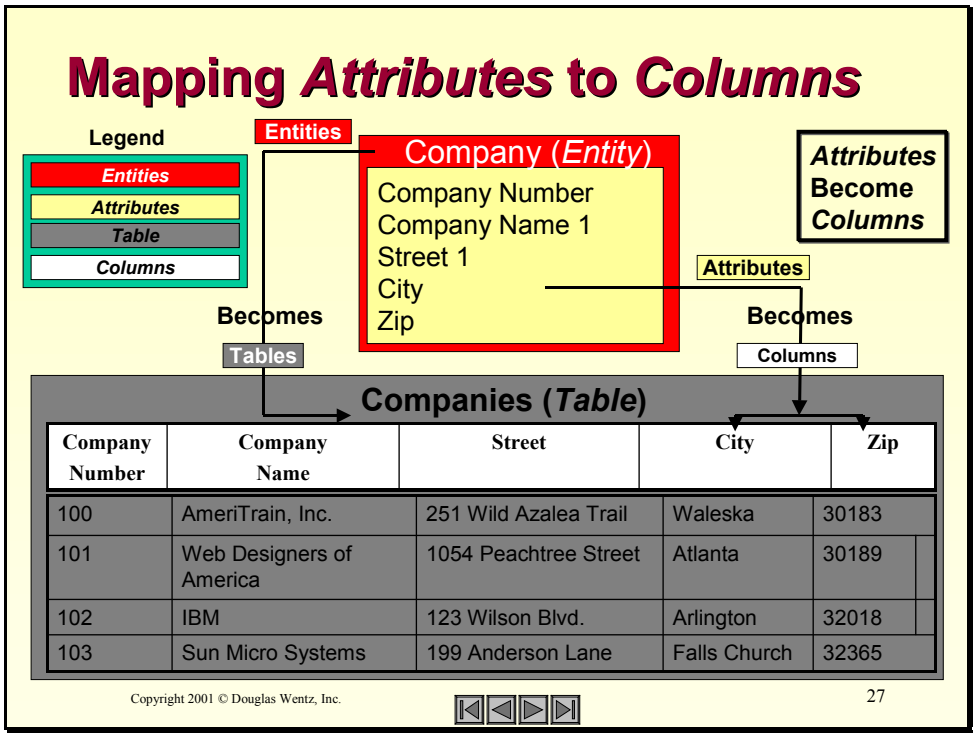
25

- Solution is to put calculations in queries and forms.



- A **Table** is the basic unit of storage in a relational database.
  - A **Table** is synonymous to a file in a traditional file system. **Tables** contain data.
  - Each **Table** is named and columns for each data element of the **Table** are defined.
  - After a **Table** has been defined and created data can be inserted into it.
  - Generally a **Entity** will directly map directly to a **Table**
  - In the example above the **Entity** Company will become a **Table** named Companies.
  - **Tables** can be singular or plural however you should be consistent in naming them.
- So in our *Project Business Model* all the **Entities** will become **Tables** as illustrated below.

Entity Name	Table Name
Company	Companies
Product	Products
Order	Orders
Line item	Lineitems
Person	Persons



- A **Table** is made up of one or more **columns**.
  - A column is the vertical aspect of a **Table** and is synonymous to a field in a traditional file system.
  - Each c**olumn** within a table has a unique name and certain data types, lengths, precision and scale.
  - A **column** may be defined as an array and can store multiple values for a single row.
  - Generally an Attribute will directly map to a **Column**.
- In the example above the **Entity** Company becomes a **Table** named Companies. The **Attribute** Company Number becomes a **Column** named Company Number. This continues for each of the Attributes as illustrated on the next page.

Attributes For Entity Company Becomes Columns For Table Companies

Attribute	Column
Company Number	CompanyNumber
Company Name	CompanyName1
Street 1	Street1
City	City
State	State

Attributes For Entity Person Becomes Columns For Table Persons

Attribute	Column
Person Number	PersonNumber
First Name	FirstName
Last Name	LastName
Middle Name	MiddleName
Suffix	Suffix

Attributes For Entity Product Becomes Columns For Table Products

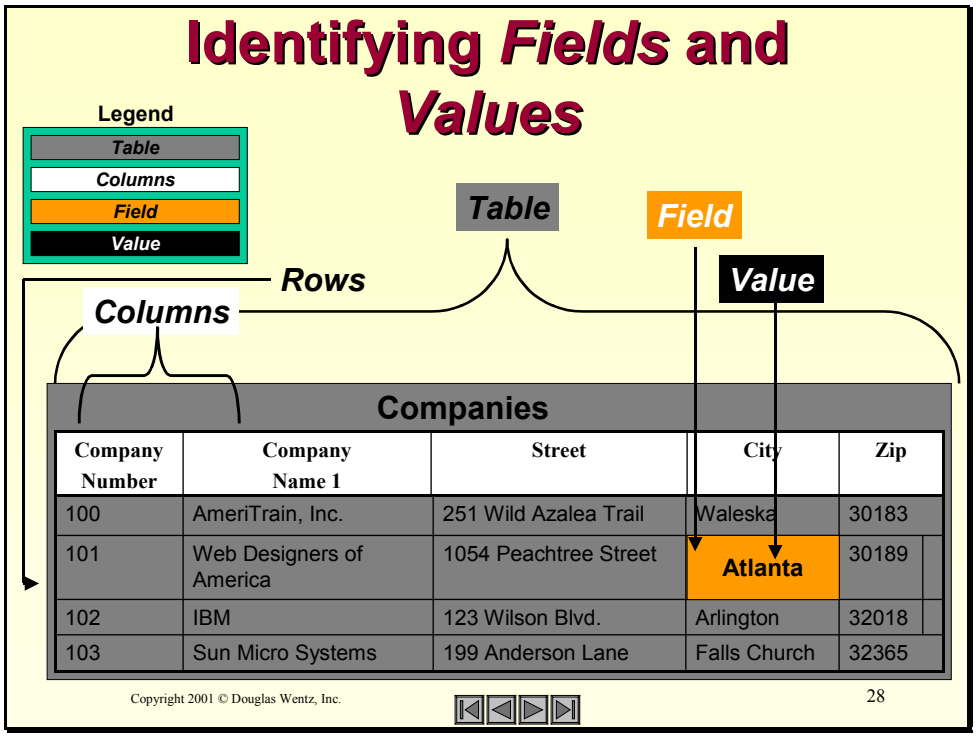
Attribute	Column
Product Number	CompanyNumber
Product Description	ProductDescription
Product Cost	ProductCost
Product Sell Price	ProductSellPrice
Product Status	ProductStatus

Attributes For Entity Order Becomes Columns For Table Orders

Attribute	Column
Order Number	OrderNumber
Customer Number	CustomerNumber
Order Date	OrderDate
Promised Ship Date	PromisedShipDate
Actual Ship Date	ActualShipDate

Attributes For Entity Line Item Becomes Columns For Table LineItems

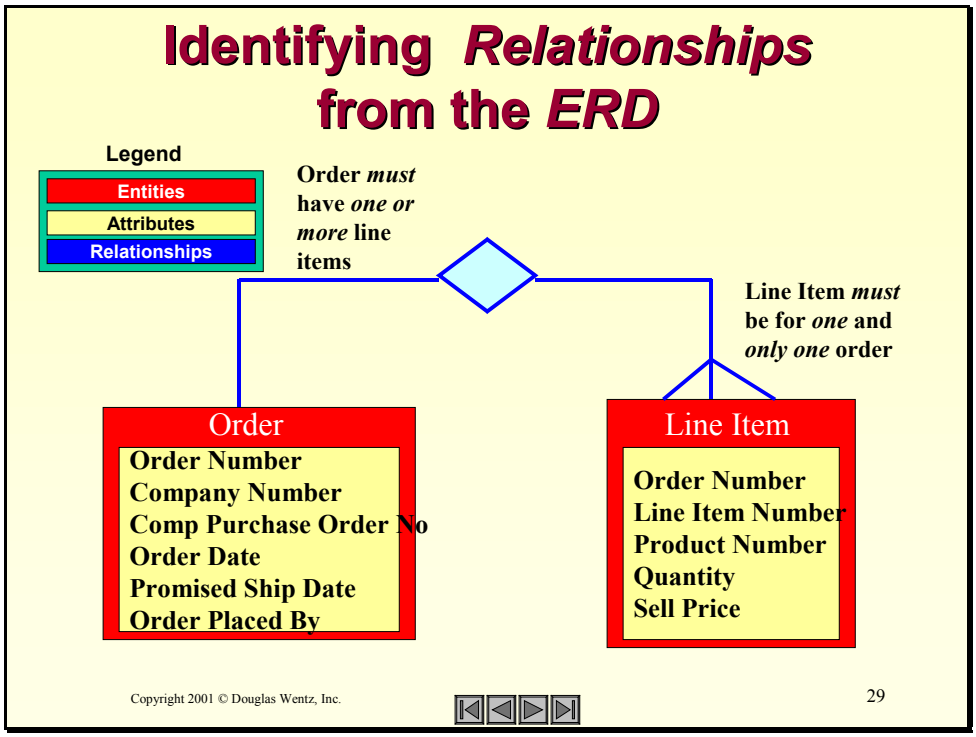
Attribute	Column
Order Number	OrderNumber
Line Item Number	LineItemNumber
Product Number	ProductNumber
Quantity	Quantity
Sell Price	SellPrice



- A **Row** is the horizontal aspect of a **Table** and is synonymous to a record in a traditional file system.
    - A **Row** typically has one value stored for each **Column** in the **Table**.
    - A **Row** may provide a list of values for a single **Column** or no values at all.
- A **Field** represents the intersection of a Column and a **Row**. *A* **Field** typically contains one value.
    - If the **Column** was defined as an array, the **Field** contains a list of values for one **Row**.
    - A **Value** is the actual piece of data stored in a **Field**.
    - If the **Field** contains no data it is considered as being NULL. Null means the **Field** contains no data. Do not confuse the term NULL with a field that contains spaces. They are different. NULL fields require no space for storage and fields that contain spaces consumes space.

Example rows of data contained in the table Companies

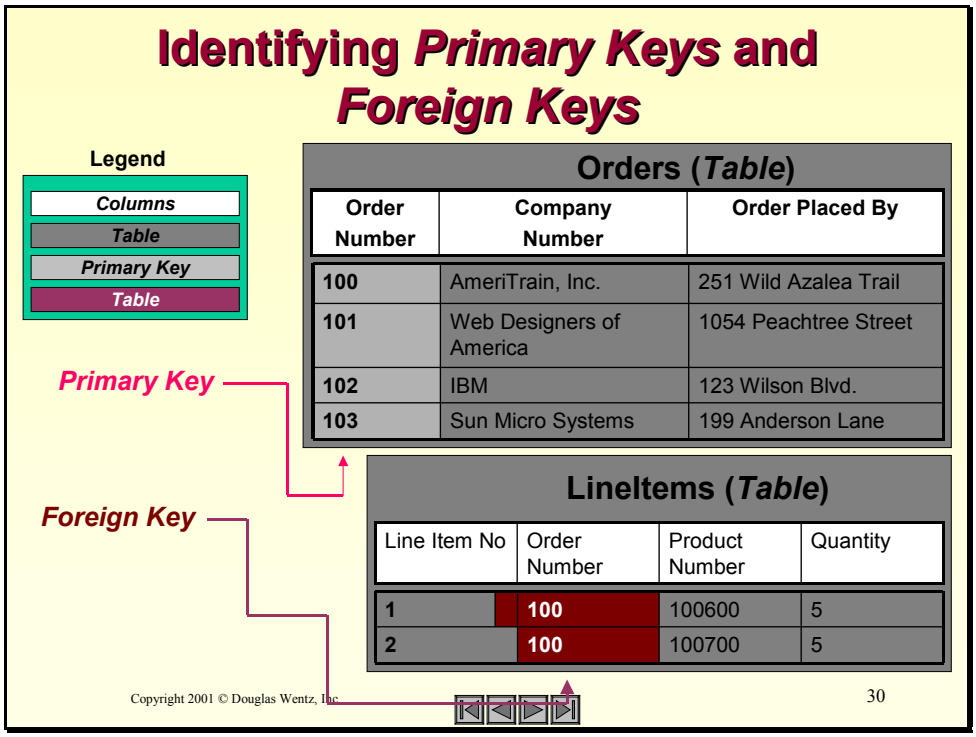
CompayNumber	CompanyName	Street	City	Zip
101	World Wide Consulting Servies	1074 Peachtree Walk	Atlanta	30054
102	Atlanta Training Services	1000 West Paces Ferry Road	Atlanta	30058
103	Global Consulting Servies	100 Peachtree Center	Atlanta	30051
104	Douglas Wentz	351 Wild Azalea Trail	Waleska	30183
105	United Training Services	251 Birchwood Court	Atlanta	30004
106	Global Business Solutions	458 Peachtree Road	Atlanta	30058
107	Oracle Consulting Services	104 High Tech Parkway	Atlanta	30054
108	South East Consulting Servies	958 Pine Walk Circle	Atlanta	30054
109	IBM	1071 Peachtree Street	Atlanta	30054
110	Sun Micro Systems	3 Peachtree Center	Atlanta	30054
112	Skill Builders	24 Salt Pond Road Unit C-4	South Kingston	2880
113	Global Business Solutions	2400 West Michigan Ave. Suite 4	Pensacola	32526
114	Contemporary Technologies, Inc.	444 Liberty Avenue	Pittsburgh	15222
115	IT Courseware	7245 S. Havana St. Suite 100	Englewood	80112
116	Applied Digital Solutions Inc.	400 Royal Palm Way, Suite 410	Palm Beach	33480
117	Bright Computer Training	2111 South Collins Street, Suite 201	Arlington	76010
118	Sideris Consulting Group, Inc.	214 North Main Street	Natict	1780
119	Trubix, Inc.	PO Box 2235	Littleton	80161
120	Animated Learning, Inc.	P. O. Box 1607	Nipomo	93444



- The above **one to many** relationship enforces referential integrity.
- Referential integrity will prevent the deletion of rows in the Orders table if an associated line item exists in the LineItems table. If a customer placed an order we do not want to delete an order if it has products ordered in the LineItems table.
- The following assumptions can be readily made;
    - The solid line from the Order **Entity** towards the Line Item **Entity** and the crow's feet indicates that an Order may have many line items. An Order may have more than one line item. It also indicates that a Order must have a **Line Item**.
    - *T*he solid line form the Line Item **Entity** to the Order **Entity** indicates that a Line Item must be for one and only one Order. A Line Item cannot be for more than one Order. This would not make good business sense.

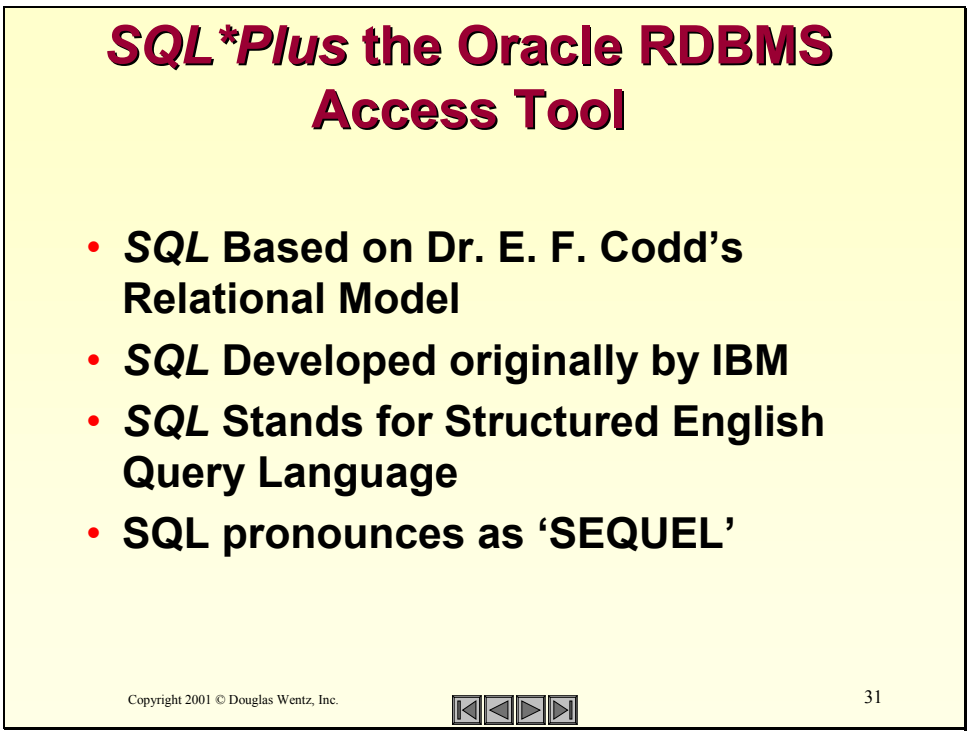
Note: Similar assumptions can be made from the rest of the **Entities** on the ERD.

Note: It is possible to have look up tables that are not related to any other tables. An example of a look up table would be the

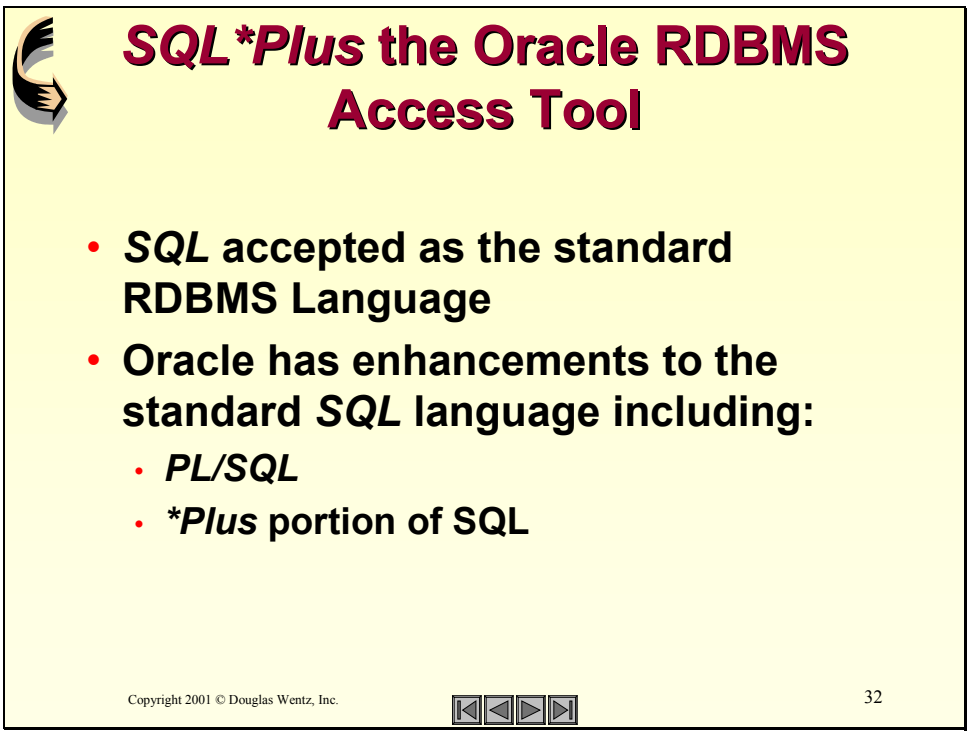


- A **primary key** uniquely identifies a **row** within a **table**.
  - A **primary key** is defined as one or more **columns** of a **table** that are used to make each **row** unique.
  - Each **table** can only have one **primary key** defined.
  - A **primary key** is not required however, it is usually created.
- A **foreign key** identifies a relationship between two **tables** or a table and its self.
  - A **foreign key** is defined as one or more **columns** of the **table** that are used to create the link.
  - A **foreign key** typically relates to the **primary key** of the referenced table.
  - A **table** can have many **foreign keys** defined for it.

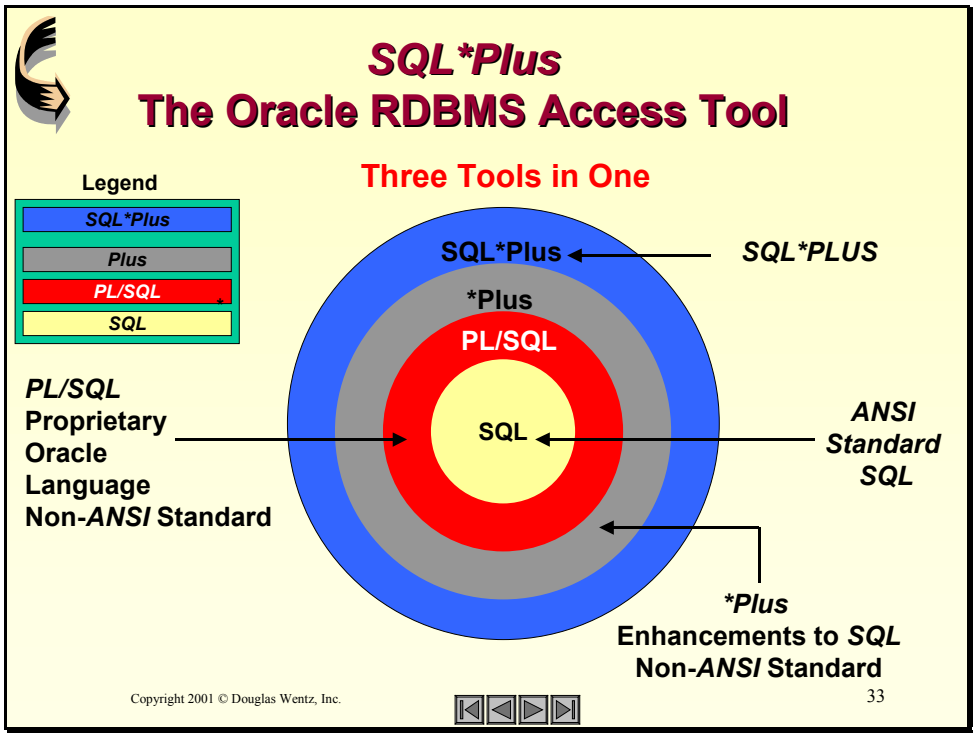
Note: In the example above the LineItems table can also have a primary key. It would be the combination of the LineItemNumber column and the OrderNumber column. This would make it unique and is called a composite key.



- SQL is a language used to access relational databases.
- Dr. E. F. Codd published the paper, "A Relational Model of Data for Large Shared Data Banks", in June 1970 in the Association of Computer Machinery (ACM) journal, *Communications of the ACM*.
- Codd's model is now accepted as the definitive model for relational database management systems (RDBMS).
- The SQL language was originally developed by IBM. The SQL language was developed to use Dr. E. F. Codd's relational model.
- In 1979, Relational Software, Inc. (now Oracle Corporation) introduced the first commercially available implementation of SQL. SQL stands for Structured English Query Language.
- SQL is pronounced as "SEQUEL"



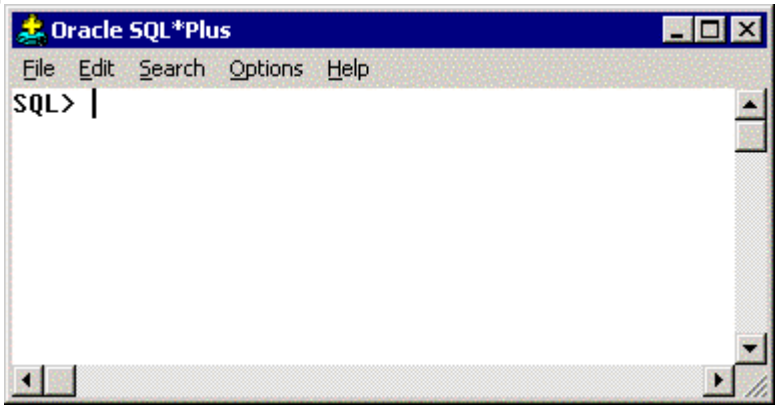
- SQL is accepted as the standard RDBMS access language.
- SQL can be used to access many relational databases including Microsoft Access, and SQL Server.
- SQL provides benefits for all types of users, including application programmers, database administrators, managers, and end-users.
- Oracle has enhancements to the basic SQL engine. These enhancement include the following;
    - PL/SQL is the procedural portion. It combines SQL and also a procedural language. It utilizes the best of both languages as we will see latter. It is incorporated into the SQL engine
    - *Plus is report formatting commands, environment commands, etc. It is also incorporated into the SQL engine.
- The first commercially available SQL relational database management system was introduced in 1979 by Oracle Corporation.

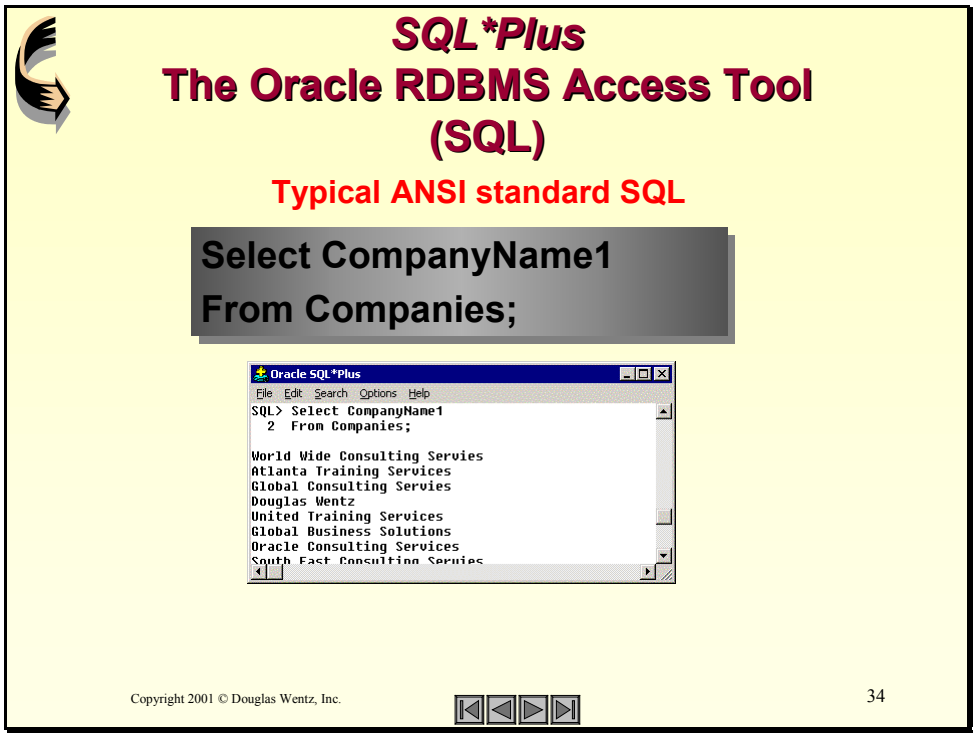


- Oracle's SQL*Plus tool is actually three tools in one. These tools include the following;
  - SQL
  - *Plus
  - PL/SQL

Note: Application programs and Oracle tools often allow users access to the database without using SQL directly, but these applications in turn must use SQL whenever executing the user's request to the Oracle database.

Example of SQL*Plus

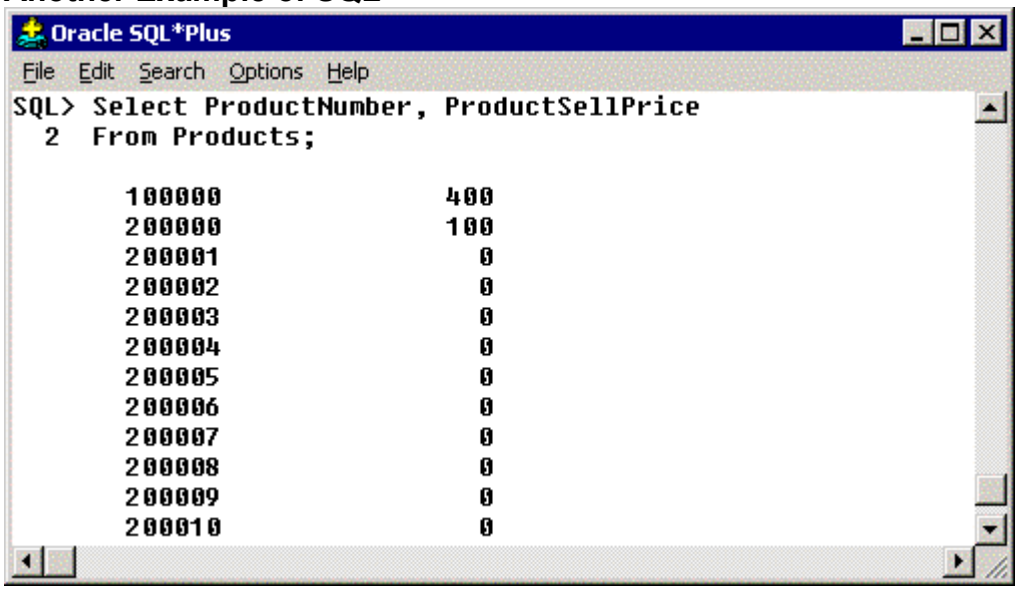


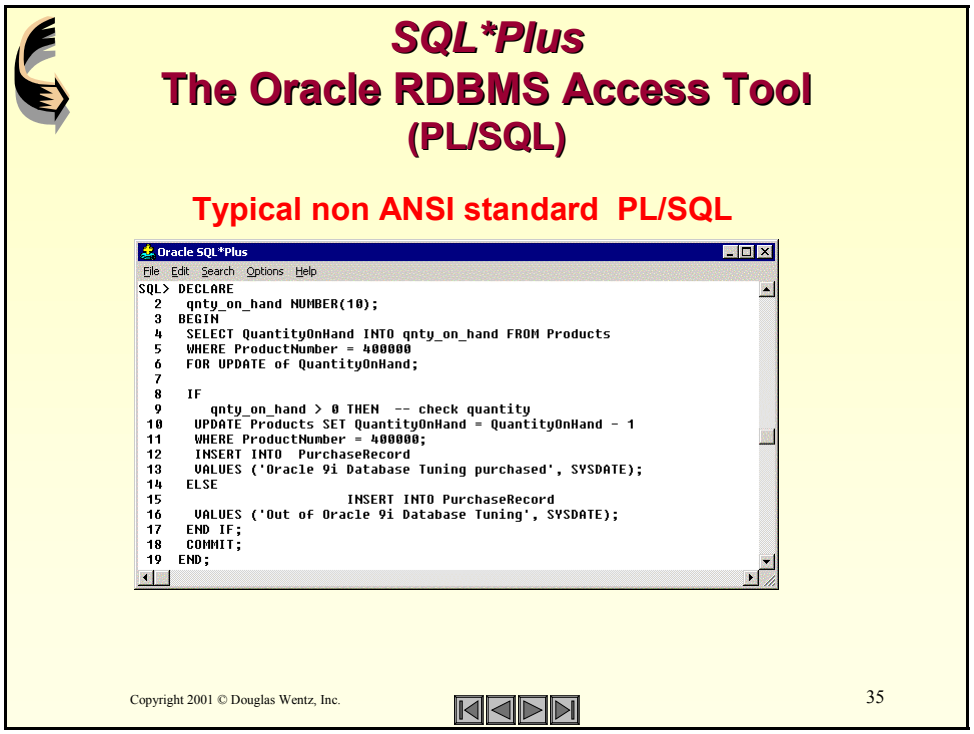


- **SQL** is the standard language to access relational databases.
- **SQL** is an English like command level language providing a way for users to interact with any relational database. These may be relational databases other than those provided by Oracle. **SQL** can;
    - Create database objects such as tables, views, synonyms and constraints.
    - Query database objects such as tables.
    - Manipulate database objects such as inserting values into tables.
    - Edit scripts.
- Oracle conforms to the SQL standards established by these organizations:
    - American National Standards Institute (ANSI)
    - International Standards Organization (ISO)
    - United States Government

Example of SQL

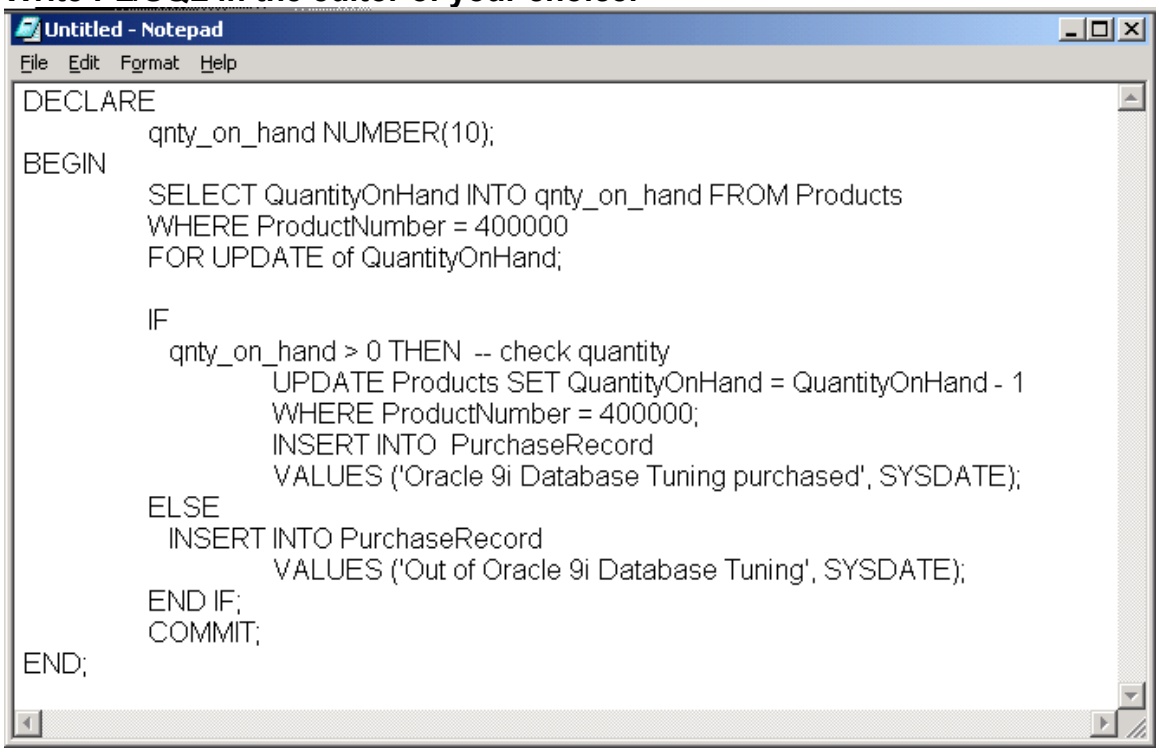
Another Example of SQL



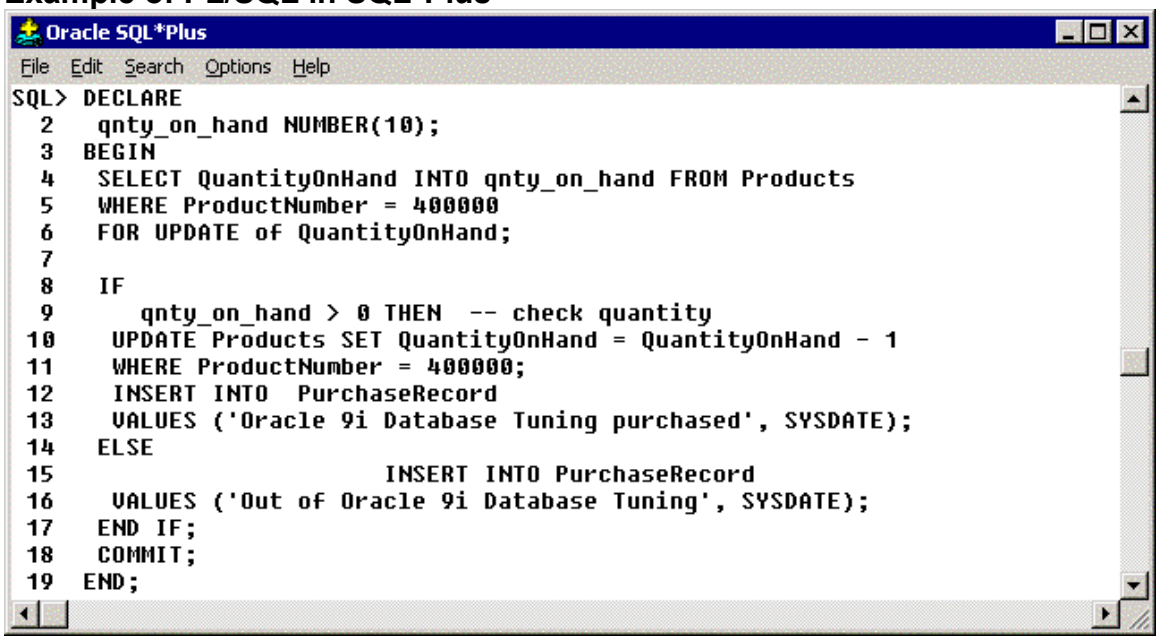


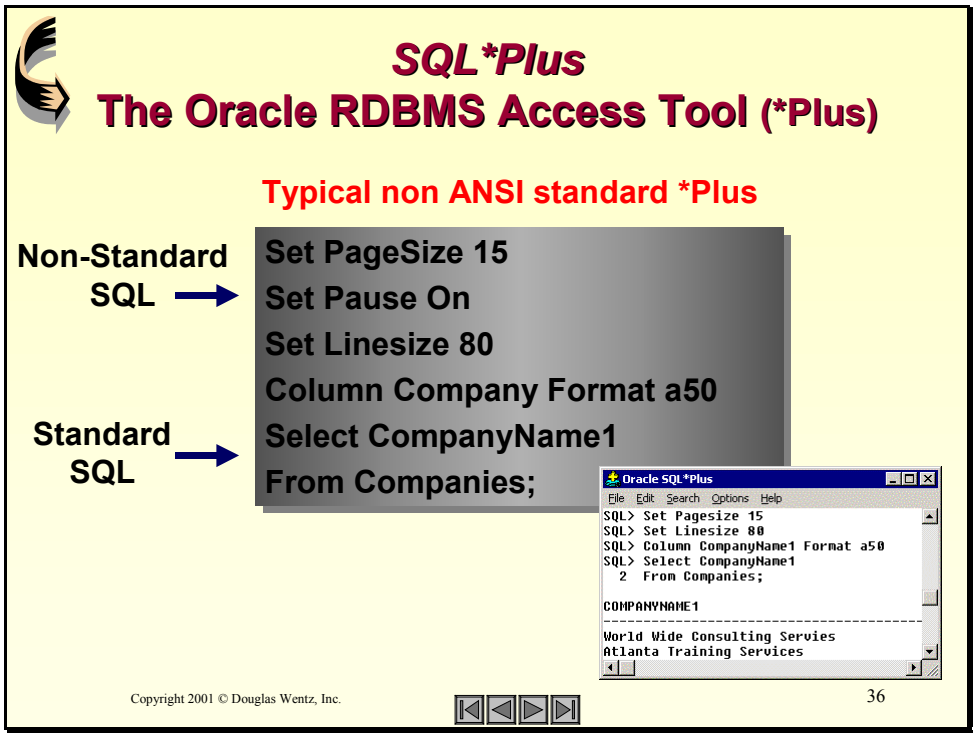
- **PL/SQL** is a non-ANSI standard procedural language. PL/SQL bridges the gap between relational database technology and procedural programming languages.
- With **PL/SQL**, you can use SQL statements to manipulate Oracle data and flow-of-control statements to process the data.
- Moreover, you can declare constants and variables, define procedures and functions, and trap runtime errors.
- Thus, PL/SQL combines the data manipulating power of SQL with the data processing power of procedural languages.
- Constructs within **PL/SQL** are similar to those found in a 3GL language.
- PL/SQL is executed using the SQL*Plus tool provided by Oracle.
- Generally, PL/SQL program code is created using a text editor and then executed using SQL*Plus or iSQL*Plus.

Write PL/SQL in the editor of your choice.



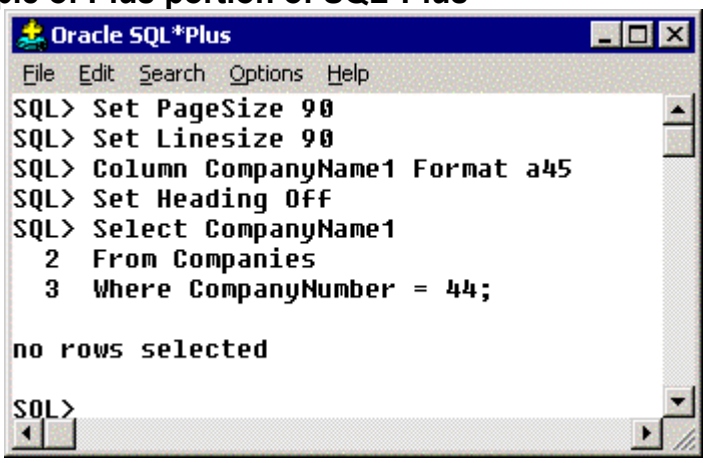
Example of PL/SQL in SQL*Plus

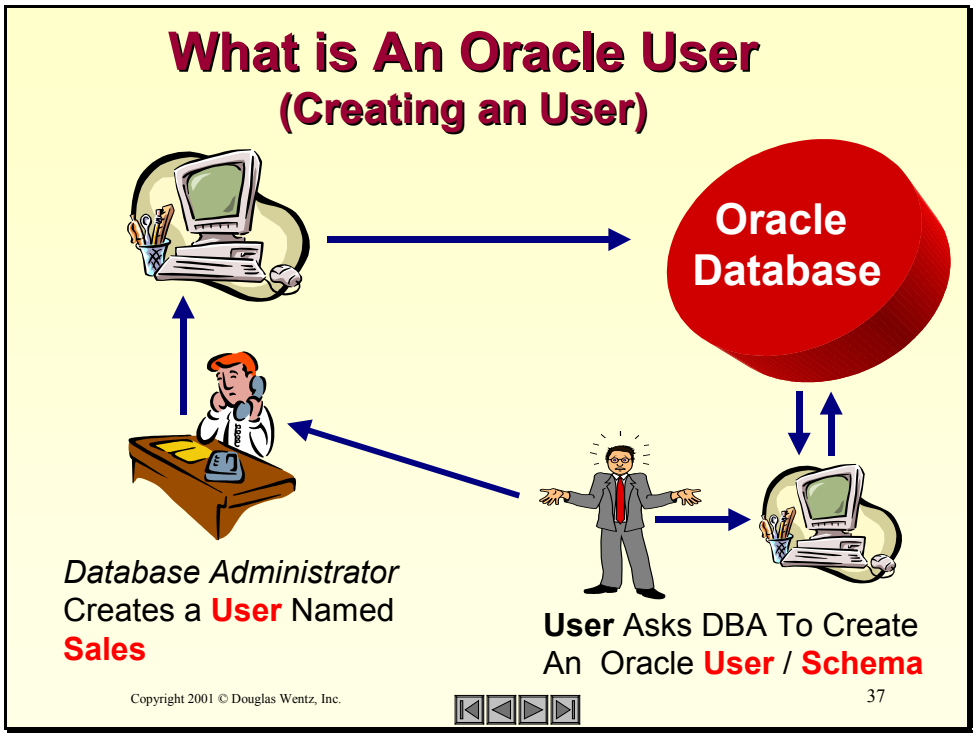




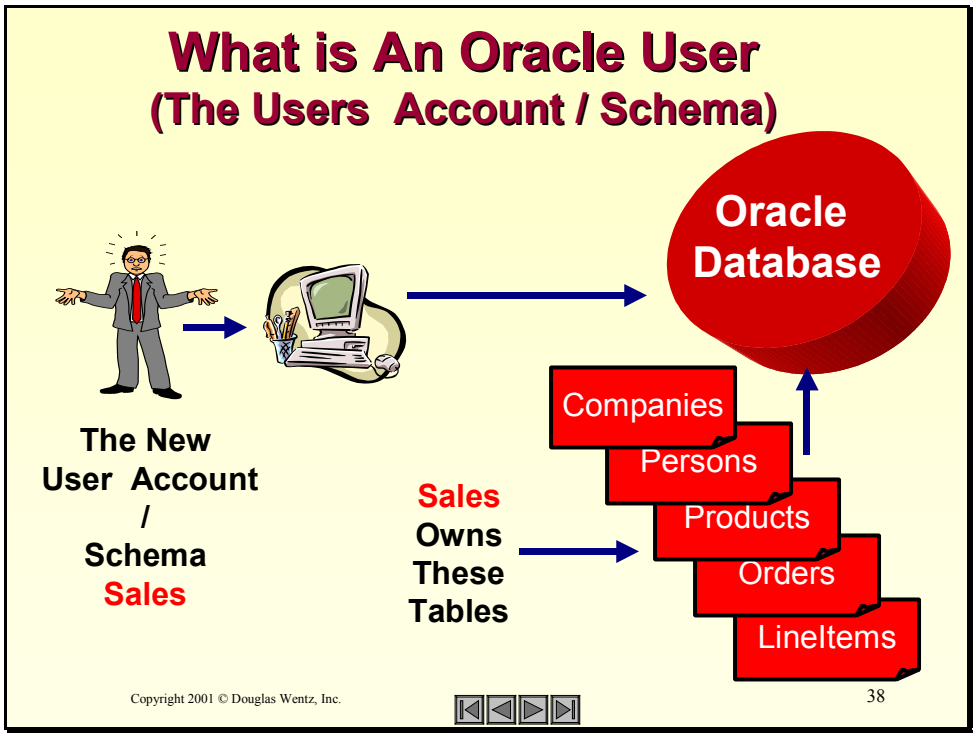
- ***Plus** – Is an non-ANSI standard report formatting and additional features language to enhance the Oracle SQL Engine. **The Plus portion of SQL*Plus** will be addressed in subsequent modules within this class. The additional features of the ***Plus** portion of **SQL*Plus** include;
    - Format Reports
    - Control the users environment
    - Computed values
    - Interact with user

Example of Plus portion of SQL*Plus

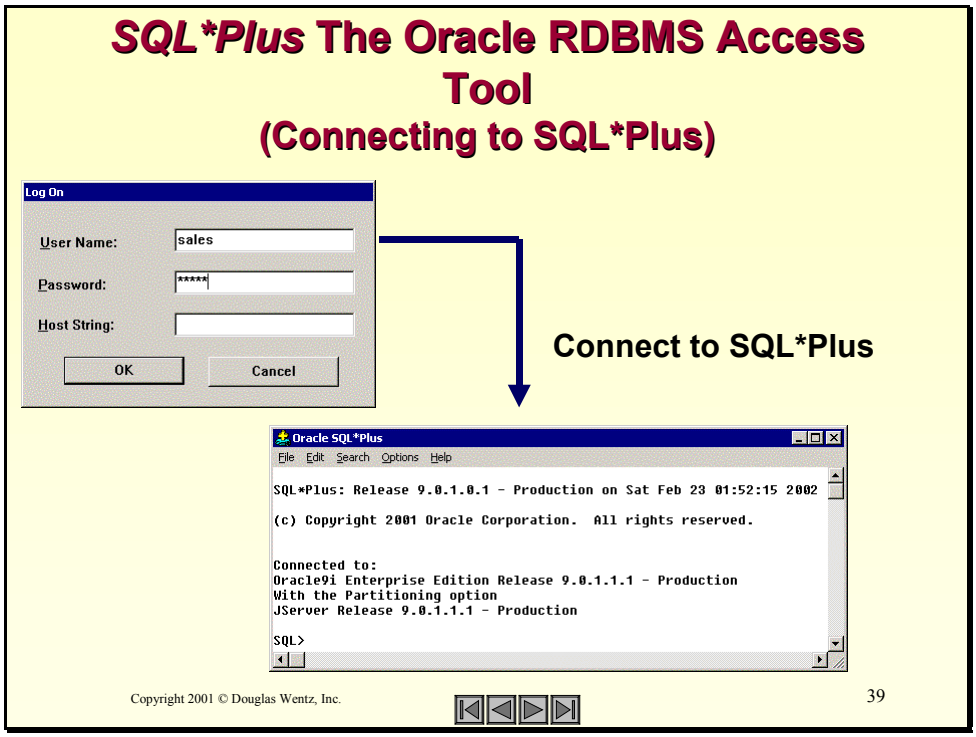




- Prior to accessing an Oracle database or accessing **SQL*Plus** an Oracle User Account will need to be created for you or an existing User Account assigned. In most instances these will be create for you by your DBA. To access the Oracle User Account the following will be required;
  - A User Account name
  - A User Password
- Whenever the DBA creates your User Account he/she will assign certain privileges to the User Account . These privileges may include;
  - Ability to create tables
  - Ability to create other users
  - Ability to execute procedures
  - Ability to create indexes
- The above list of privileges is only a partial list and more exists.



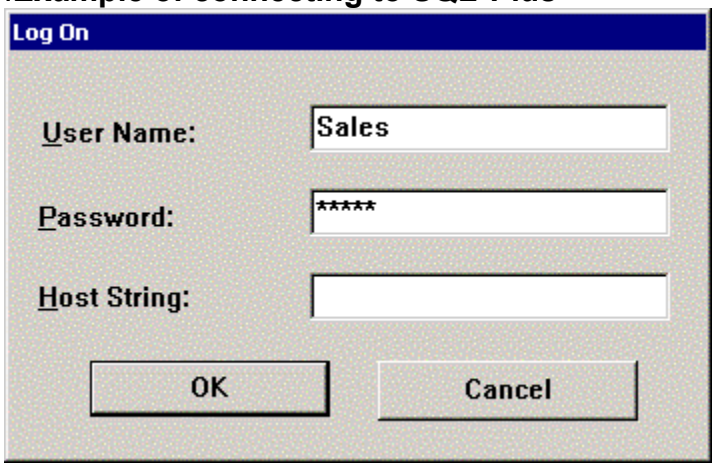
- The term logy User and Schema are used interchangeably. A Schema is owned by the database User and has the same name as the User. Each user owns a single User Account / Schema.
- When the DBA creates an User Account a Schema is created automatically with the same name as the User Account.
- Whenever objects are created under a certain User account / Schema that **User** account / Schema owns those objects created.
- From the above illustration the User Account **Sales** was created.
- Latter in this class we will see how to create the tables illustrated above. These tables are owned by the User Account / Schema of **Sales**.
- In most of the labs within this class will require access to the **Sales** Account / Schema.



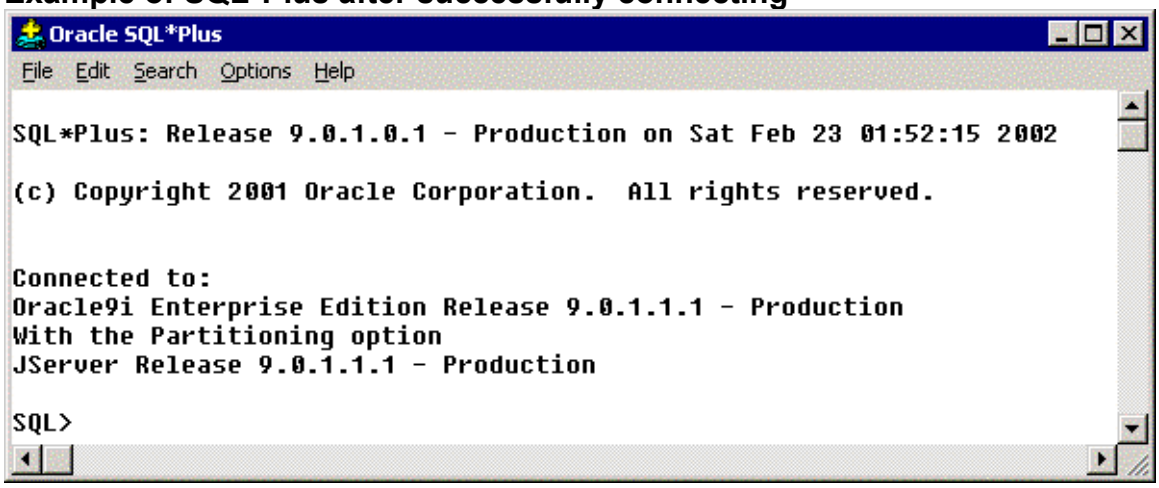
- **SQL*Plus** can be invoked in many different ways depending on your computer platform. If you are in a Microsoft environment look for the **SQL*Plus** icon or **SQL*Plus** in the START menu.
- If you are in the UNIX environment you will need to go the ORACLE_HOME/Bin directory and execute the appropriate **SQL*Plus** executable. The name of this executable will be different depending again on the computer platform.
- No matter what platform you will get the familiar login screen as illustrated above including the following;
    - The Oracle User Account assigned
    - The User Password assigned
    - The Host String will be covered in the Networking class
- After supplying the correct User Account and User Password the familiar SQL*Plus screen should appear.
- Oracle will show what release of Oracle that is currently being used and options selected during the installation process of Oracle by your database administrator or system administrator.

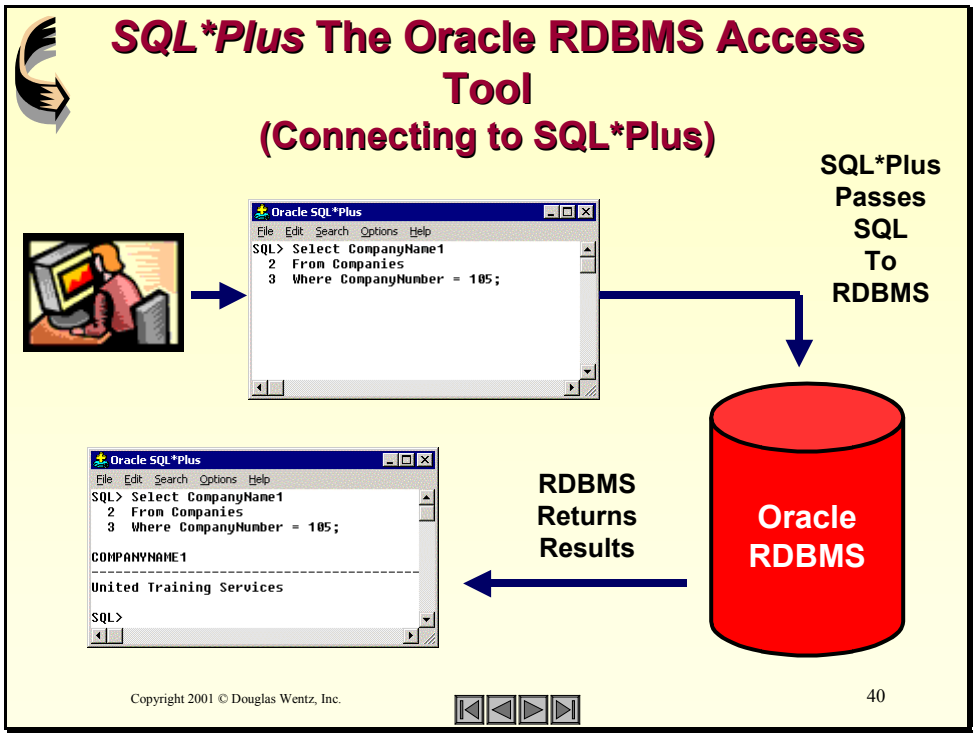
Note: The Host String / Connect String are the same. Both provide a means to connect to a remote Oracle database. These will be addressed in the Networking class

.**Example of connecting to SQL*Plus**



Example of SQL*Plus after successfully connecting



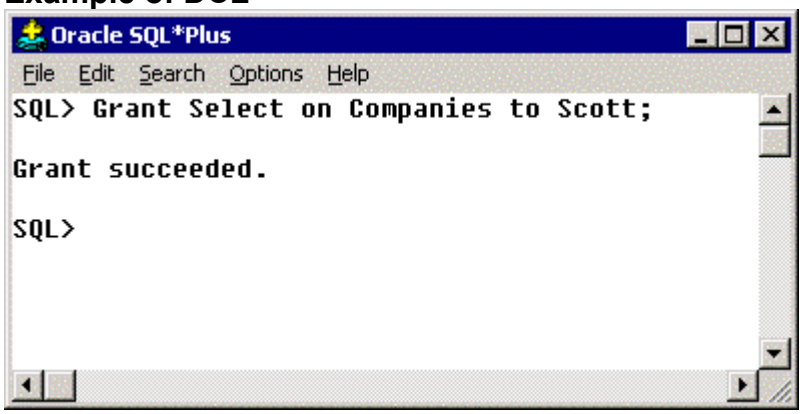


- The SQL commands can be grouped into the following categories;
  - Data Control Language (DCL)
  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)
  - Transaction Control Language (TCL)
  - Query
- The above is an example of a Query.
- The column named CompanyName1 is retrieved from the Companies table where the CompanyNumber equals 105.
- Details of each of the categories of SQL commands are outlined on the next page including examples.

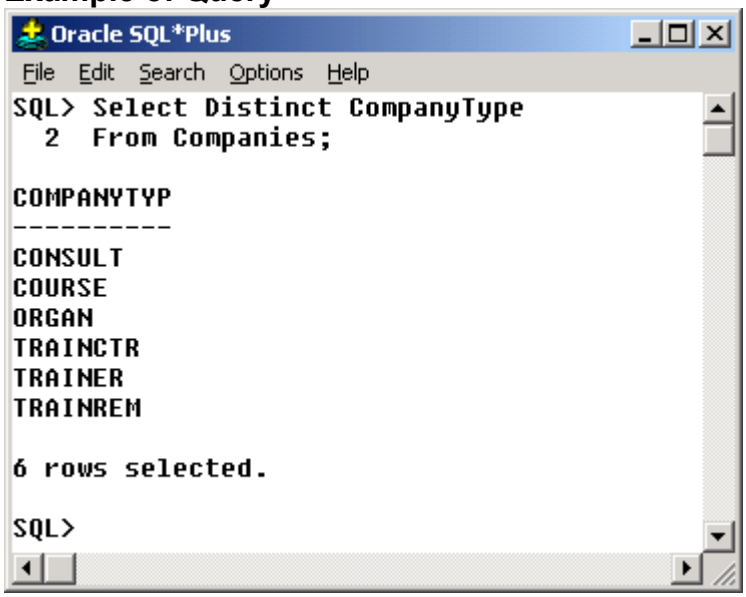
Categories of SQL Statements

Category	Description
DCL	SQL commands give the user the ability to perform certain database operations.
DDL	SQL commands allows the creating, alteration, and dropping of objects from the database.
DML	SQL commands allow the inserting, updating, and deletion of objects from the database.
TCL	SQL commands allow certain DML commands to be undone.
Query	SQL commands allow queries to be performed on the objects in the database.

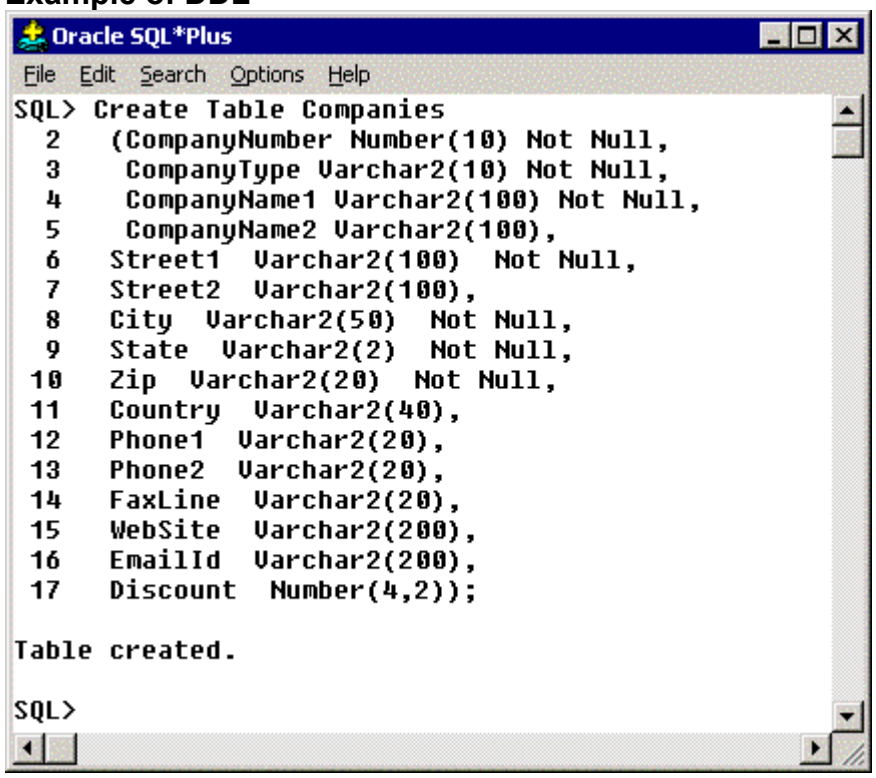
Example of DCL



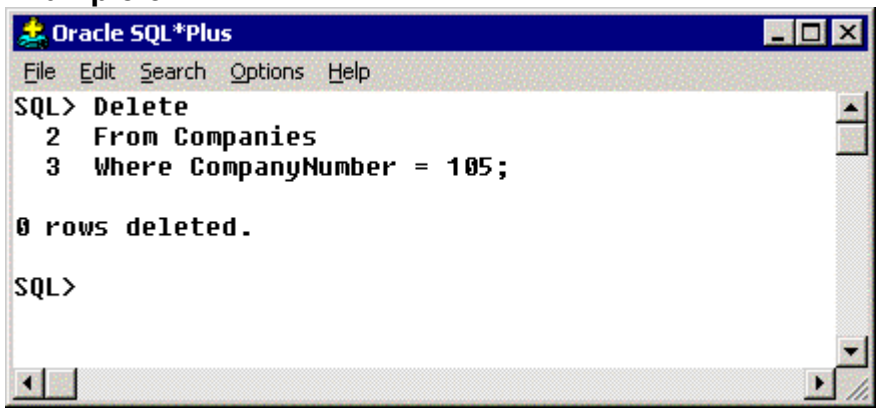
Example of Query



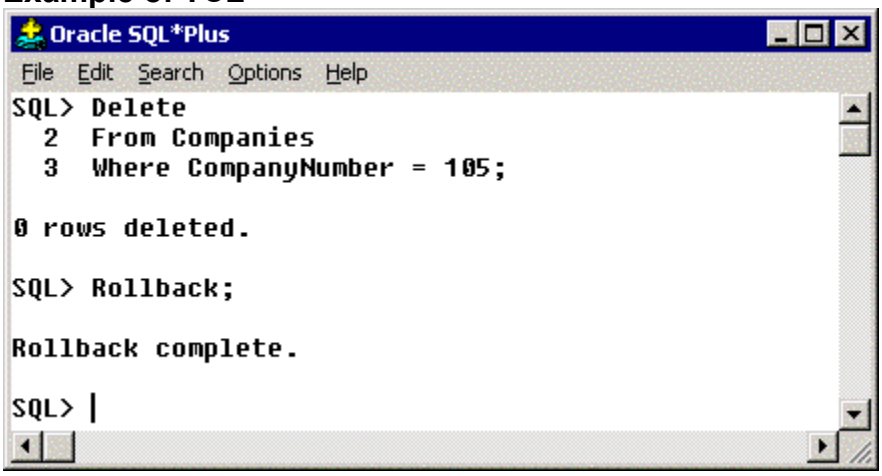
Example of DDL



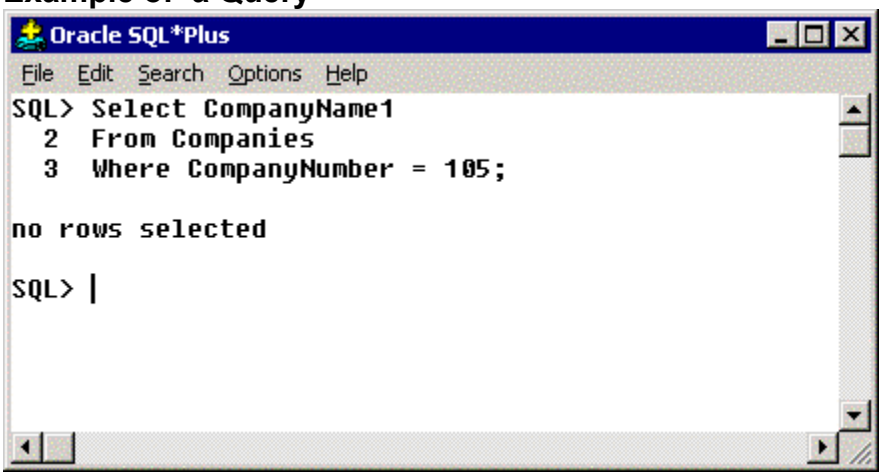
Example of DML



Example of TCL



Example of a Query



Note: Have your instructor access SQL*Plus by connecting to the account SCOTT with a password of TIGER and showing the tables contained in the SCOTT account. Perform a describe on each of the tables and select some data from each of the tables.

Oracle Certified Professional Test Questions

1. Entities generally map to?

    a. Columns
    b. Tables
    c. Attributes
    d. Fields
    e. Values
    f. Rows

2. Attributes generally map to what?

    a. Columns
    b. Tables
    c. Attributes
    d. Fields
    e. Values
    f. Rows

3. A column in a table that contains a picture would be assigned what data type?

    a. DATA
    b. NUMBER
    c. CHAR
    d. VARCHAR
    e. BLOB

4. A column in a table that contains a persons Social Security Number would be assigned what data type?

    a. DATA
    b. NUMBER
    c. CHAR
    d. VARCHAR2
    e. BLOB

5. A table can have only one what?

    a. Foreign Key
    b. Column
    c. Field
    d. Primary Key

e. Attribute

6. Most relational databases are in what normal form ?

   a. First Normal Form
   b. Second Normal Form
   c. Third Normal Form
   d. Fourth Normal Form

7. To graphically illustrate the entities, attributes and relationships of a Business System what would you create?

   a. Table
   b. Attribute
   c. Entity
   d. Entity Relationship Diagram
   e. Data Flow Diagram

8. The ANSI standard part of SQL*Plus is what tool?

   a. PL/SQL
   b. *Plus
   c. SQL
   d. VI
   e. NotePad

9. To connect to SQL*Plus you will need a?

   a. User Id & Password
   b. User Id & Resource Locator
   c. User Id & Connect String
   d. User Id & Host String

10. In our business model we have 5?

    a. Tables
    b. Columns
    c. Attributes
    d. Data Types

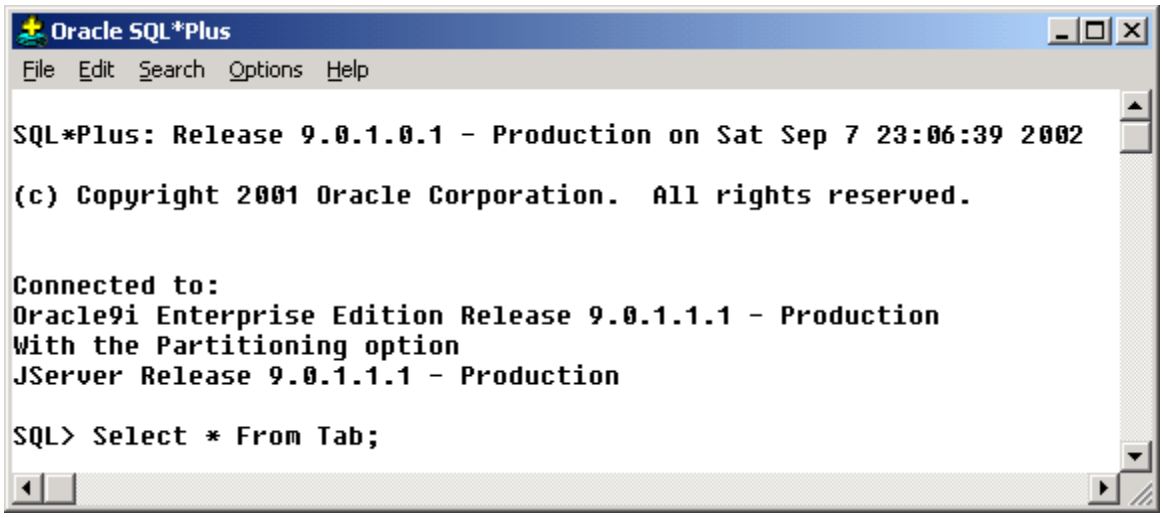
Lab Exercises

1. Review the script DBAOCPInstall.sql in your favorite editor to become familiar with the syntax of creating tables, inserting values into tables, creating the user SALES , etc. This script can be found in DBAOCP\Install\DBAOCPInstall.sql

2. Find SQL*Plus and connect to the user account SYSTEM with a password of MANAGER.

3. Perform the following query.

   Select * From Tab;



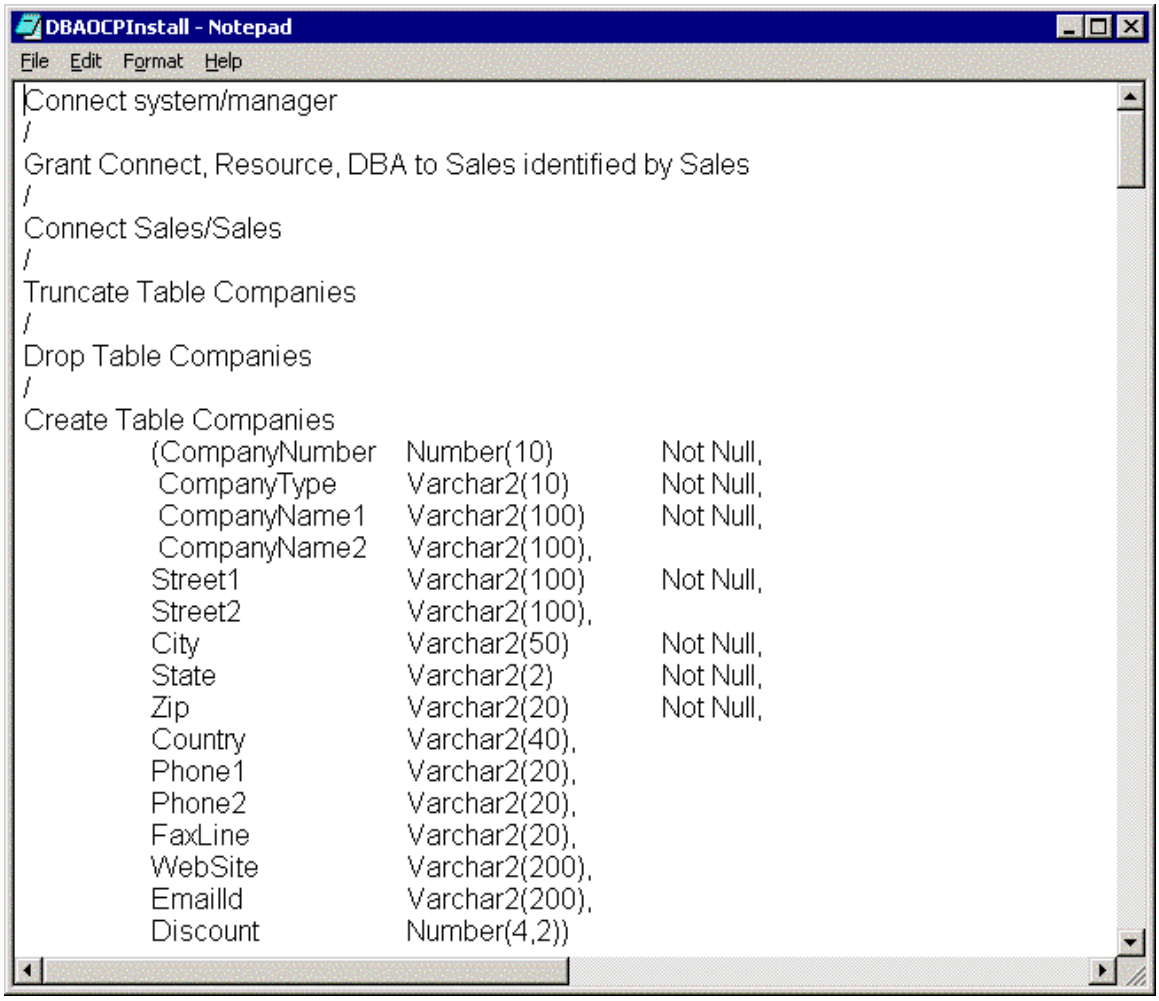
4. Connect to the SCOTT account with the password of TIGER in SQL*Plus. Perform the following query "Select * From Tab;" to retrieve a list of all of the tables in the SCOTT account. Perform a describe on each of the tables by performing the following query "Describe Emp". Perform this for each of the tables in the SCOTT account. Exit from SQL*Plus by typing "EXIT" at the command prompt.

Oracle Certified Professional Test Questions – Answers

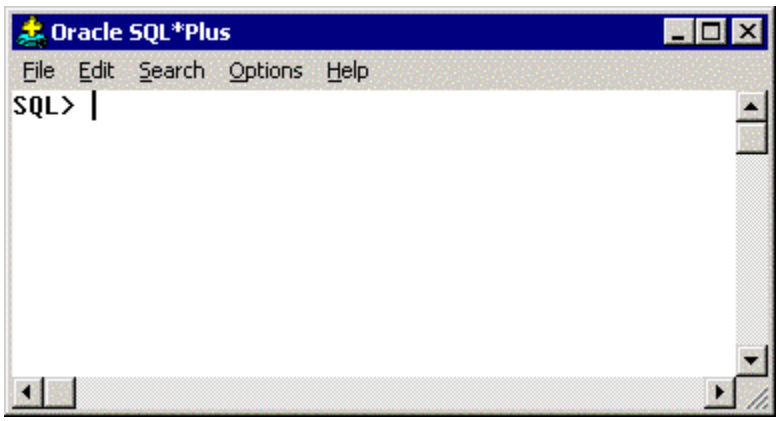
1. B
2. A
3. E
4. C
5. D
6. C
7. D
8. C
9. A
10. A

Lab Exercise - Answers

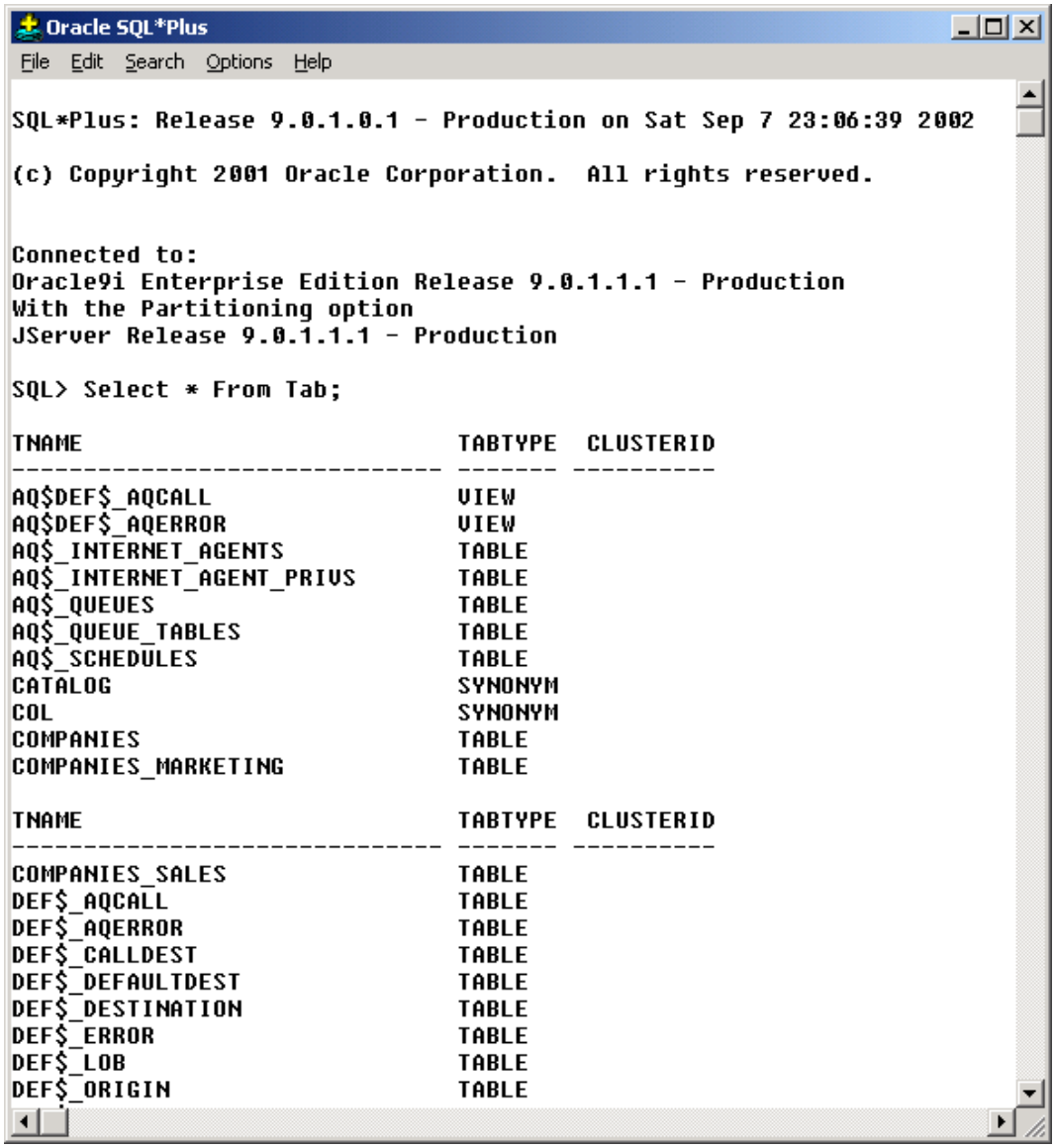
Lab exercise 1



Lab exercise 2



Lab Exercise 3



Lab Exercise 4