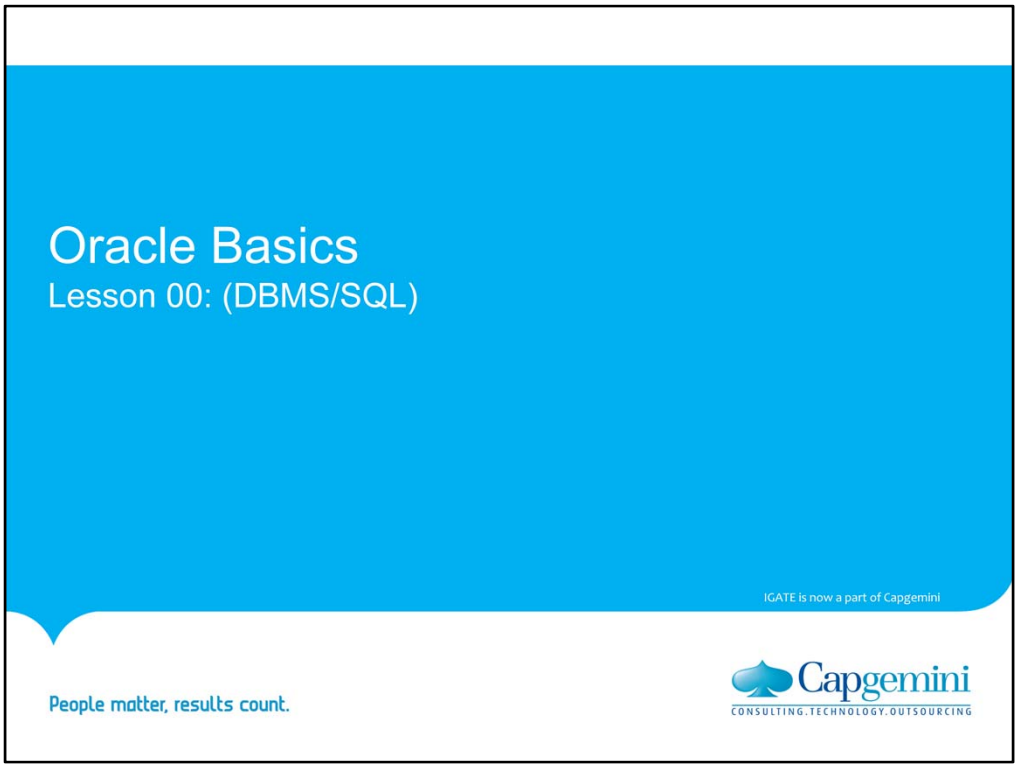


Document History

Date	Course	Software Version	Developer / SME	Change Record Remarks
	Version No.	No.		
13-Nov-08	1	9i	Rajita Dhumal	Content Creation
14-Nov-08	1.1	9i	CLS team	Review
14-Jan-10	1.2	9i	Anu Mitra	Review
14-Jan-10	1.2	9i	Rajita Dhumal, CLS Team	Incorporating Review comments
25-Apr-11	2	9i	Anu Mitra	Integration refinements
17-May-13	2.1	9i	Hareshkumar Chandiramani	Courseware refinements
21-Apr-15	2.2	9i	Kavita Arora	Courseware refinements

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

Course Goals and Non Goals

- Course Goals
  - To understand basic DBMS, and use SQL commands.
- Course Non Goals
  - Nothing Specific.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

Pre-requisites

- A proficiency level in familiarity with Windows.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Intended Audience

- Software Programmers
- Software Analysts

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

Day Wise Schedule

- Day 1
  - Lesson 1: Introduction to Database
  - Lesson 2: Basics of SQL
  - Lesson 3: Data Query Language
  - Lesson 4: Aggregate (Group) Functions
  - Lesson 5: SQL (Single-row) functions
  - Lesson 6: Joins
- Day 2
  - Lesson 6: (contd) Sub-queries
  - Lesson 7: Database Objects

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Day Wise Schedule

- Day 3
  - Lesson 8: Data Manipulation Language
  - Lesson 9: Transaction Control Language
  - Start with PL SQL

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

References

- RDBMS Concepts - A Primer
  - http://safari.oreilly.com
- Introduction to Database Systems; by C.J.Date
- Relational Database Theory; by Atzeni, De Antonellis

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Next Step Courses (if applicable)

- Oracle PL/SQL

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Other Parallel Technology Areas

- MS-Access
- SQL Server
- DB2

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

DBMS/SQL

Getting Started with Database

Lesson Objectives

- To understand the following topics:
  - Introduction to Database
  - Characteristics of DBMS
  - Data models
  - Relational DBMS
  - Database Administrator

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

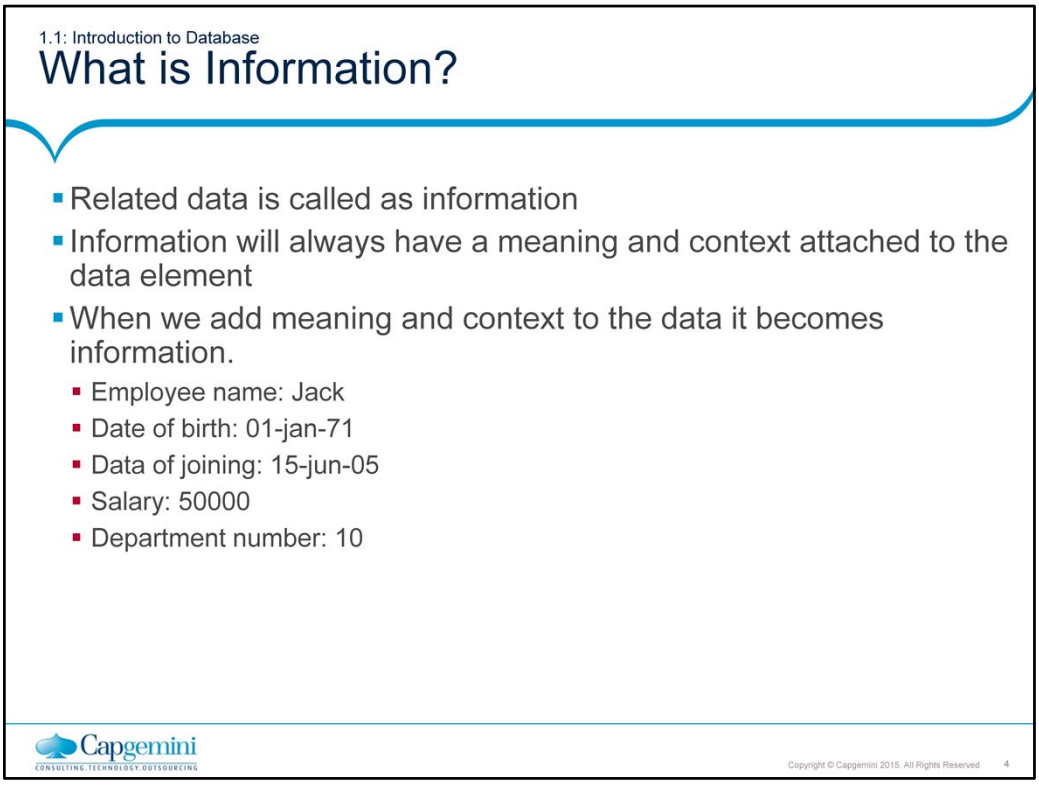
1.1: Introduction to Database

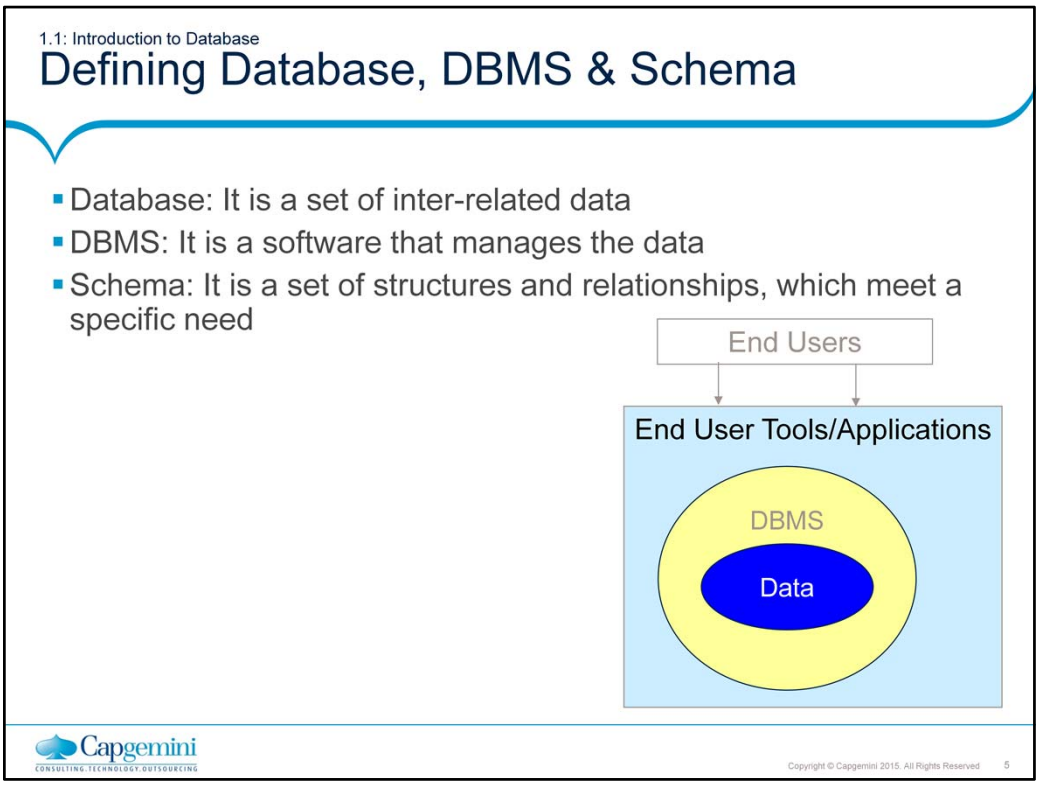
What is Data?

- Data (plural of the word datum) is a factual information used as a basis for reasoning, discussion, or calculation
- Data may be numerical data which may be integers or floating point numbers, and non-numerical data such as characters, date etc.
- Data by itself normally doesn't have a meaning associated with it.
- e.g:- Jack
  01-jan-71
  15-jun-05
  50000

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3


1.1: Introduction to Database
What is Information?
Related data is called as information
Information will always have a meaning and context attached to the data element
When we add meaning and context to the data it becomes information.
Employee name: Jack
Date of birth: 01-jan-71
Data of joining: 15-jun-05
Salary: 50000
Department number: 10
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 4

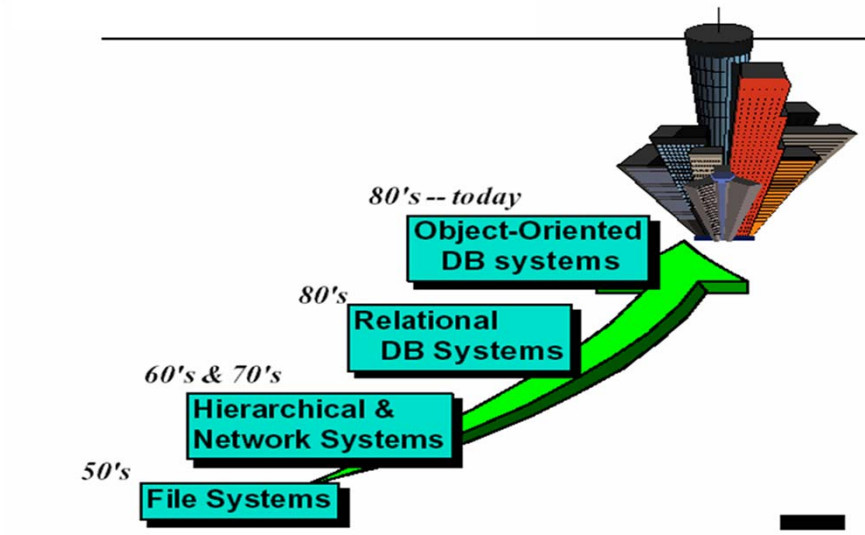


Introduction to Database:

- A logically coherent collection of related data (“information”) with inherent meaning, built for a certain application, and representing a subset of the "real-world". For eg: Customer database in bank, Employee Details
- The software that manages the database is known as “Database Management System” or “DBMS”. Hence DBMS can be described as "a computer-based record keeping system which consists of software for processing a collection of interrelated data". The general purpose of a DBMS is to provide for the definition, storage, and management of data in a centralized area that can be shared by many users
- A set of structures and relationships that meet a specific need is called as a “schema”.

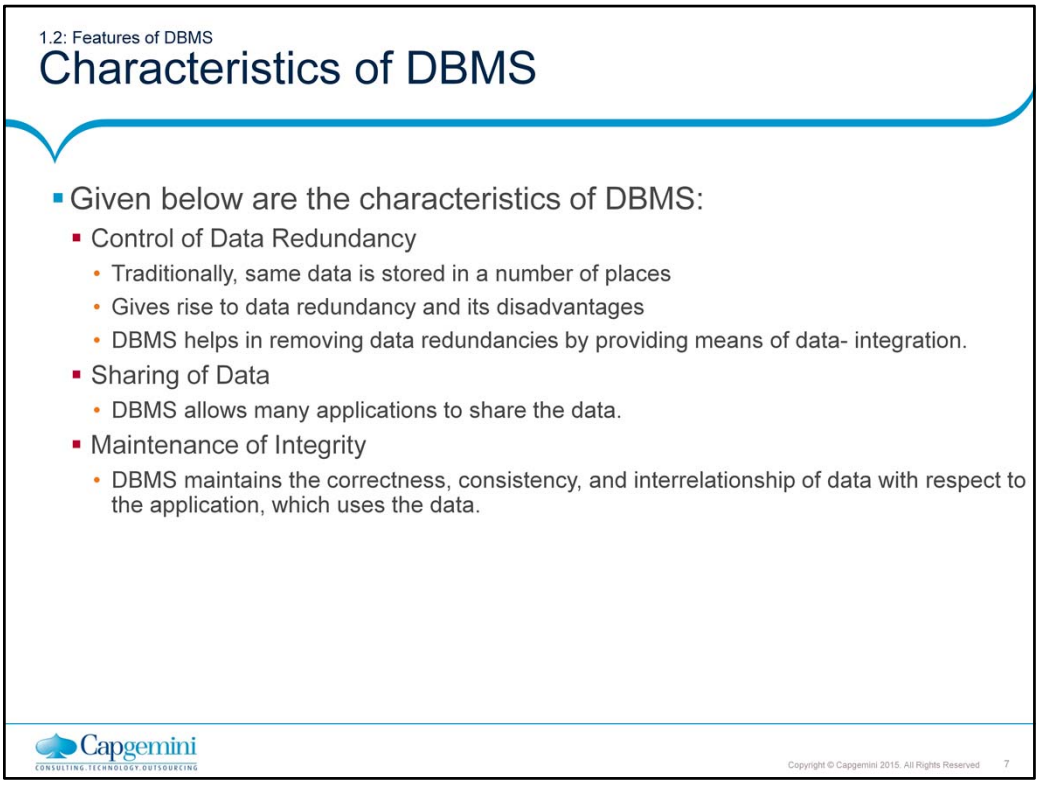
1.1: Introduction to Database

Evolution of Databases



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

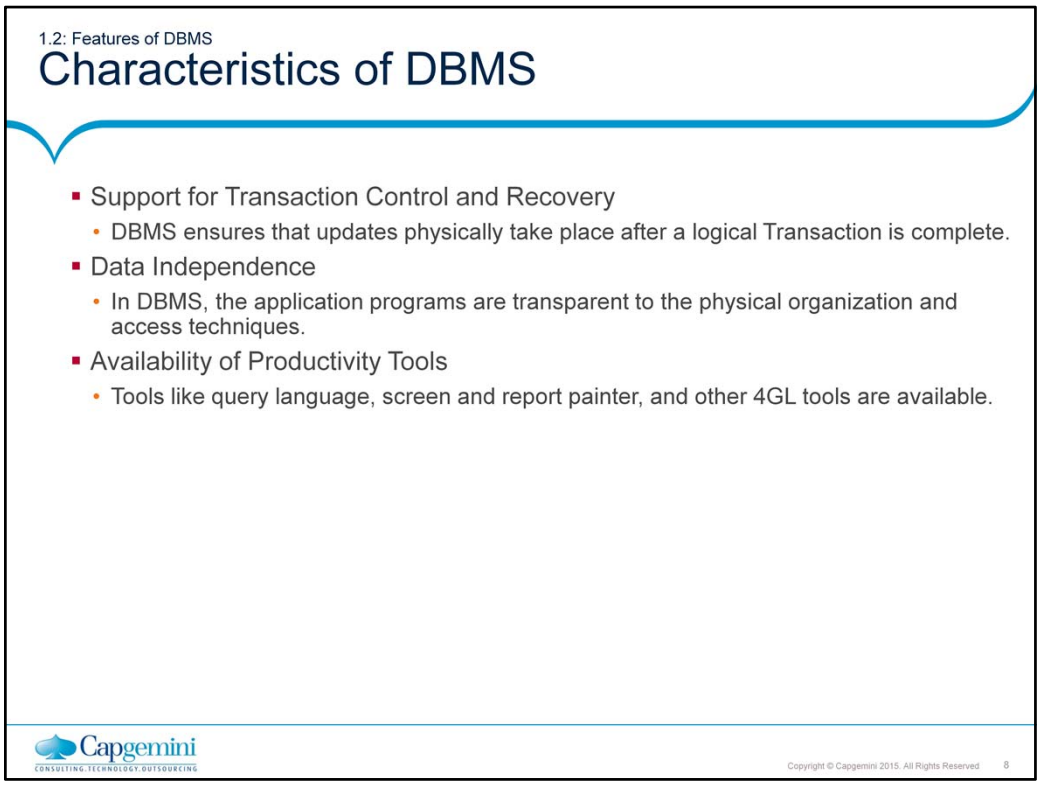


Characteristics of DBMS:

Some of the characteristics of the DBMS are given below:

- **Control of Data Redundancy**
  - When the same data is stored in a number of files, it results in data redundancy. In such cases, if the data is changed at one place, the change has to be duplicated in each of the files.
  - The main disadvantages of data redundancy are:
    - Storage space is wasted.
    - Processing time may be wasted as more data needs to be handled.
    - Inconsistencies may creep in.
  - DBMS helps in removing redundancies by providing means of integration.
- **Sharing of Data**
  - DBMS allows many applications to share the data.
- **Maintenance of Integrity**
  - Integrity of data refers to the correctness, consistency and interrelationship of data with respect to the application that uses the data. Some of the aspects of data integrity are:
    - Many data items can only take a restricted set of values.

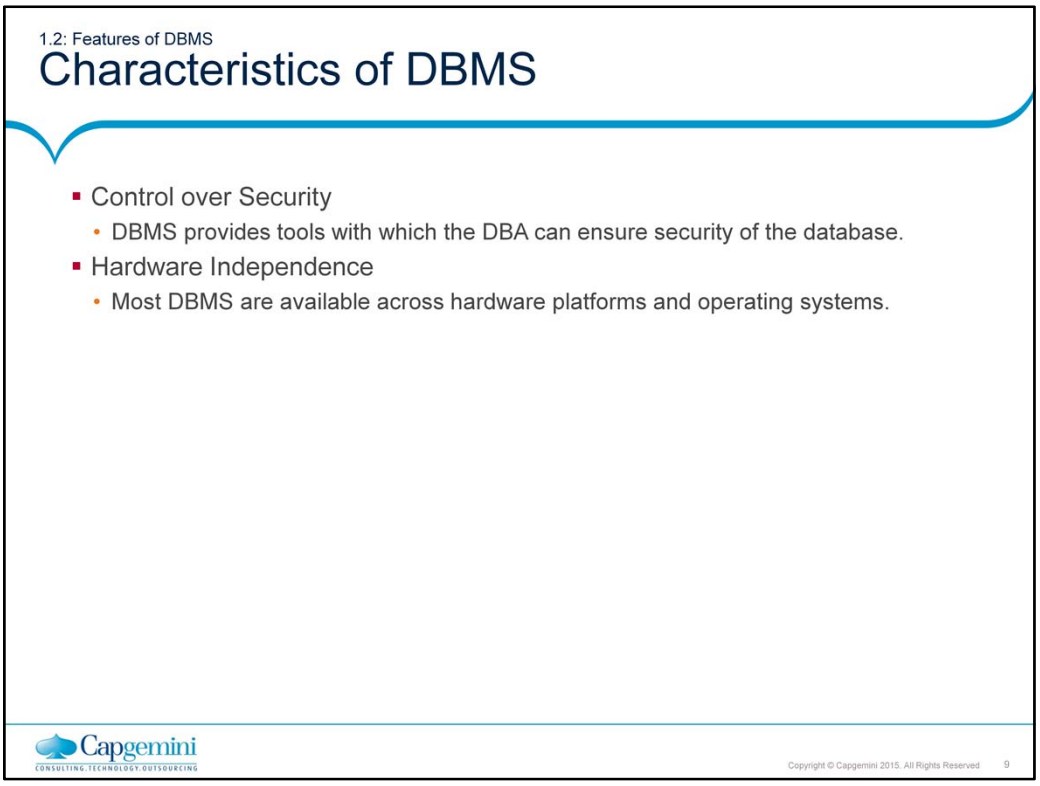
.



Characteristics of DBMS (contd.):

- Certain field values cannot be duplicated across records. Such restrictions, called primary key constraints, can be defined to the DBMS.
- Data integrity, which defines the relationships between different files, is called referential integrity rule, which can also be specified to the DBMS

- **Support for Transaction Control and Recovery**
  - Multiple changes to the database can be clubbed together as a single “logical transaction”.
  - The DBMS ensures that the updates take place physically, only when the logical transaction is complete.
- **Data Independence**
  - In conventional file based applications, programs need to know the “data organization” and “access technique” to be able to access the data.
  - This means that if you make any change in the manner the data is organized, then you have to make changes to the application programs that apply to the data.
  - In DBMS, the application programs are transparent to the “physical organization” and “access techniques”.



Characteristics of DBMS (contd.):

- **Availability of Productivity Tools**
  - Tools like query language, screen and report painter, and other 4GL tools are available.
  - These tools can be utilized by the end-users to query, print reports, etc. SQL is one such language, which has emerged as standard.
- **Security**
  - DBMSes provide tools, which can be used by the DBA to ensure security of the database.
- **Hardware Independence**
  - Most DBMSes are available across hardware platforms and operating systems.
  - Thus the application programs need not be changed or rewritten when the “hardware platform” or “operating system” is changed or upgraded.

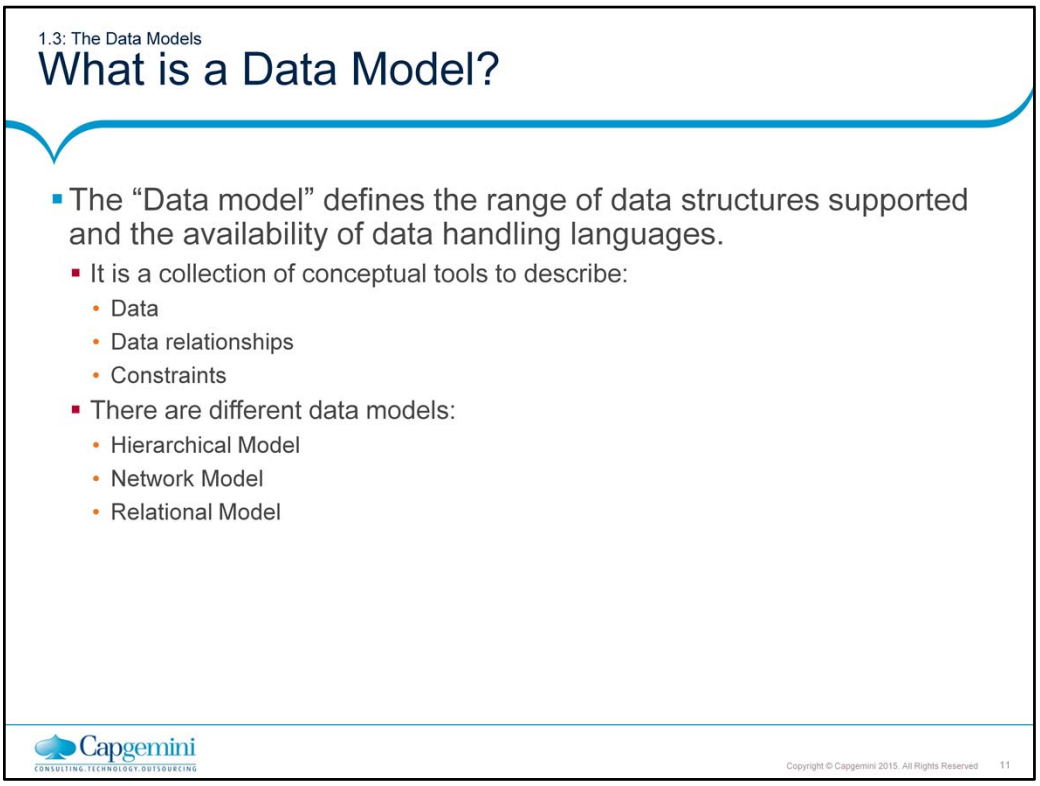
1.2: Features of DBMS

Levels of Abstraction

- There are three levels of database abstraction:
  - Conceptual Level:
    - The overall integrated structural organization of the database.
  - Physical Level:
    - The information about how the database is actually stored in the disk.
  - View / External Level:
    - The user view of the database. It is different for different users based on application requirement.

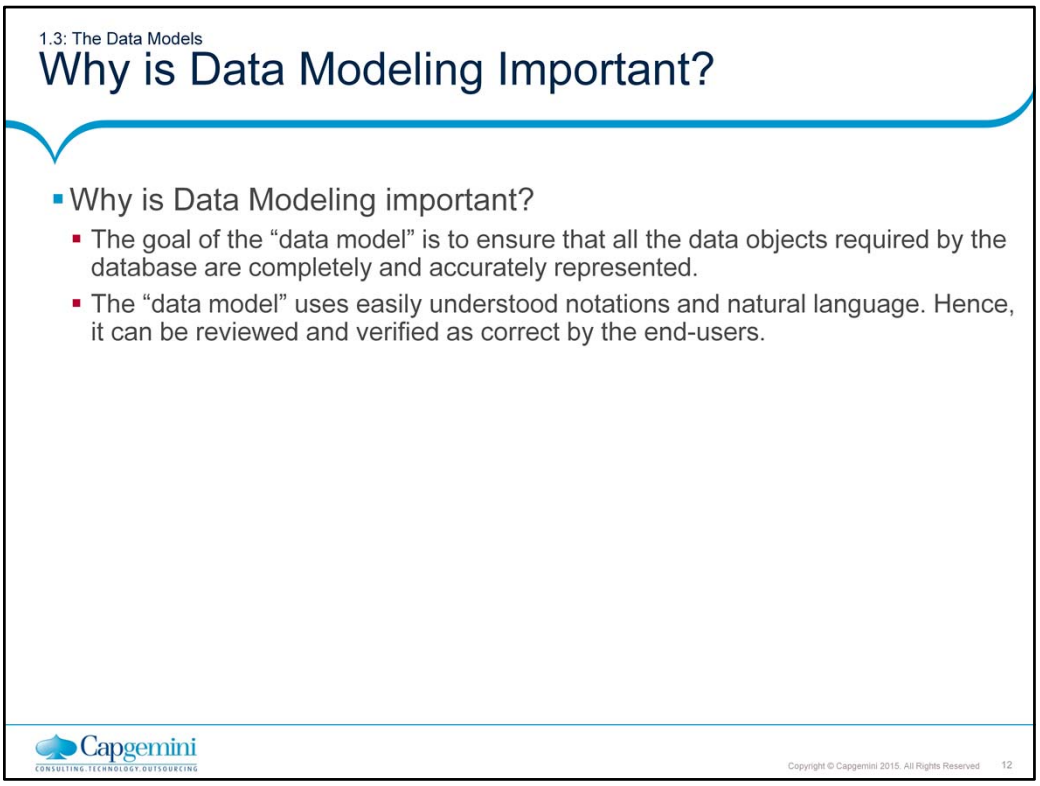
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10



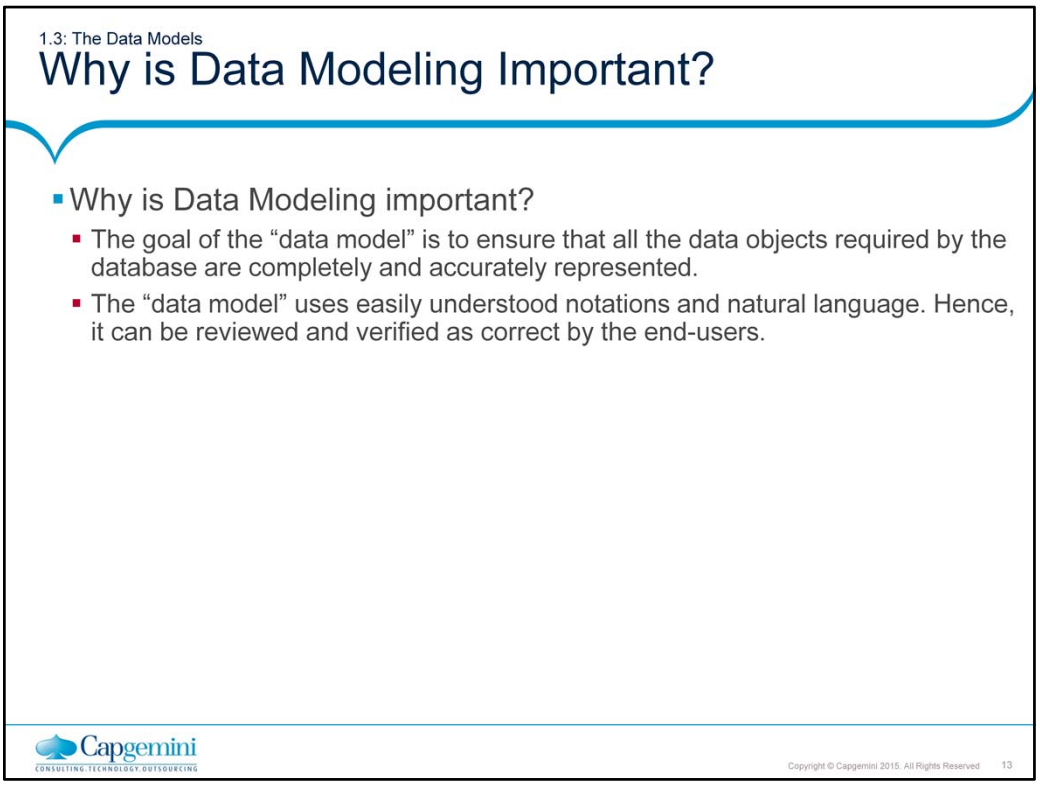
What is a Data Model?

- A “Data model” is a conceptual representation of the data structures that are required by a database. The data structures include:
  - the data objects
  - the associations between data objects, and
  - the rules which govern operations on the objects
- As the name implies, the “Data model” focuses on the data that is required, and how it should be organized rather than the operations that will be performed on the data.
- **The DBMS MODELS**
  - The range of “data structures” that are supported, and the availability of data handling languages depend on the model of DBMS on which it is based. The models are:
    1. The hierarchical model
    2. The network model
    3. The relational model



Why is Data Modeling Important?

- The “data model” is also detailed enough to be used, by the database developers, as a “blueprint” for building the physical databases. The information contained in the “data model” will be used to define the relational tables, primary and foreign keys, stored procedures, and triggers.
- Poorly designed databases require more time in the long-term. Without careful planning you may create a database that:
    1. Omits data required to create critical reports.
    2. Produces results that are incorrect or inconsistent.
    3. Is unable to accommodate changes in the user requirements.



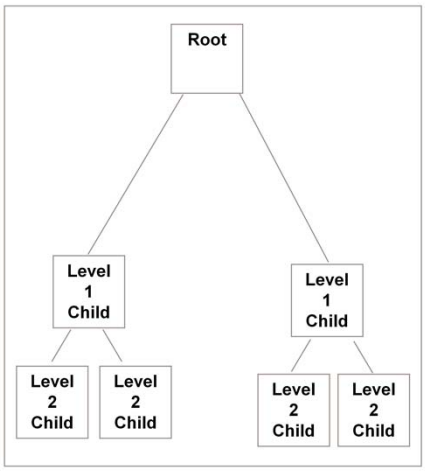
Why is Data Modeling Important?

- The “data model” is also detailed enough to be used, by the database developers, as a “blueprint” for building the physical databases. The information contained in the “data model” will be used to define the relational tables, primary and foreign keys, stored procedures, and triggers.
- Poorly designed databases require more time in the long-term. Without careful planning you may create a database that:
    1. Omits data required to create critical reports.
    2. Produces results that are incorrect or inconsistent.
    3. Is unable to accommodate changes in the user requirements.

1.3: The Data Models

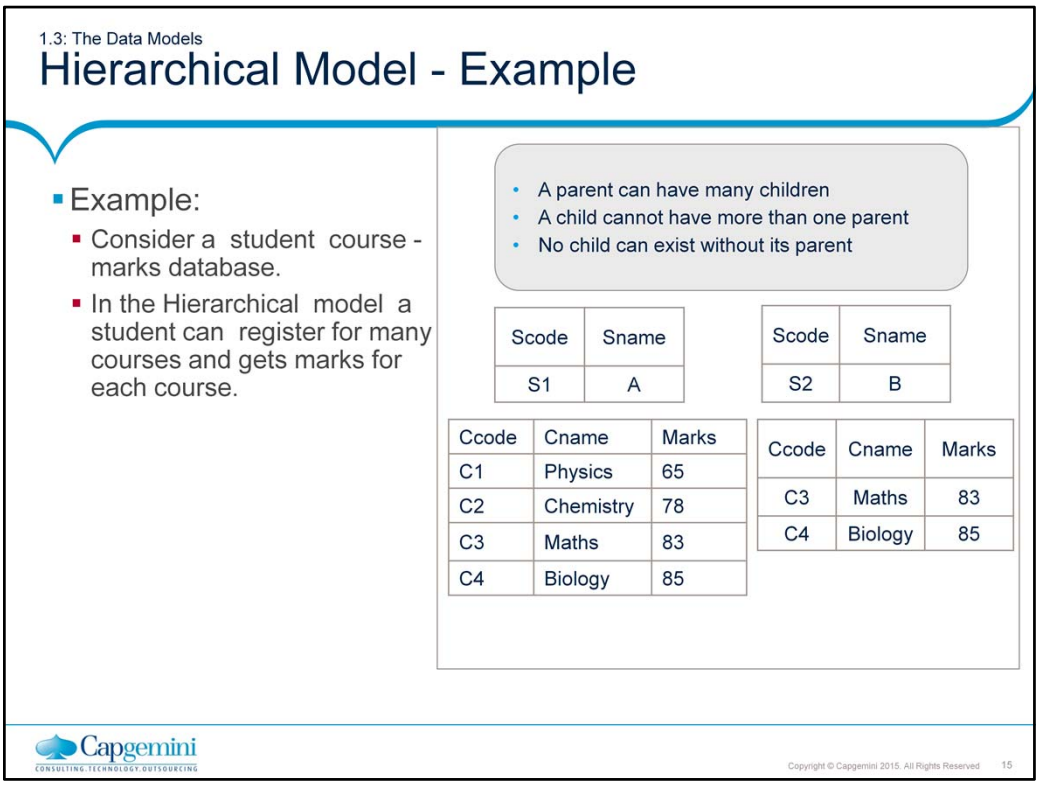
Hierarchical Model

- The Hierarchical model:
  - In this model, data is represented by a simple tree-structure.
  - Relationships between entities are represented as parent-child.
  - Many-to-many relationships are not allowed.
  - Parents and children are tied together by links called “pointers”.



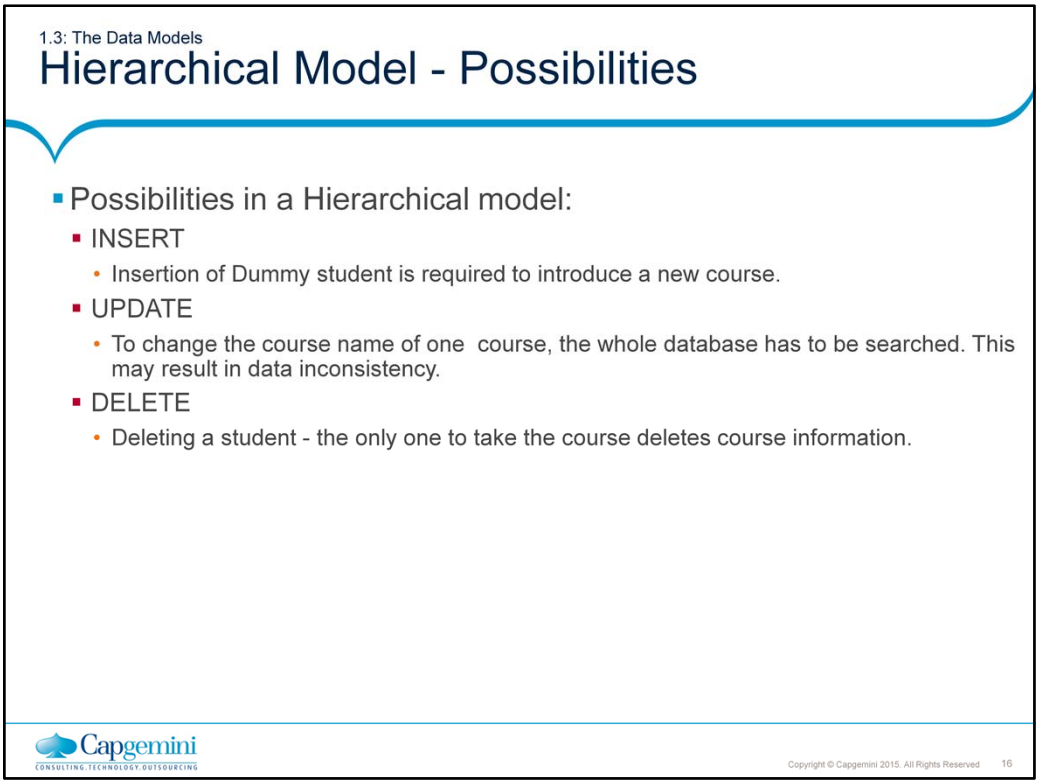
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14



Example of a Hierarchical model:

- Consider a student course - marks database. In the Hierarchical model a student can register for many courses, and get marks for each course.
- The student record is called as “root”. It has got a course - marks record that is called as “child record”.
- In general:
  - A parent can have many children.
  - A child cannot have more than one parent.
  - No child can exist without its parent.



Possibilities in a Hierarchical model:

In the Hierarchical model, following possibilities exist:

- INSERT
  - Since no child record can exist without it's parent, it is not possible to insert the new course details without introducing a dummy student record.

- UPDATE
  - To change the course name of one course, the whole database has to be searched. This may result in data inconsistency.

1.3: The Data Models

Network Model

- The Network model:
  - The Network model solves the problem of data redundancy by representing relationships in terms of “sets” rather than “hierarchy”.
  - A record occurrence may have any number of immediate superiors.
  - The Network model supports many-to-many relationships.
  - There is no restriction on number of parents.
  - A record type can have a number of parent and child record types.
  - It is more complex than the Hierarchical model because of links.
  - It is a superset of the Hierarchical model.

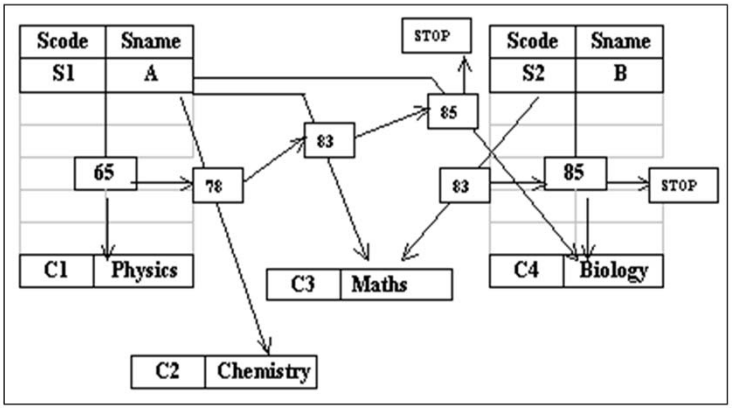
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

1.3: The Data Models

Network Model - Example

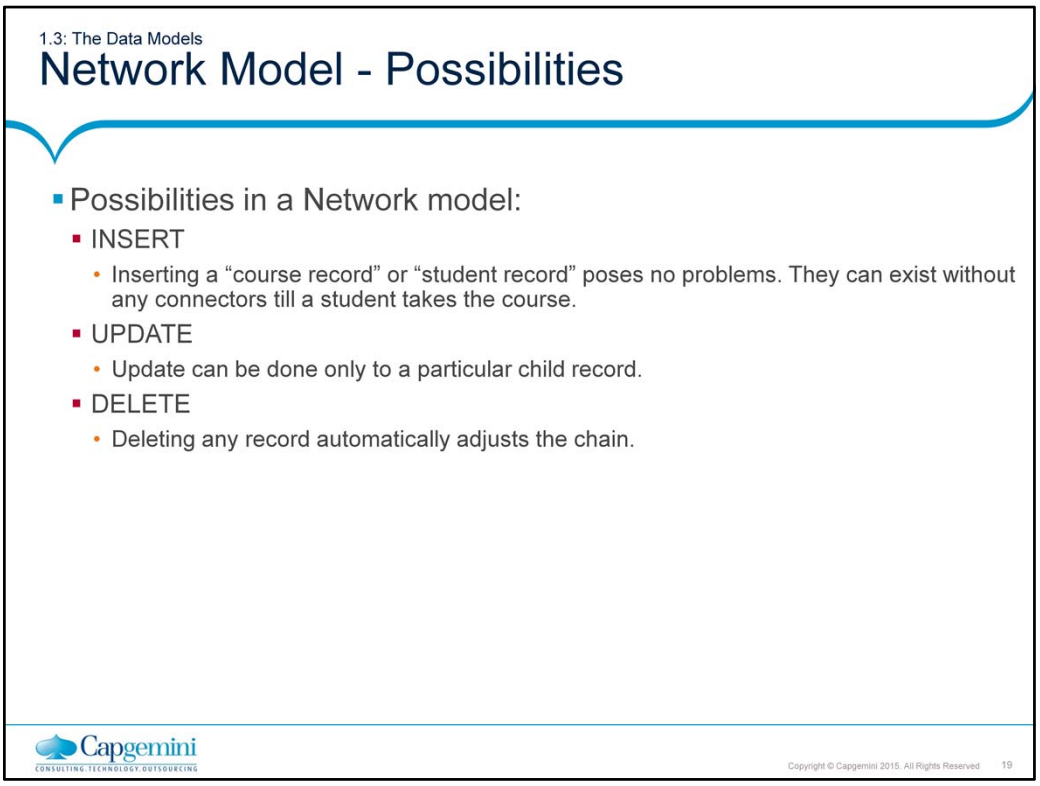
- In the example of student course – marks, "student record" and "course record" is linked together through "marks record".



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Example of Network model:

- In the Network model, the "student record" and "course record" is linked together through a "marks record".
- There are no restrictions on number of parents.
- A record type can have any number of "parent" and "child" record types.
- The Network model is more complex than the Hierarchical model because of it's links.
- The Network model can represent any structure that is designed in the Hierarchical model. Hence, it is a superset of the Hierarchical model.



Possibilities in a Network model:

In Network model, following possibilities exist:

- INSERT
  - Inserting a course record or student record poses no problems, as they can exist without any connectors till a student takes the course.
- UPDATE
  - Update can be done only to a particular child record.
- DELETE
  - Deleting any record automatically adjusts the chain.

1.3: The Data Models

Relational Model

- The Relational model:
  - The Relational model developed out of the work done by Dr. E. F. Codd at IBM in the late 1960s. He was looking for ways to solve the problems with the existing models.
  - At the core of the Relational model is the concept of a “table” (also called a “relation”), which stores all data.
  - Each “table” is made up of:
    - “records” (i.e. horizontal rows that are also known as “tuples”), and
    - “fields” (i.e. vertical columns that are also known as “attributes”)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

1.3: The Data Models

Relational Model

- The Relational model:
  - Examples of RDBMS:
    - Oracle
    - Informix
    - Sybase
  - Because of lack of linkages, the Relational model is easier to understand and implement.

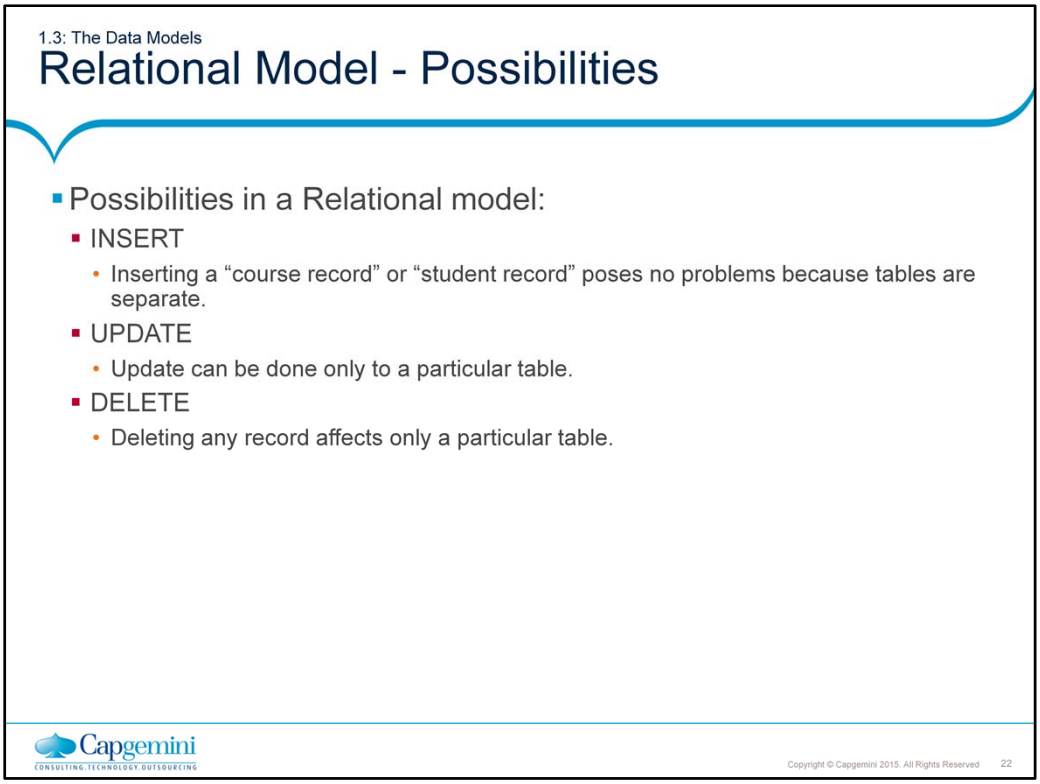
Student Table	
Scode	Sname
S1	A
S2	B

Course Table	
Ccode	Cname
C1	Physics
C2	Chemistry
C3	Maths
C4	Biology

Marks Table		
Ccode	Scode	Marks
C1	S1	65
C2	S1	78
C3	S1	83
C4	S1	85
C3	S2	83
C4	S2	85

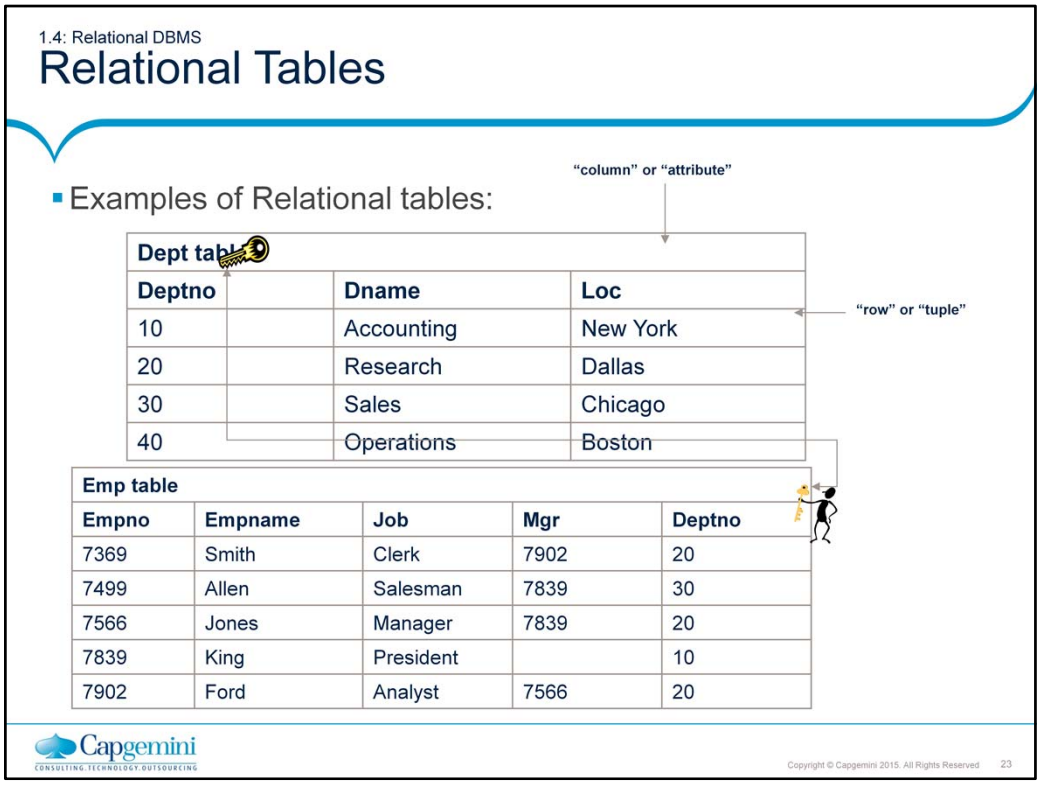
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21



In Relational model, following possibilities exist:

- INSERT
  Inserting a course record or student record poses no problems, as the insert takes place on different tables
  Inserting a child record would first check for the existence of the column column value being present in the parent table.
- UPDATE
  - Update can be done only to a particular record, few records by specifying a condition.
- DELETE
  - Deleting a record/few records of a particular table can be done by specifying a condition.
  - Deleting a parent record would not be allowed if child records exist.

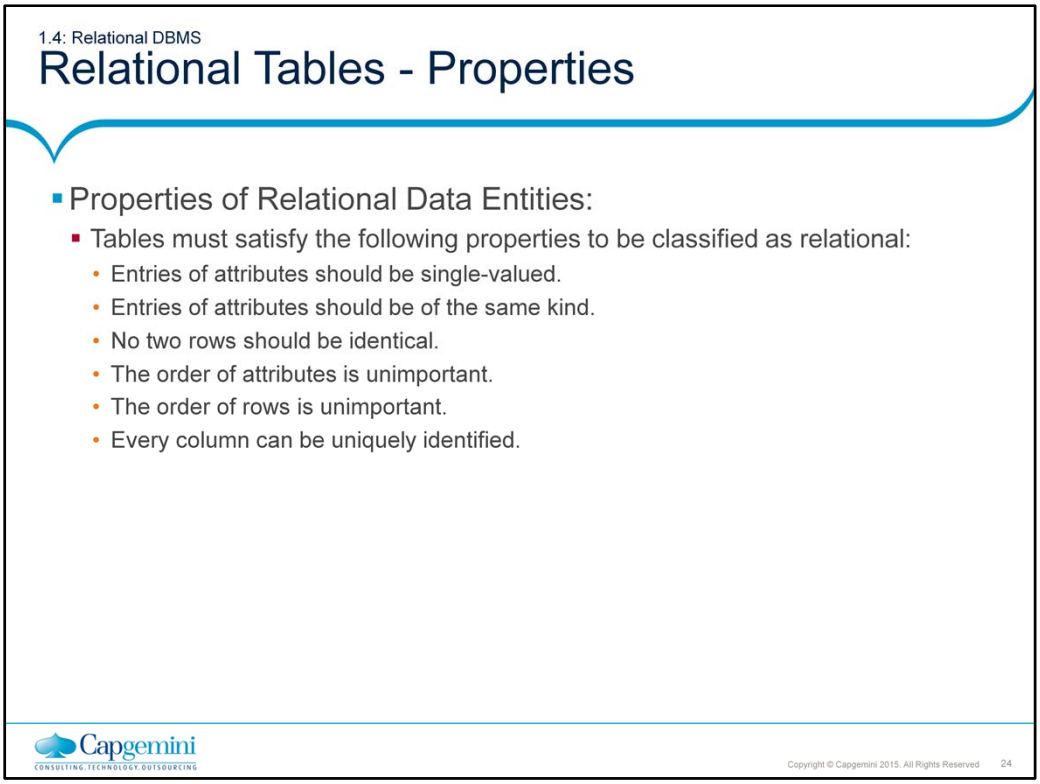


Relational DBMS (RDBMS):

- The Relational model presents an orderly, predictable, and intuitive approach for:
  - Organizing data,
  - Manipulating data, and
  - Viewing data

RDBMS Terminology

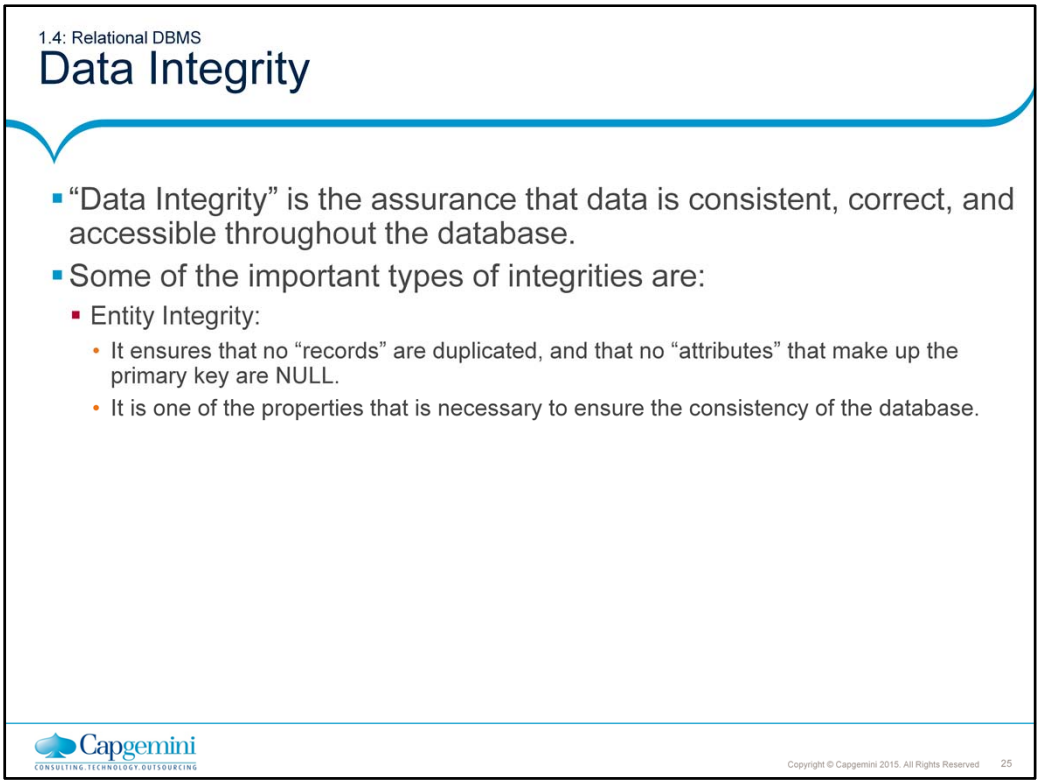
- Relational data consists of relations.
  - A relation (or relational table) is a "two dimensional" table with special properties.
- A relational table consists of:
  - a set of named columns, and
  - an arbitrary number of rows
- The columns are called as "attributes" or "fields". The rows are called as "tuples" or "records".
- Each "attribute" is associated with a "domain".
  - A "domain" is a set of values that may appear in one or more columns.



Properties of Relational Data Entities:

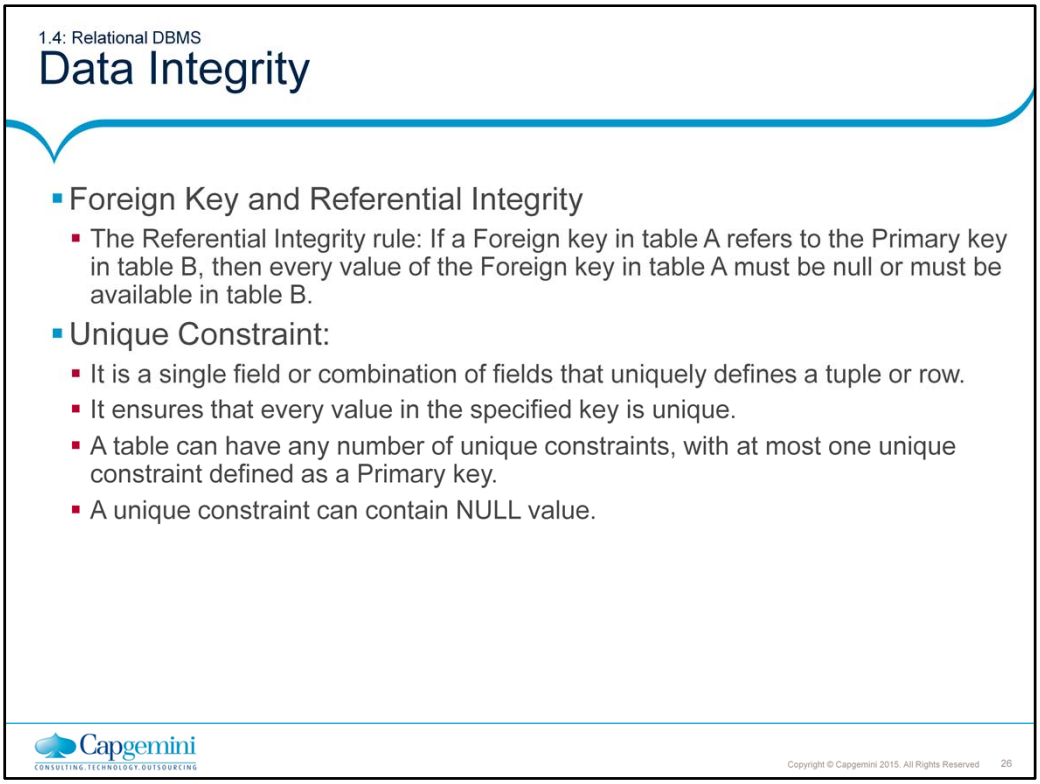
Relational tables have six properties,which must be satisfied for any table to be classified as Relational.These are :

1. Single valued attributes
2. Same datatype attribute values
3. Primary key
4. Attribute order not important
5. Row order not important
6. Uniquely identifiable column



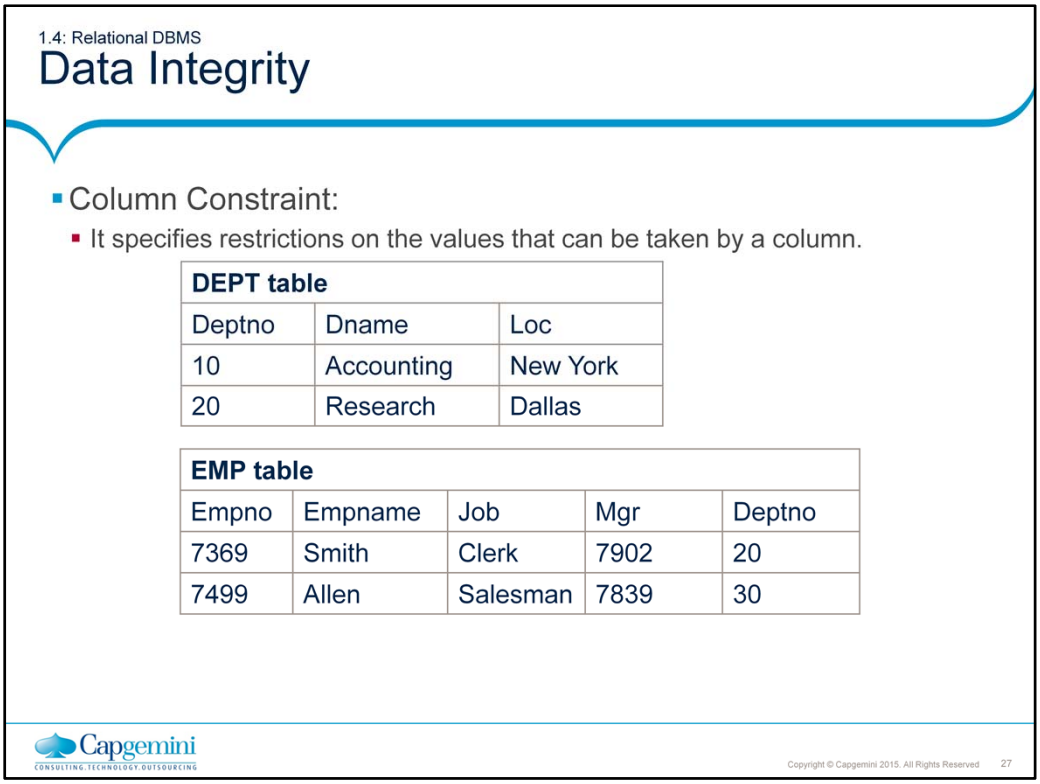
Data Integrity:
Data Integrity refers to the wholeness and soundness of the database.
Some of the most important integrities are given below.

- **Domain Constraints**
  - A "domain" is a set of values that are permitted to appear in one or more columns. Once a "domain" is specified and a "column" is associated with the "domain", then the "column" can take only those values that are permitted by the "domain".
- **Primary Key and Entity Integrity**
  - "Primary key" is a "column" or "set of columns" in a table which uniquely identifies a "row" in a table.
    - No two rows of the table can have the same values for the Primary key.
    - Entity integrity is maintained by ensuring that none of the columns that make up the Primary key can take "NULL" (unknown) values.



Data Integrity:

- **Foreign Key and Referential Integrity**
  - Foreign Key concept relates two tables. The child table's column which refers to the parent table's primary key column is called a foreign key column.
  - How to differentiate between parent and child table ?
    - Look at the values in the common column of both the tables. If there are unique values in the column, then this column is a primary key column and is present in the parent table.
    - If there are duplicate values in the common column, then this column is a foreign key column and is present in the child table.
    - A foreign key column can contain null values also, unlike a primary key column
- **Unique Constraint**
  - Unique constraint is applied to one or more columns to ensure that values in that column cannot repeat.
  - Applying a primary key constraint automatically makes the column unique and not null



Data Integrity (contd.):

- **Update Cascade referential Integrity**
  - This means that if a Primary key value is updated, then all Foreign key values dependent on it will be updated to the new value of Primary key. That is to say, if we change the deptno 10 to 50, then all the employees in deptno 10 will be shifted, as well, to deptno 50 (column deptno will be automatically updated).
- **Column Constraints**
  - These are the constraints, which specify restrictions on the values that can be taken by a column. These restrictions may be defined with or without other values in the same row.

1.5 Database Administrator

Database Administrator

- A Database Administrator (DBA) is the database architect.
  - DBA is responsible for the design and implementation of new databases, and:
    - centrally manages the database.
    - decides on the type of data, internal structures, and their relationships
    - ensures the security of the database
    - controls access to the data through user codes and passwords
    - can restrict the views or operations that the users can perform on the database

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28

Summary

- In this lesson, you have learnt:
  - What is a Database?
  - Characteristics of DBMS
  - The Data models, including:
    - the Hierarchical model
    - the Network model
    - the Relational model
  - Relational DBMS (RDBMS)
  - What is Data Integrity

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 29

Review Question

- Question 1: A DBA ___.
  - Option 1: ensures the security of the application server.
  - Option 2: controls access to the data through the user codes and passwords.
  - Option 3: manages users.

- Question 2: The Physical Level is ___.
  - Option 1: the overall structural organization of the d/b.
  - Option 2: the information about how the database is actually stored in the disk.
  - Option 3: the user view of the database.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 30

Review Question

- Question 3: There are different data models such as ___.
- Question 4: In Network model, each table is made up “tuples” and “fields”.
  - True / False
- Question 5: A table can have any number of “Unique constraints”.
  - True / False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31

Review Question: Match the Following

1. Hierarchical model	a) Inconsistencies may creep in.
2. Network model	b) Many-to-many relationships are not allowed.
3. Data redundancy	c) It is a superset of the Hierarchical model.
4. In DBMS	d) Application programs are transparent to the physical organization and access techniques.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 32

DBMS/SQL

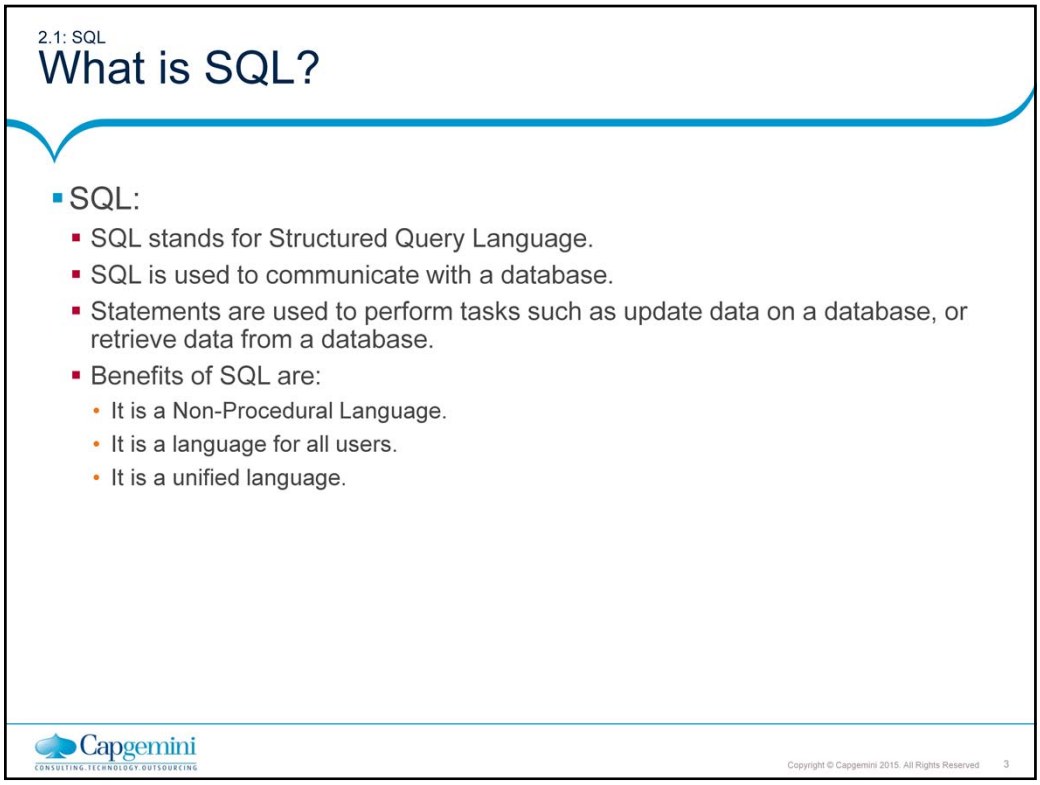
Lesson 02 Basics of SQL

Lesson Objectives

- To understand the following topics:
  - SQL Language
  - Rules for SQL Statements
  - Standard SQL Statement groups
  - Logging to Oracle Server

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

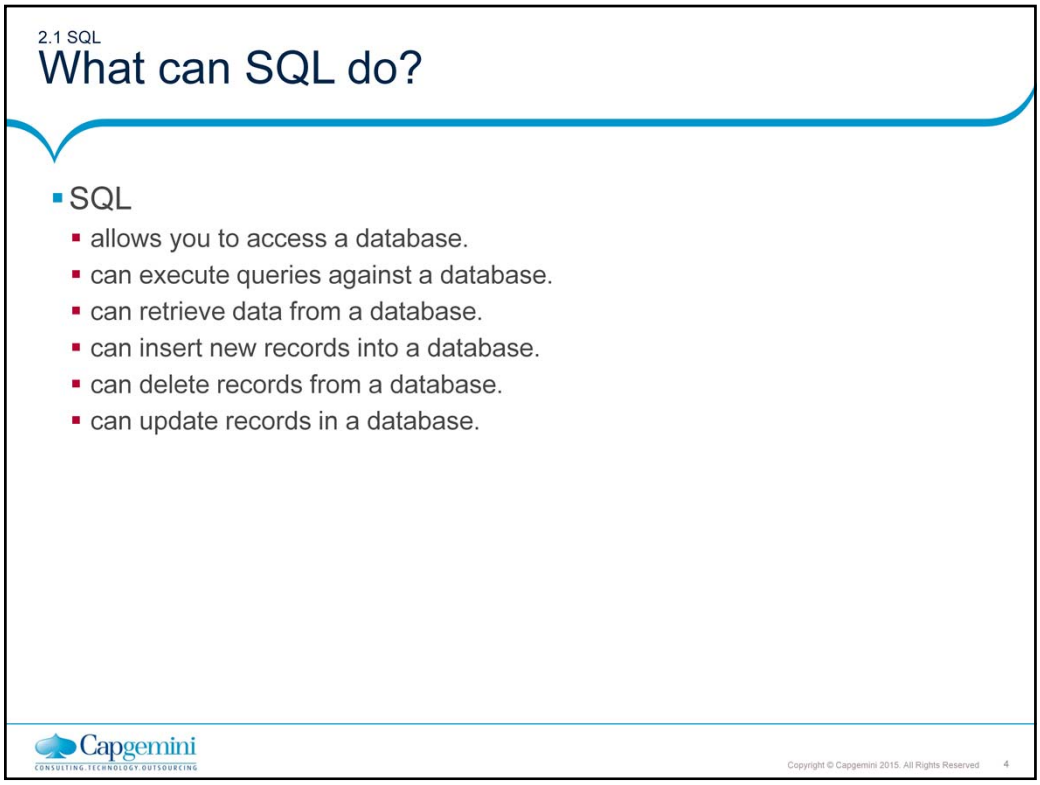
 2



What is SQL?

SQL stands for Structured Query Language. Pronounced as SEQUEL, SQL was originally developed by IBM based on Dr. E.F. Codd's relational model to define, manipulate and control data in a database

•SQL is a set of commands that allows you to access a Relational Database.
•SQL is an ANSI standard computer language.
•SQL can execute queries against a database.
•SQL can retrieve data from a database.
•SQL can insert new records in a database.
•SQL can delete records from a database.
•SQL can update records in a database.
•SQL is easy to learn.
•SQL is a non-procedural, English-like language that processes data in "groups of records" rather than processing one record at a time.
•SQL provides automatic navigation to the data.



Almost all modern Relational Database Management Systems like MS SQL Server, Microsoft Access, MSDE, Oracle, DB2, Sybase, MySQL, Postgres, and Informix use SQL as standard database language.
However, although all those RDBMS use SQL, they use different SQL dialects
For example: MS SQL Server specific version of the SQL is called T-SQL, Oracle version of SQL is called PL/SQL which also supports Procedural Language features, MS Access version of SQL is called JET SQL, etc.

- Oracle has further modified SQL to support ORDBMS features.

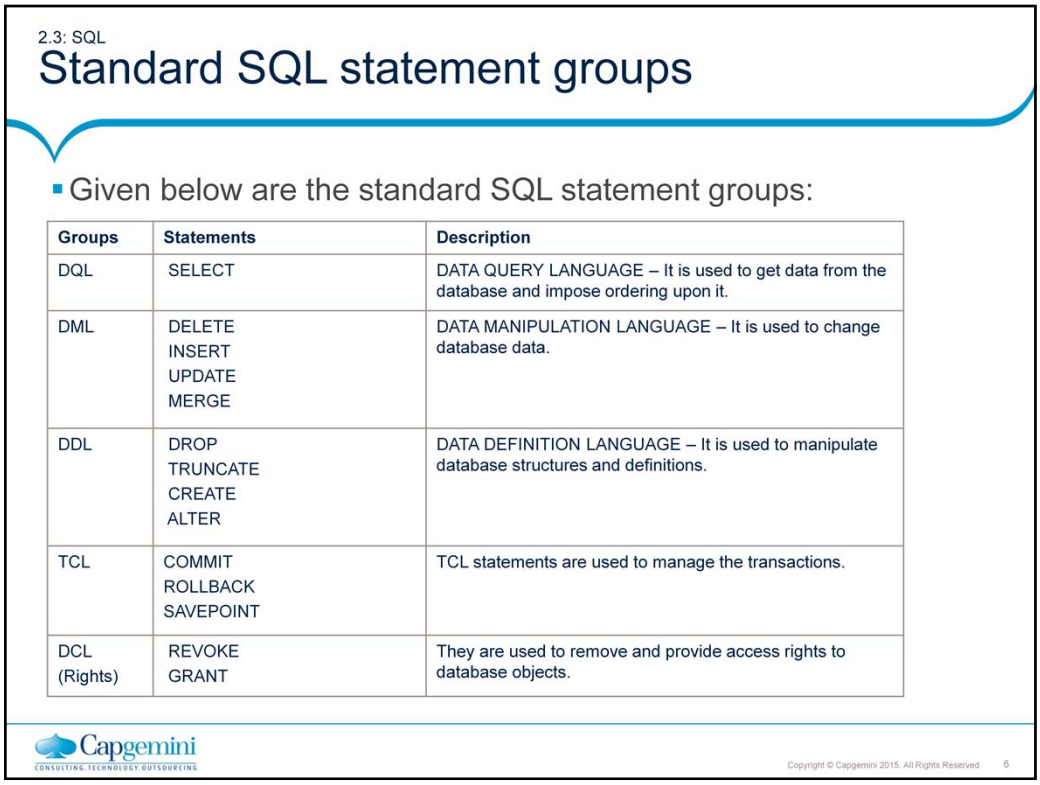
2.2: SQL

Rules for SQL statements

- Rules for SQL statements:
  - SQL keywords are not case sensitive. However, normally all commands (SELECT, UPDATE, etc) are upper-cased.
  - "Variable" and "parameter" names are displayed as lower-case.
  - New-line characters are ignored in SQL.
  - Many DBMS systems terminate SQL statements with a semi-colon character.
  - "Character strings" and "date values" are enclosed in single quotation marks while using them in WHERE clause or otherwise.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5



Groups	Statements	Description
DQL	SELECT	DATA QUERY LANGUAGE – It is used to get data from the database and impose ordering upon it.
DML	DELETE INSERT UPDATE MERGE	DATA MANIPULATION LANGUAGE – It is used to change database data.
DDL	DROP TRUNCATE CREATE ALTER	DATA DEFINITION LANGUAGE – It is used to manipulate database structures and definitions.
TCL	COMMIT ROLLBACK SAVEPOINT	TCL statements are used to manage the transactions.
DCL (Rights)	REVOKE GRANT	They are used to remove and provide access rights to database objects.

Standard SQL Statement groups:

•"SQL commands" are "instructions" used to communicate with the database to perform "specific tasks" that work with data.

- ➢A database is a collection of structures with appropriately defined authorizations and accesses. The tables, indexes are structures in the database and are called as "objects" in the database.
- ➢The names of tables, indexes, and those of columns are called "identifiers".

•SQL commands can be used not only for searching the database, but also to perform various other functions.

For example: You can create tables, add data to tables, or modify data, drop the table, set permissions for users, etc.

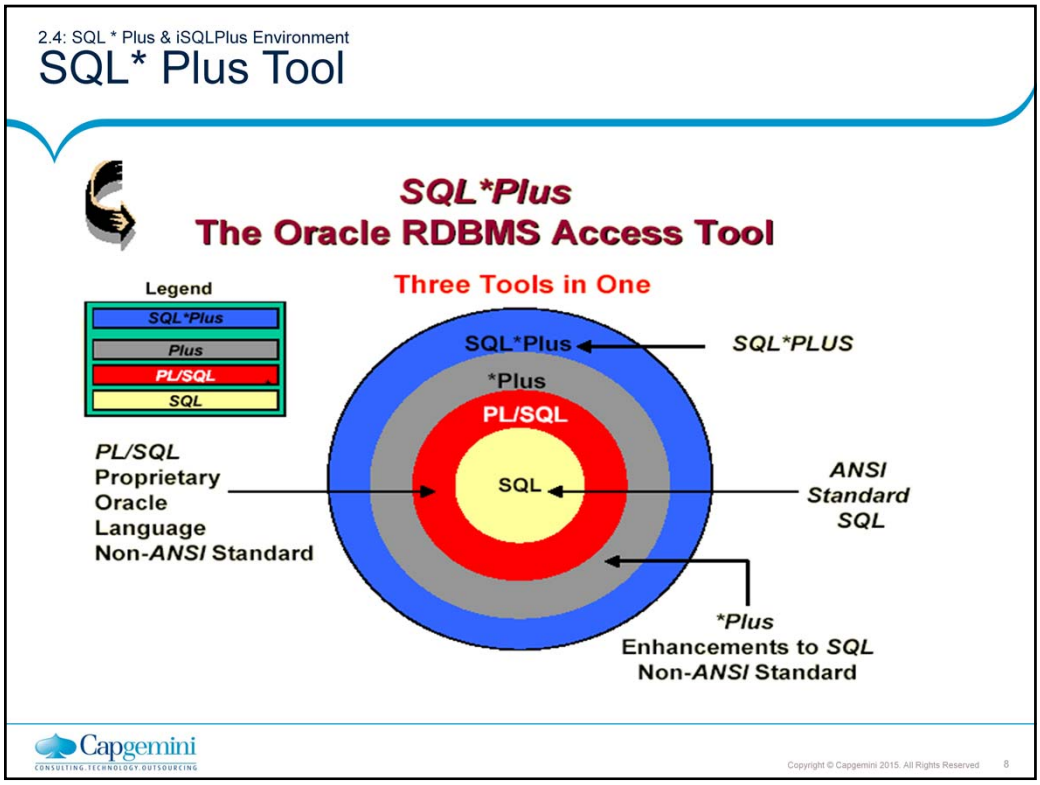
Standard SQL Statement groups:

•"SQL commands" are "instructions" used to communicate with the database to perform "specific tasks" that work with data.

- A database is a collection of structures with appropriately defined authorizations and accesses. The tables, indexes are structures in the database and are called as "objects" in the database.
- The names of tables, indexes, and those of columns are called "identifiers".

•SQL commands can be used not only for searching the database, but also to perform various other functions.

For example: You can create tables, add data to tables, or modify data, drop the table, set permissions for users, etc.



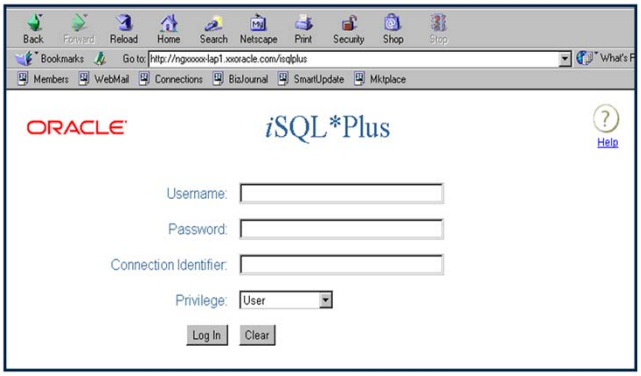
SQL*PLUS:

• SQL* PLUS is an Oracle tool which accepts SQL, and SQLPLUS commands and PL/SQL blocks, and executes them.SQL*Plus commands can be abbreviated

2.4: SQL * Plus & iSQLPlus Environment

Logging to Oracle Server using iSQLPlus

- To log into the iSQL*Plus environment:
  - In the Windows browser, type the URL in the address field. The user will be directed to iSQL*Plus environment screen.



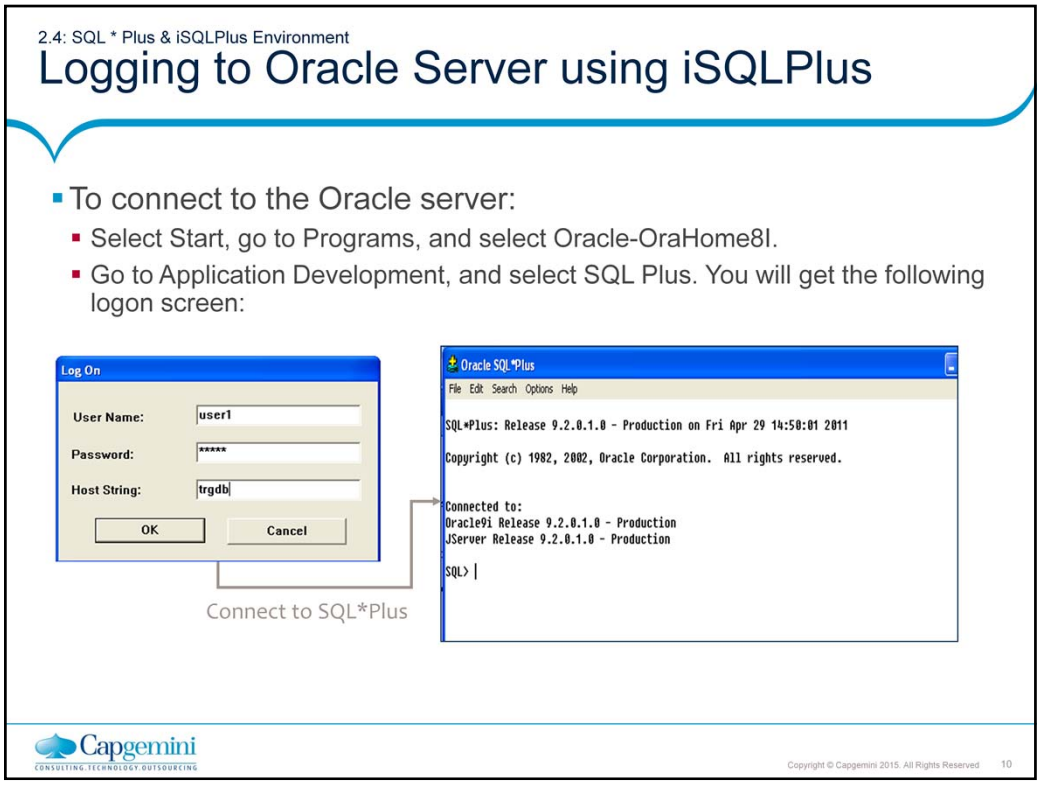
Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Oracle provides a browser based facility also to access SQL* Plus i.e iSQLPlus. Unlike SQL*Plus the commands in iSQLPlus cannot be abbreviated. Both SQL*Plus and iSQLPLus are Oracle proprietary interfaces.

How to access SQL* PLUS ?

1. Get your Oracle user name and password, and connect string from your DBA or Instructor.
2. As shown on the slide you get the login screen wherein the required details can be put in to login into Oracle



Similarly you can use the username, password and Host String details to use the SQL * Plus tool. This is a GUI based tool to interact with Oracle

A. **To enter a SQL> command**, type the command at the SQL > prompt. An SQL command can be continuously entered on a single line, or it can be spread over multiple lines. In case of any typing error, use BACKSPACE or DEL key to erase, and then continue typing.

   SQL> SELECT empno,hiredate FROM emp WHERE hiredate >
   '01-jan-82' ;

   The command can also be entered as:
   SQL>SELECT empno,hiredate
   FROM emp
   WHERE hiredate >'01-jan-82';

Summary

- In this lesson, you have learnt:
  - What is SQL?
    - Rules for SQL statements
    - Standard SQL statement groups
  - SQL*PLUS environment

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

Review Question

- Question 1: SQL ___.
  - Option 1: cannot execute queries against a database.
  - Option 2: can manipulate data from a database.
  - Option 3: cannot retrieve data from a database.
  - Option 4: can insert new records in a database.
  - Option 5: can delete records from a database.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Review Question

- Question 2: SQL categories are ___.
  - Option 1: DDL
  - Option 2: DML
  - Option 3: DSL
  - Option 4: DQL
  - Option 5: TCL
  - Option 6: TDL

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Add the notes here.

DBMS/SQL

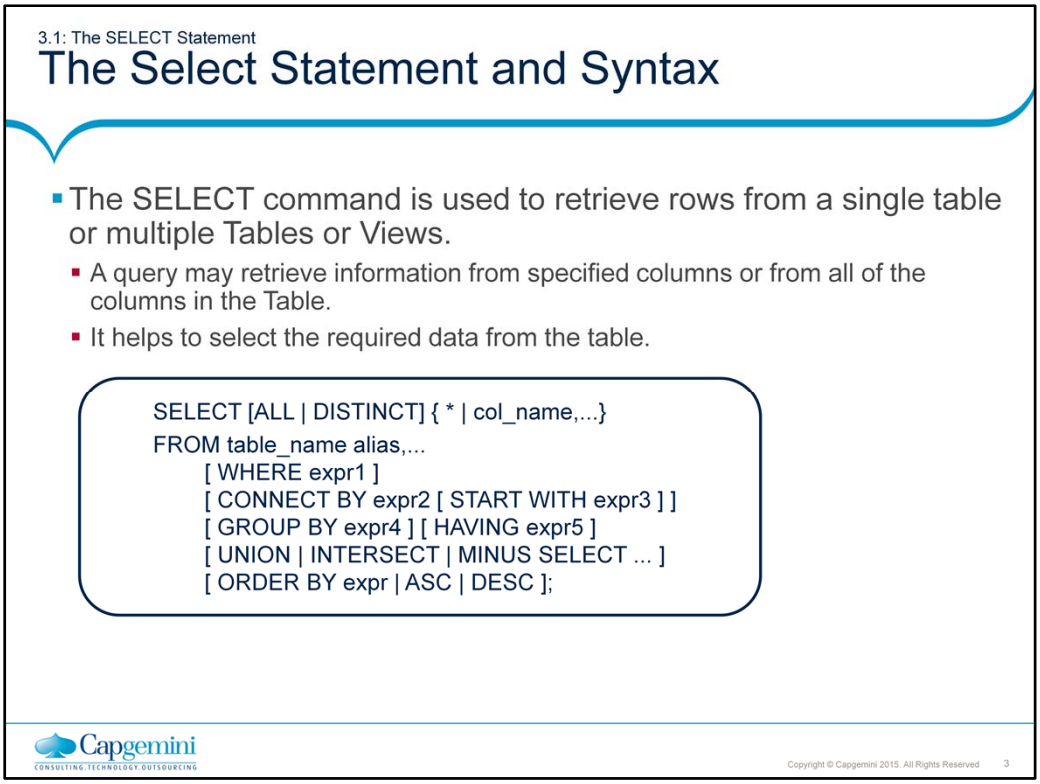
Data Query Language (The Select Statement)

Lesson Objectives

- To understand the following topics:
  - The SELECT statement
  - The WHERE clause
  - The Mathematical, Comparison and Logical operators
  - The DISTINCT clause
  - The ORDER BY clause
  - Tips and Tricks in SELECT Statement

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

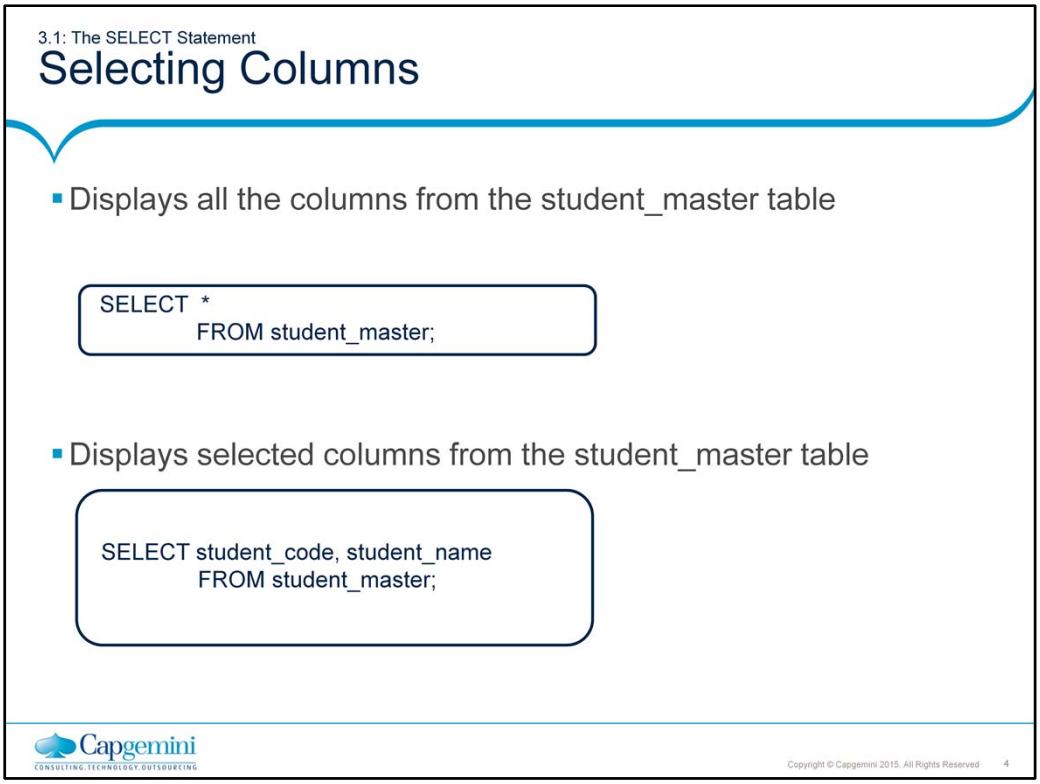


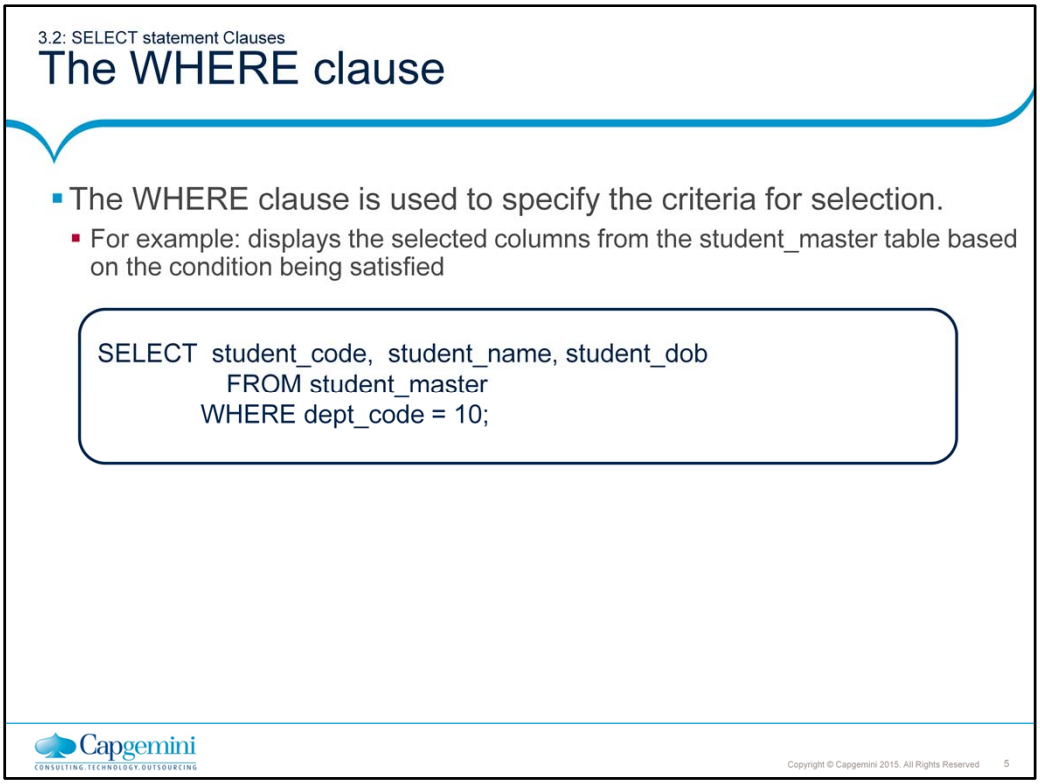
The SELECT Statement:

- The SELECT statement is used to select data from a table. The tabular result is stored in a result table (called the result-set). The statement begins with the SELECT keyword. The basic SELECT statement has three clauses:
  - SELECT
  - FROM
  - WHERE
- The SELECT clause specifies the table columns that are retrieved.
- The FROM clause specifies the tables accessed.
- The WHERE clause specifies which table rows are used. The WHERE clause is optional; if missing, all table rows are used.

Note:

- Each clause is evaluated on the result set of a previous clause. The final result of the query will be always a “result table”.
- Only FROM clause is essential. The clauses WHERE, GROUP BY, HAVING, ORDER BY are optional.
- All the examples that follow are based on EMP and DEPT tables that are already available.


3.1: The SELECT Statement
Selecting Columns
Displays all the columns from the student_master table
SELECT *
FROM student_master;
Displays selected columns from the student_master table
SELECT student_code, student_name
FROM student_master;
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
4



The WHERE Clause:

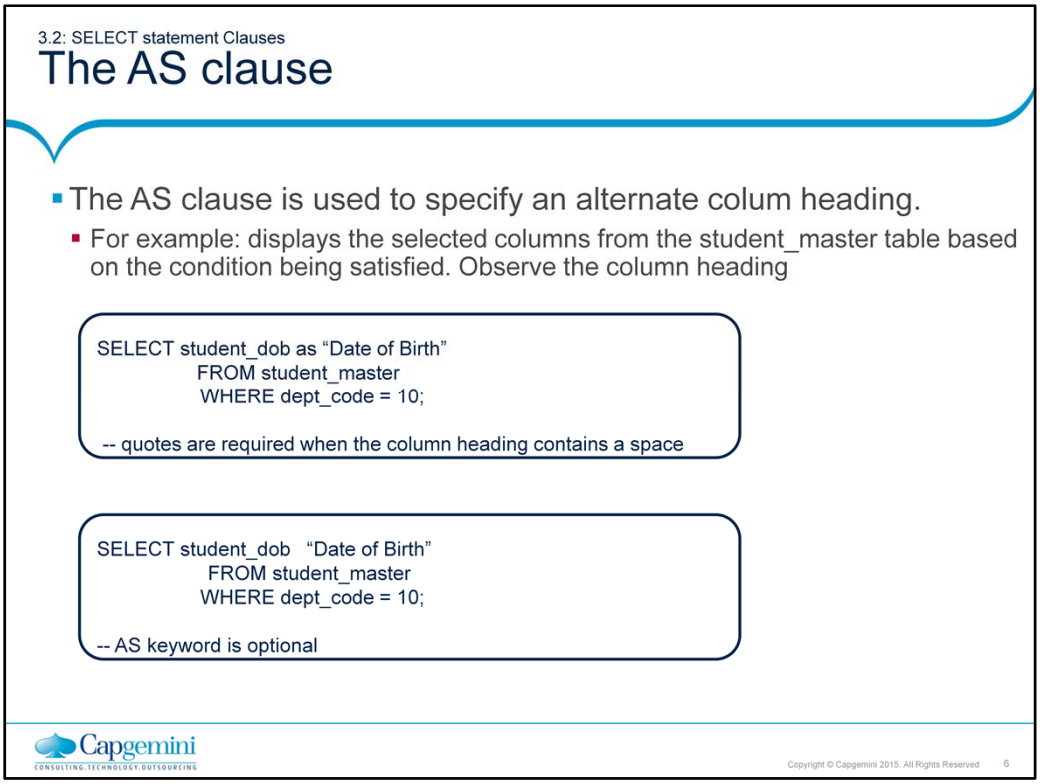
- The WHERE clause is used to perform "selective retrieval" of rows. It follows the FROM clause, and specifies the search condition.
- The result of the WHERE clause is the row or rows retrieved from the Tables, which meet the search condition.
- The clause is of the form:

```
WHERE <search condition>
```

Comparison Predicates:

- The Comparison Predicates specify the comparison of two values.
  - It is of the form:
    < Expression> < operator > < Expression>
    < Expression> <operator> <subquery>
  - The operators used are shown on the next slide:

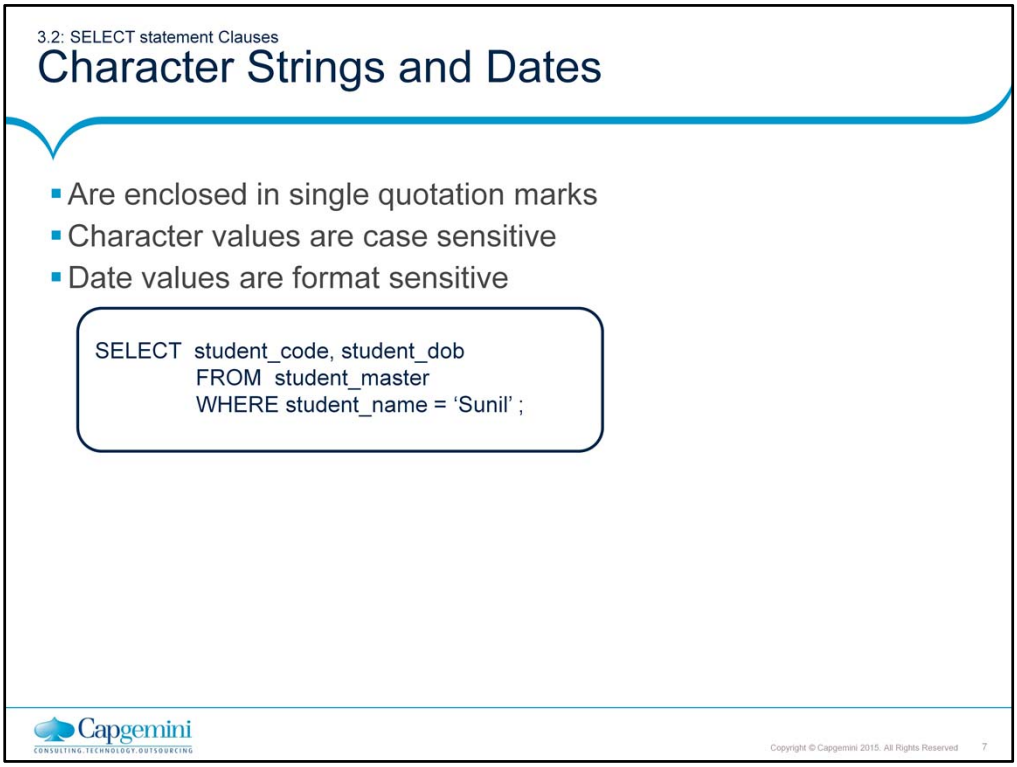
contd.



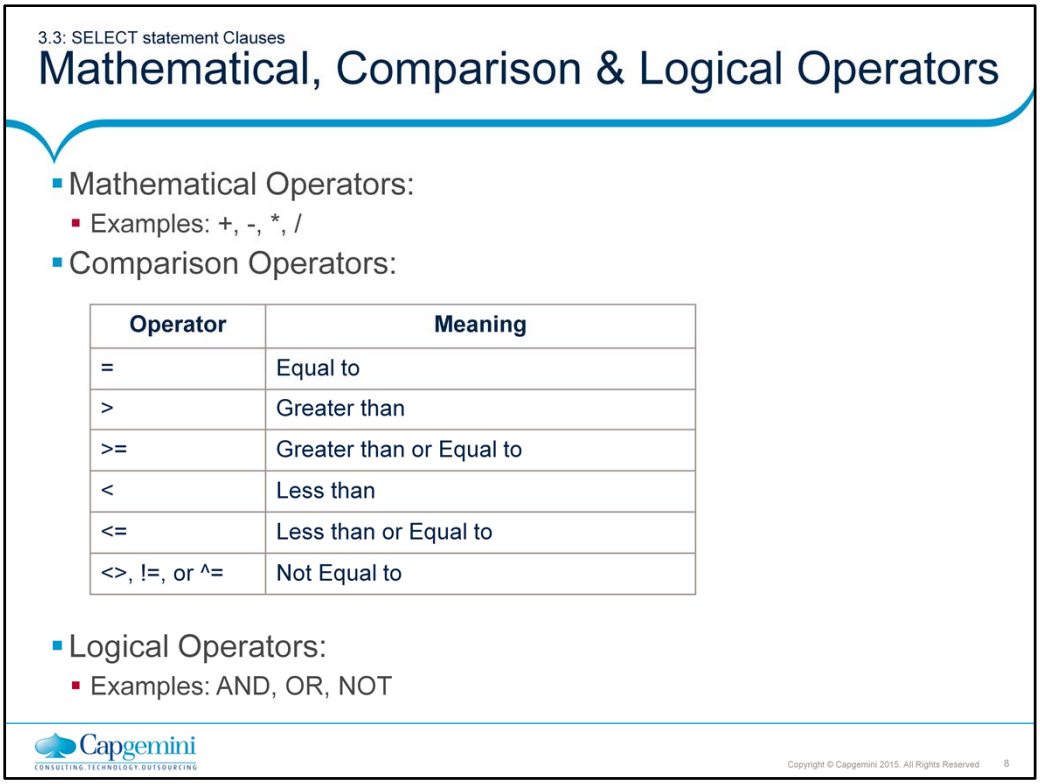
The AS Clause:

- The AS clause is used to give a different column heading (other than column name) to one or more columns used in the select statement. It follows the column name, and can be used for one or more columns.
- The AS keyword is optional.
- The clause is of the form:

```
Select column1  heading1, column2  as  heading1,
column3 as "heading3 contains space" from
table_name
```



Oracle Database store dates in an internal numeric format, representing the century, year, month, day, hours, minutes, and seconds.
The date datatype is covered in detail later.



Operators:

- Operators are used in “expressions” or “conditional statements”. They show equality, inequality, or a combination of both.
- Operators are of three types:
  - mathematical
  - logical
  - range (comparison)
- These operators are mainly used in the WHERE clause, HAVING clause in order to filter the data to be selected.
- **Mathematical operators:**
  These operators add, subtract, multiply, divide, and compare equality of numbers and strings. They are +, -, *, /
- **Comparison Operators:**
  These operators are used to compare the column data with specific values in a condition. “Comparison Operators” are also used along with the “SELECT statement” to filter data based on specific conditions. The table in the slide describes each Comparison operator. Comparison operators indicate how the data should relate to the given search value.
- **Logical Operators:**
  There are three Logical Operators namely AND, OR and NOT. These operators compare two conditions at a time to determine whether a row can be selected for the output or not. When retrieving data by using a SELECT statement, you can use logical operators in the WHERE clause. This allows you to combine more than one condition.

3.3: SELECT statement Clauses

Other Comparison Operators

Other Comparison operators	Description
[NOT] BETWEEN x AND y	Allows user to express a range. For example: Searching for numbers BETWEEN 5 and 10. The optional NOT would be used when searching for numbers that are NOT BETWEEN 5 AND 10.
[NOT] IN(x,y,…)	Is similar to the OR logical operator. Can search for records which meet at least one condition contained within the parentheses. For example: Pubid IN (1, 4, 5), only books with a publisher id of 1, 4, or 5 will be returned. The optional NOT keyword instructs Oracle to return books not published by Publisher 1, 4, or 5.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

3.3: SELECT statement Clauses

Other Comparison Operators

Other Comparison operators	Description
[NOT] LIKE	Can be used when searching for patterns if you are not certain how something is spelt. For example: title LIKE 'TH%'. Using the optional NOT indicates that records that do contain the specified pattern should not be included in the results.
IS[NOT]NULL	Allows user to search for records which do not have an entry in the specified field. For example: Shipdate IS NULL. If you include the optional NOT, it would find the records that do not have an entry in the field. For example: Shipdate IS NOT NULL.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

3.3: SELECT statement Clauses

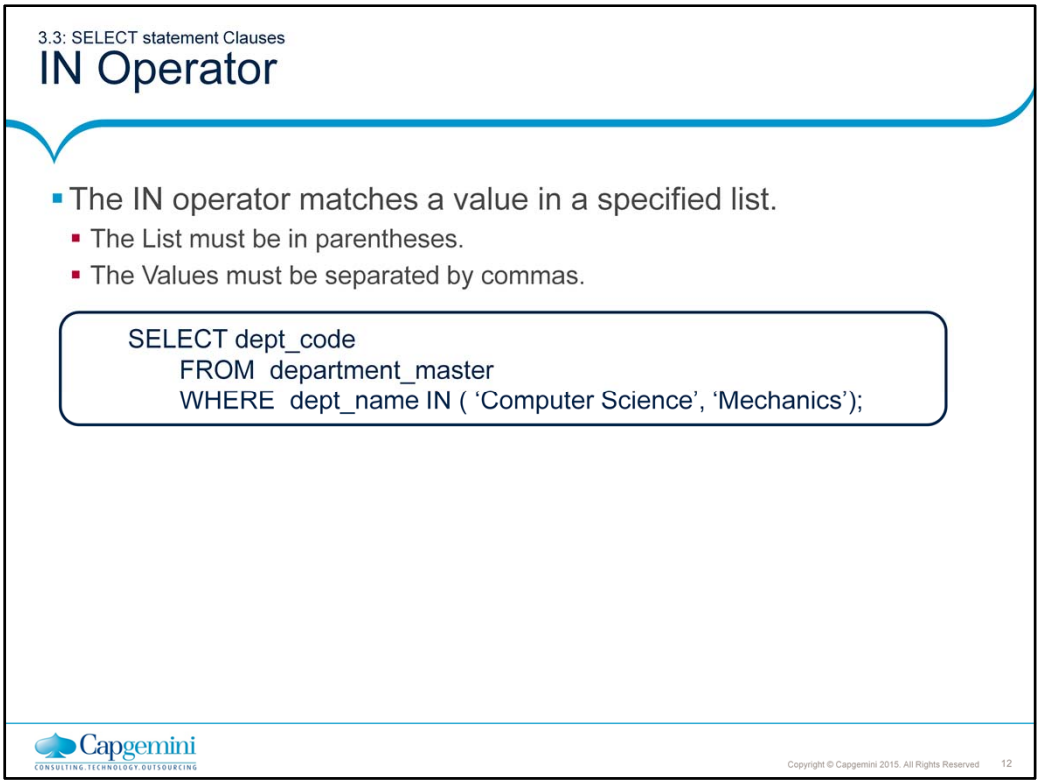
BETWEEN … AND Operator

- The BETWEEN … AND operator finds values in a specified range:

```
SELECT staff_code,staff_name
      FROM  staff_master
      WHERE  staff_dob  BETWEEN '01-Jan-1980'
         AND  '31-Jan-1980';
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

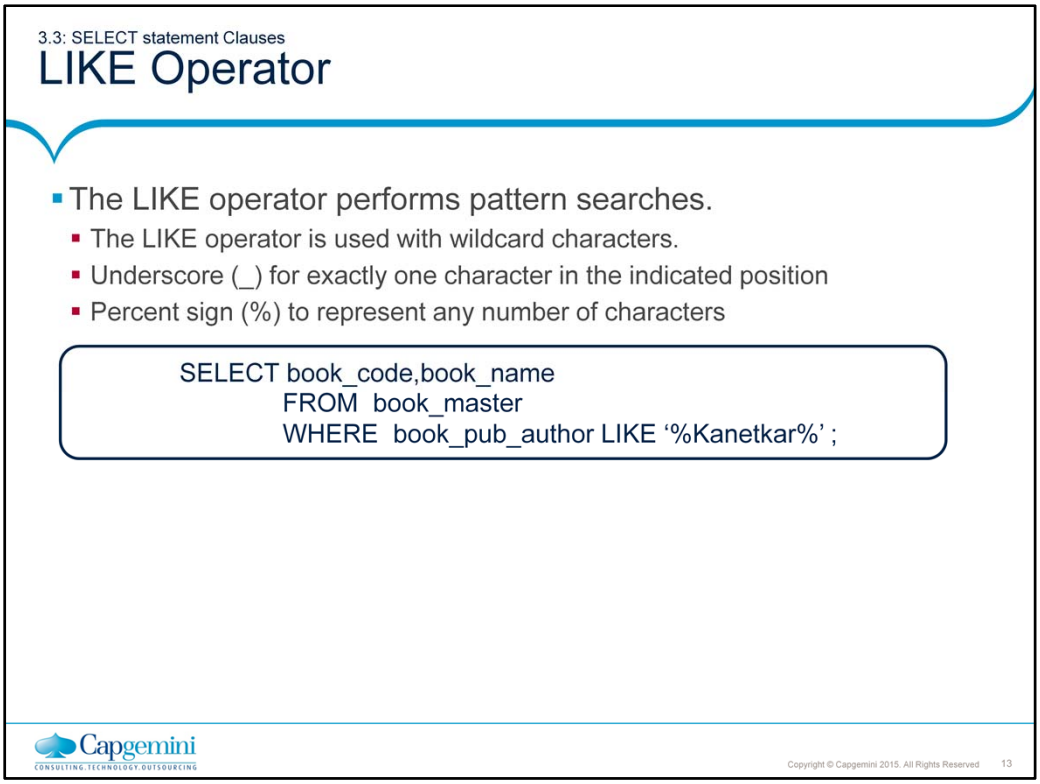
 11



IN predicate:

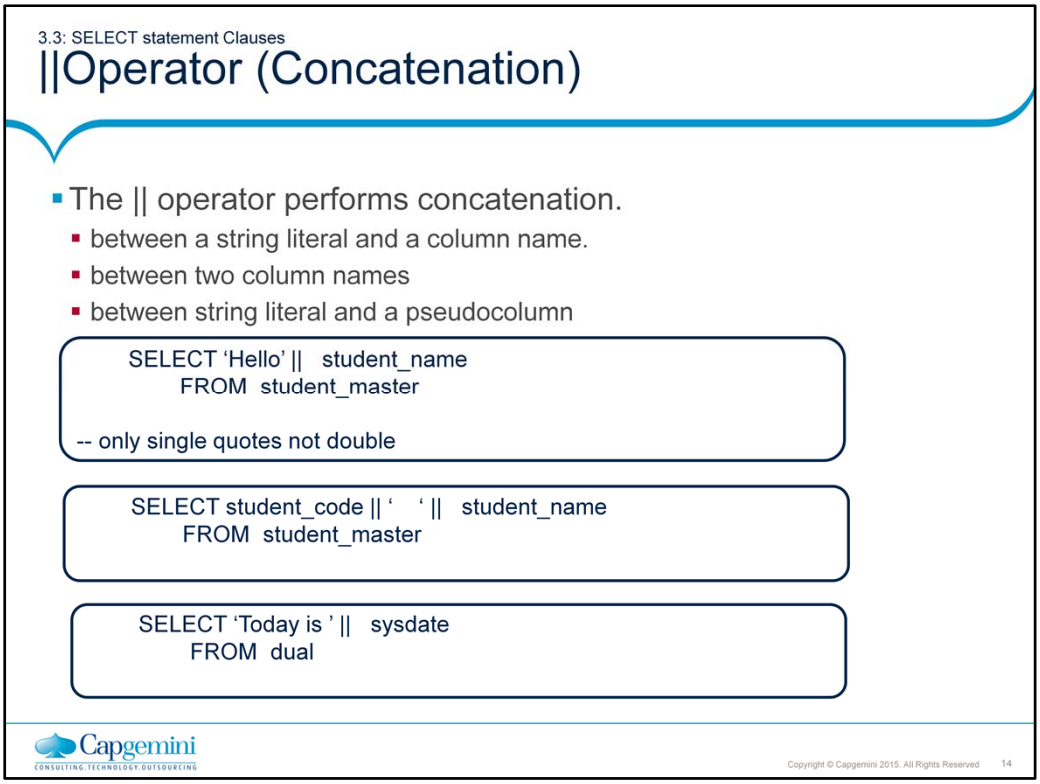
- It is of the form:
  <Expression> IN <LIST>
  <Expression> IN <SUBQUERY>

- The data types should match.



LIKE predicate:

- It is of the form:
  <COLUMN > LIKE < PATTERN>
- The pattern contains a search string along with other special characters % and _. The % character represents a string of any length where as _ (underscore) represents exactly one character.
- A pattern %XYZ% means search has to be made for string XYZ in any position. A pattern '_XYZ%' means search has to be made for string XYZ in position 2 to 4.
- To search for characters % and _ in the string itself we have to use an “escape” character.
  For example: To search for string NOT_APP in column status, we have to use the form Status like 'NOT_APP' ESCAPE '\'
- The use of \ as escape character is purely arbitrary.



Retrieval of Constant values by using Dual Table

A "dual" is a table, which is created by Oracle along with the data dictionary. It consists of exactly one column, whose name is dummy, and one record. The value of that record is X.

```
SQL>desc dual;
Name                          Null?    Type
DUMMY                                  VARCHAR2(1)
Sql>Select * from dual;
D
-
X
```

The owner of dual is SYS. However, "dual" can be accessed by every user.

As "dual" contains exactly one row (unless someone has fiddled with it), it is guaranteed to return exactly one row in SELECT statements.

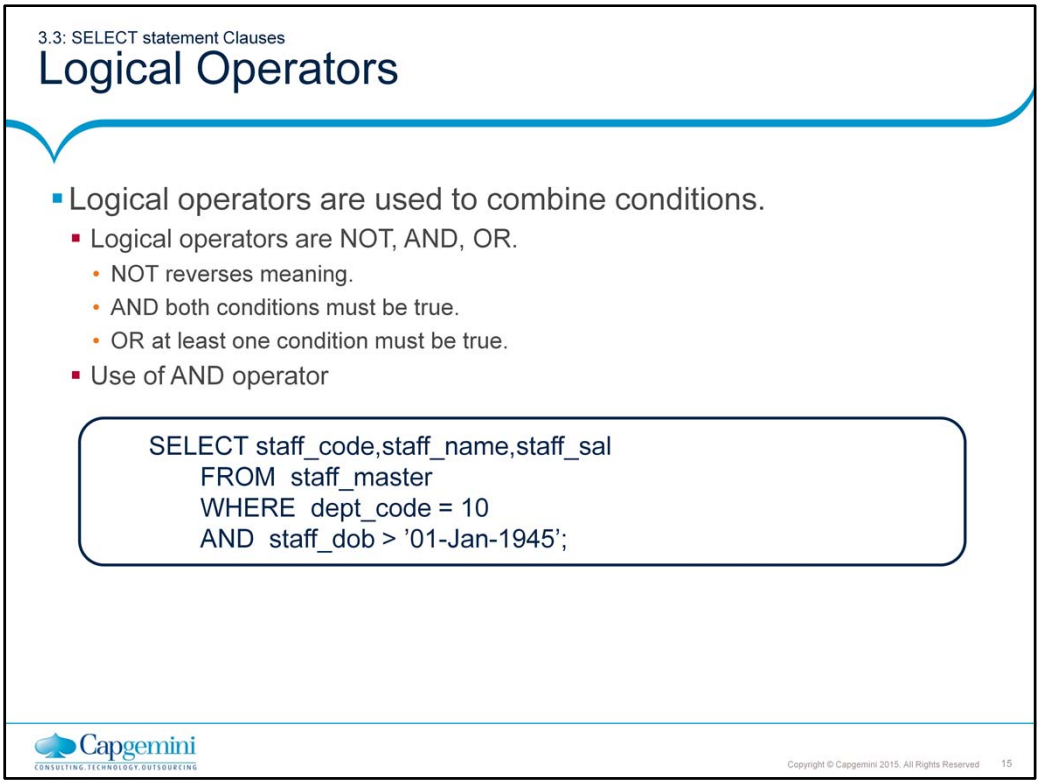
```
SQL>select sysdate from dual;
```

For example, you can use it for math:

```
SQL>SELECT (319/212)+10 FROM DUAL;
```

And, you can use it to increment sequences:

```
SQL>SELECT employee_seq.NEXTVAL FROM
DUAL;
```



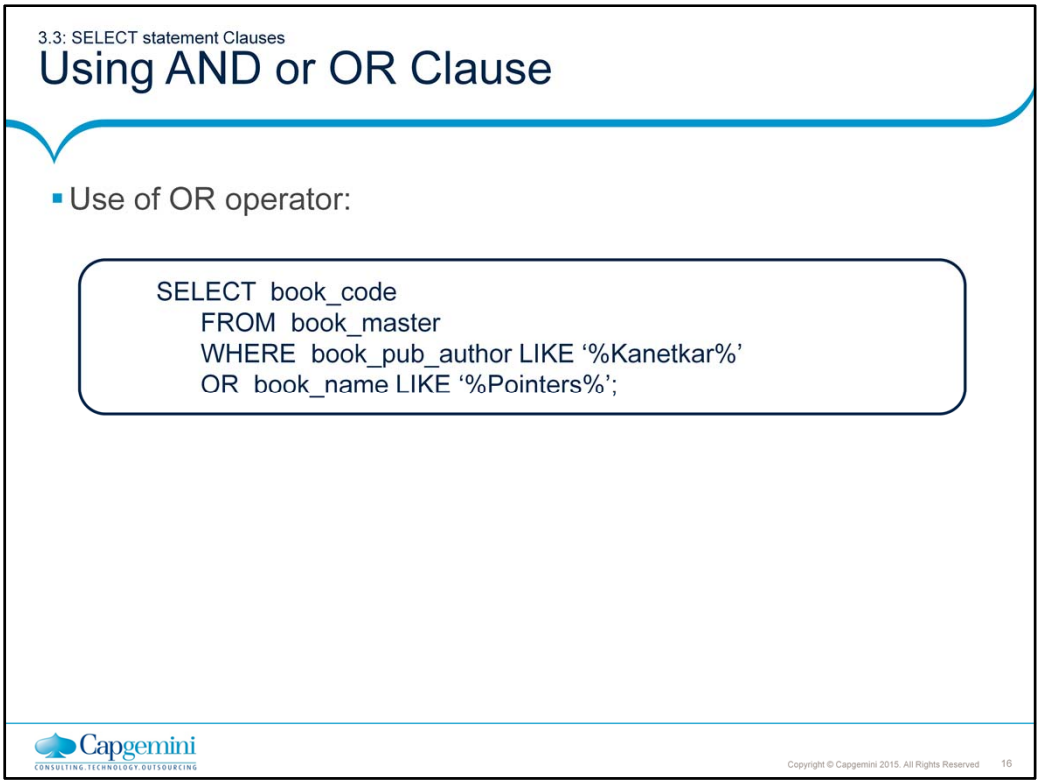
The AND operator displays a record if both the first condition and the second condition is true.

One More Example:

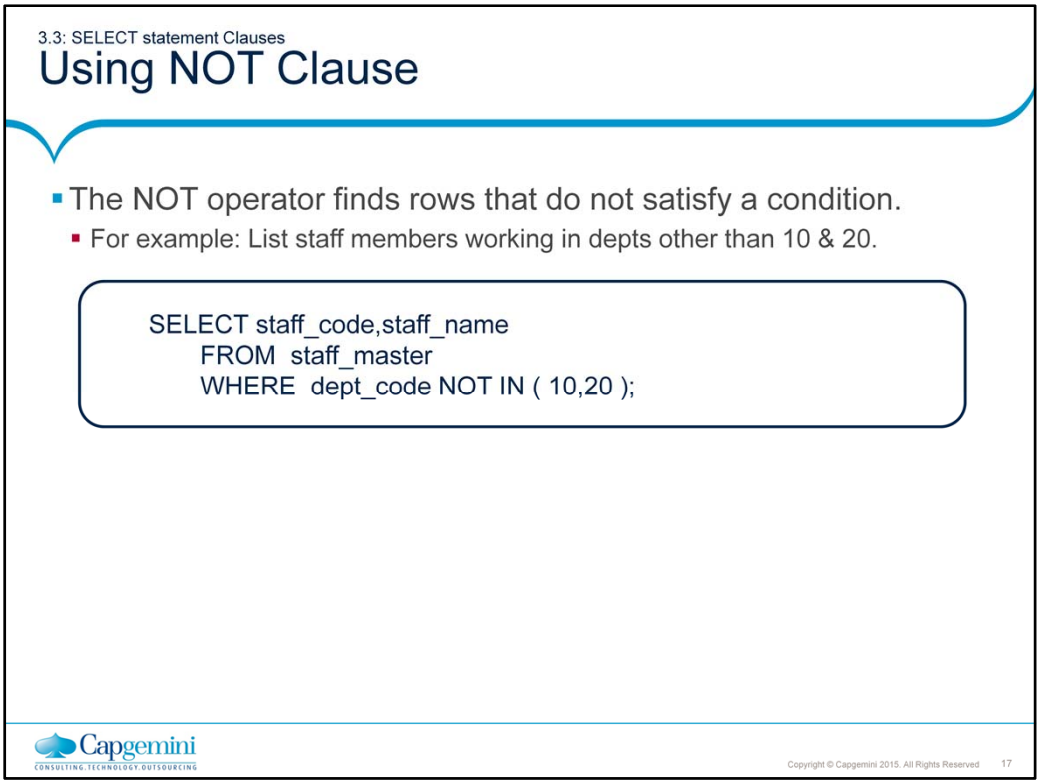
```
SQL>  SELECT title, pubid, category
  2   FROM books
  3   WHERE pubid = 3
  4   AND category = 'COMPUTER';
```

<u>Combining Predicates by using Logical Operators</u>:

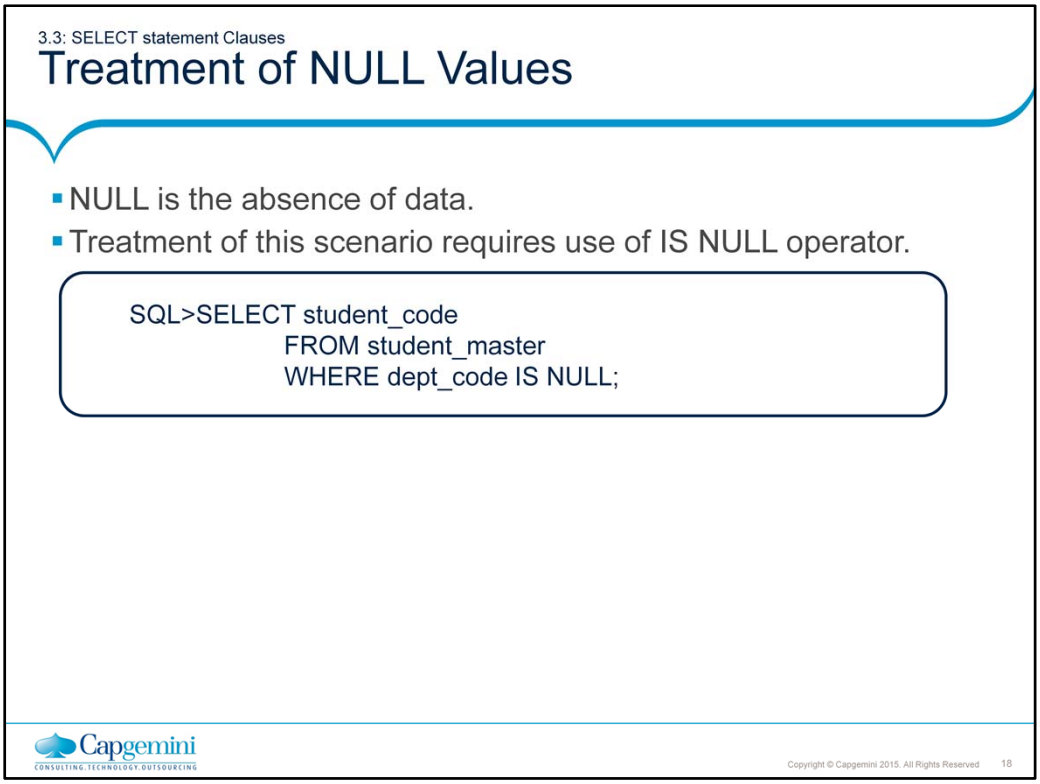
- The predicates can be combined by using logical operators like AND, OR, NOT. The evaluation proceeds from left to right and order of evaluation is:
  - * Enclosed in parenthesis
  - AND
  - OR



The OR operator displays a record if either the first condition or the second condition is true.
You can also combine AND and OR as shown in above example. (use parenthesis to form complex expressions).



•**Note**: NOT is a negation operator.



NULL predicate:

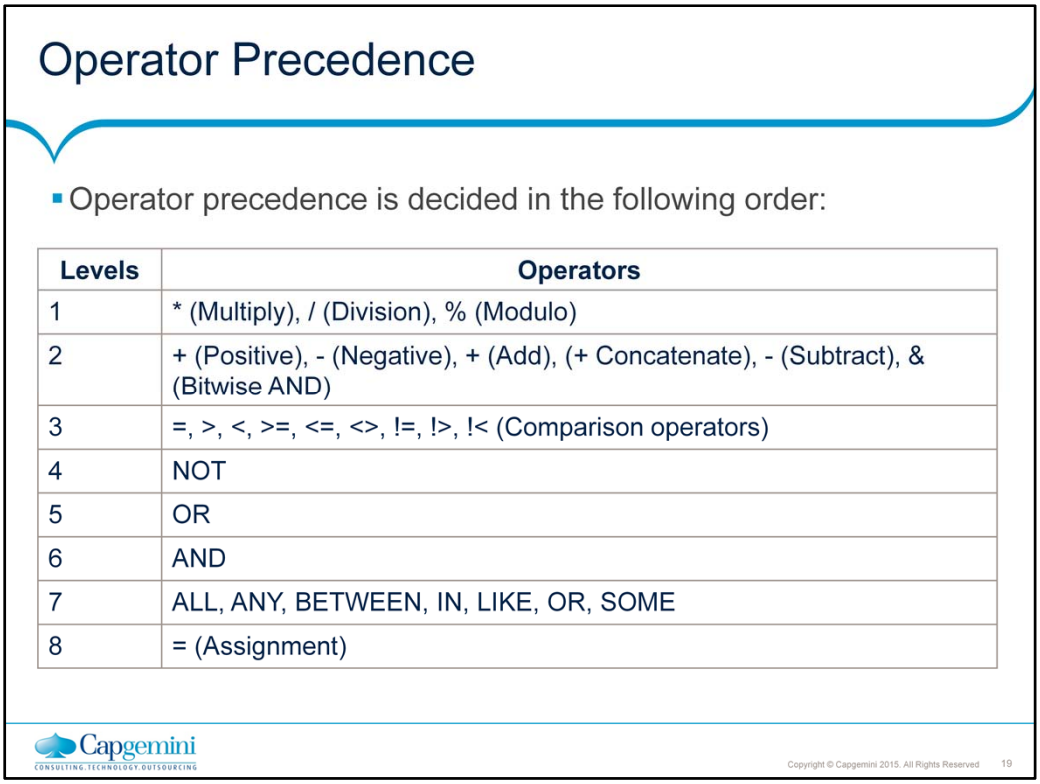
The NULL predicate specifies a test for NULL values. The form for NULL predicate is:

< COLUMN SPECIFICATION > IS NULL.

< COLUMN SPECIFICATION > IS NOT NULL.

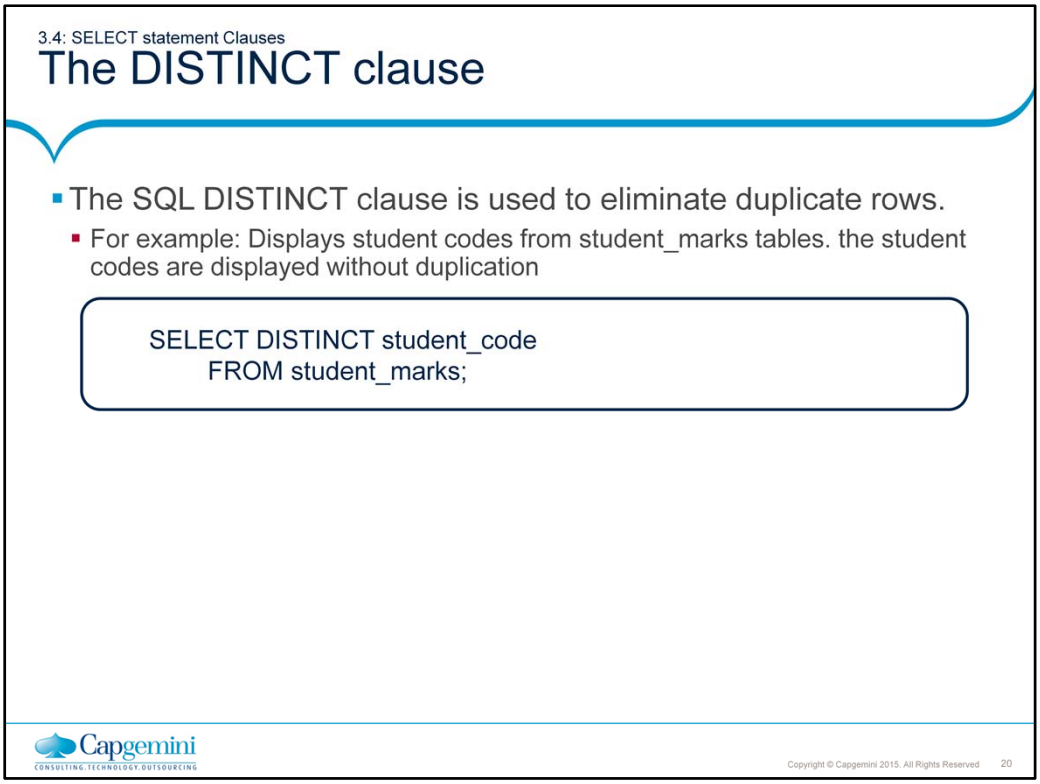
< COLUMN SPECIFICATION > IS NULL returns TRUE only when column has NULL values.

<COLUMN> = NULL cannot be used to compare null values.



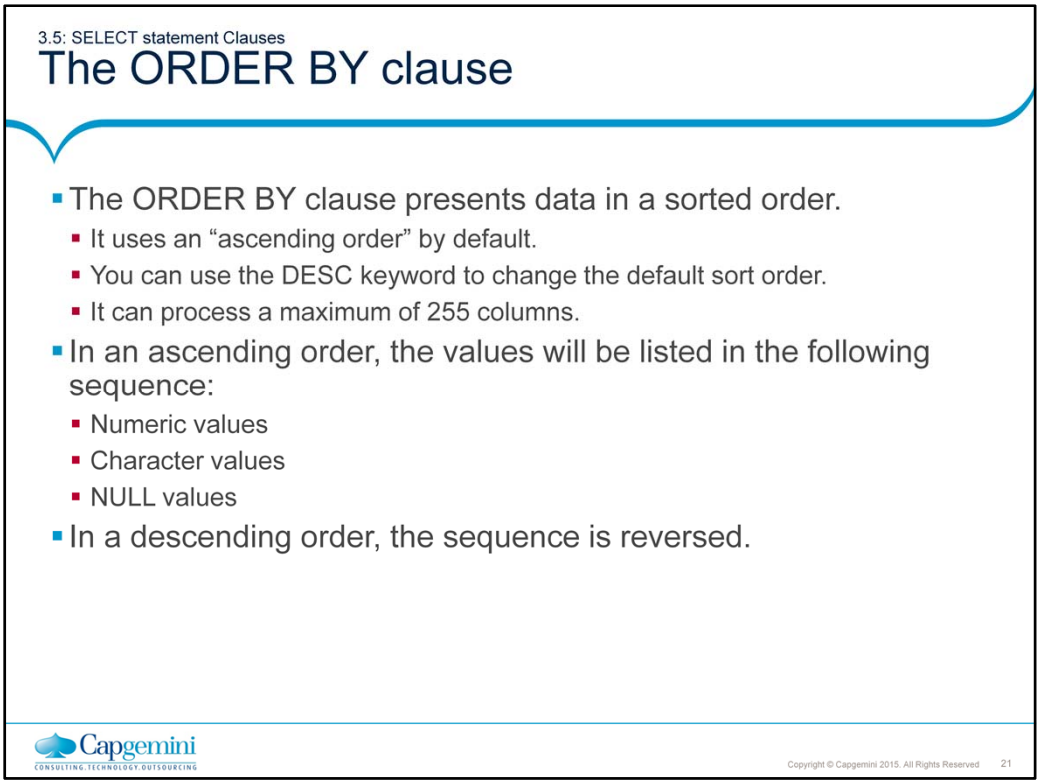
Operator Precedence:

- When a complex expression has multiple operators, the operator precedence (or order of execution of operators) determines the sequence in which the operations are performed.
- The order of execution can significantly affect the resulting value.
- The operators have the precedence levels as shown in the table given in the slide.
- An operator on higher levels is evaluated before an operator on lower level.



The DISTINCT clause:

In the examples discussed so far, some of the values have been repeated. However, by default, all values are retrieved. If you wish to remove duplicate values, then use the query as shown in the slide above.



The Order By Clause:

- A query with its various clauses (FROM, WHERE, GROUP BY, HAVING) determines the rows to be selected and the columns. The order of rows is not fixed unless an ORDER BY clause is given.
- An ORDER BY clause is of the form:

```
ORDER BY < Sort  list> ASC/DESC
```

- The columns to be used for ordering are specified by using the “column names” or by specifying the “serial number” of the column in the SELECT list.
- The sort is done on the column in “ascending” or “descending” order. By default the ordering of data is “ascending” order.

contd.

3.5: SELECT statement Clauses

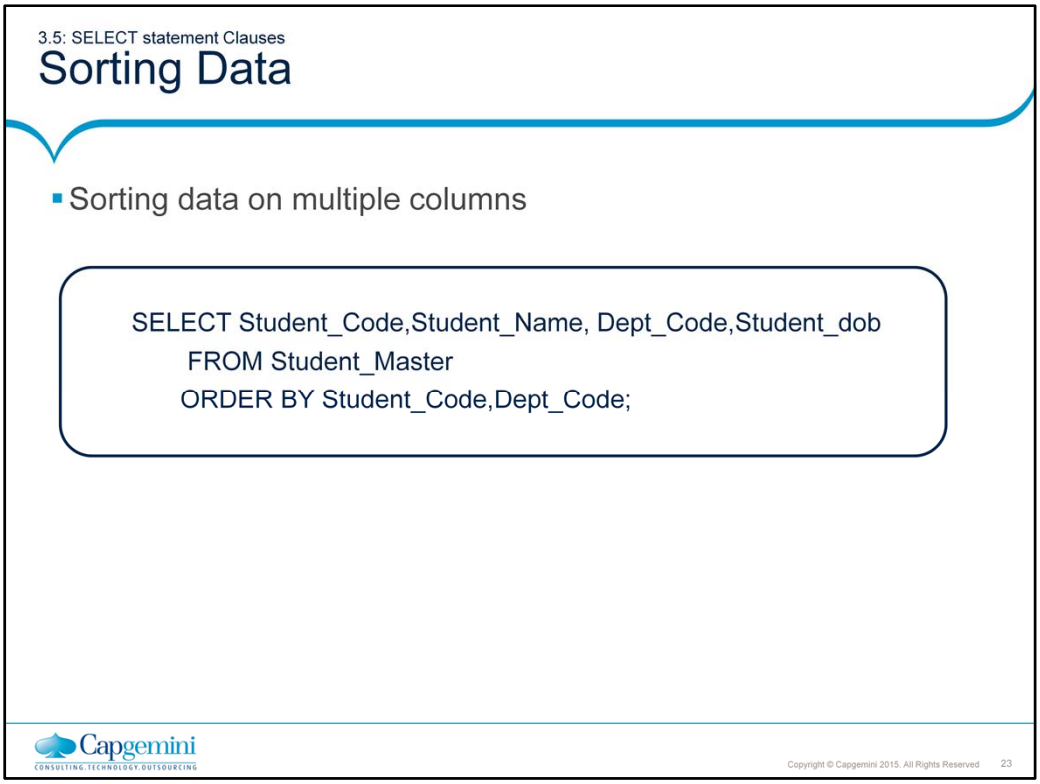
Sorting Data

- The output of the SELECT statement can be sorted using ORDER BY clause
  - ASC : Ascending order, default
  - DESC : Descending order
- Display student details from student_master table sorted on student_code in descending order.

```
SELECT Student_Code,Student_Name,Dept_Code, Student_dob
    FROM Student_Master
    ORDER BY Student_Code DESC ;
```

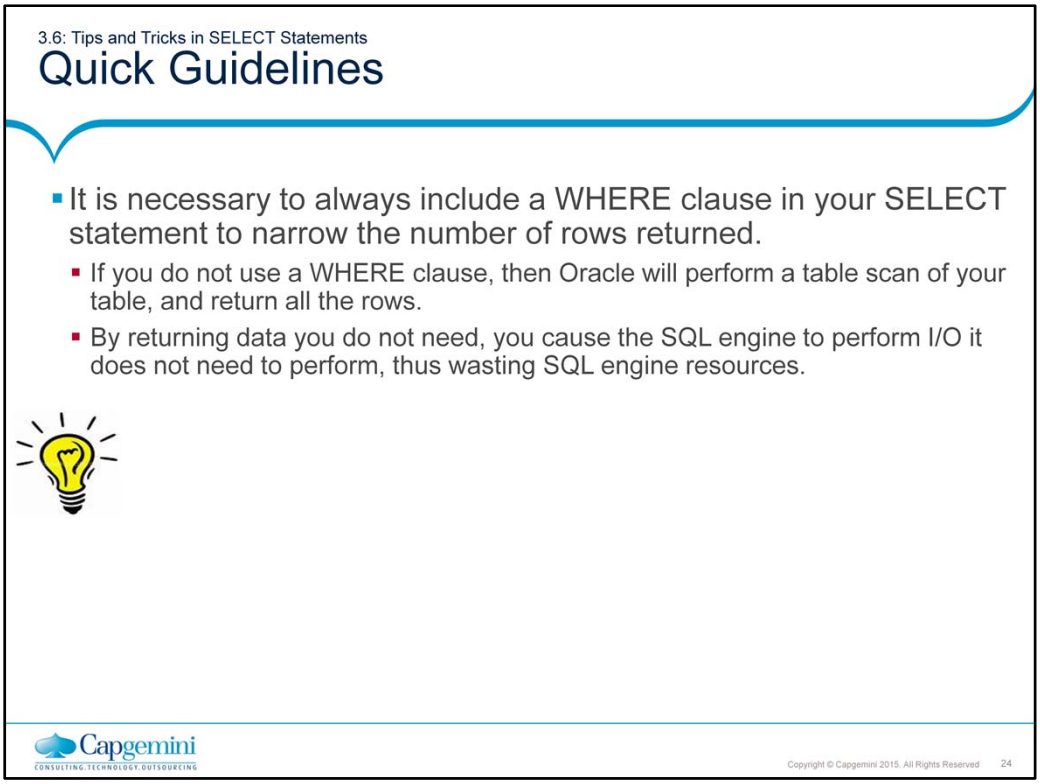
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22



The query on the slide sorts the data on both the columns in ascending order which is default. But you could also sort the data in different order for the columns.
For Example in the query given below the data is sorted in ascending order on student_code and dept_code is sorted in descending order

```
SELECT
Student_Code,Student_Name,Dept_Code,Student_dob
  FROM Student_Master
 ORDER BY Student_Code,Dept_Code DESC;
```



Tips and Tricks in SELECT Statements:

- It is necessary to always include a WHERE clause in your SELECT statement to narrow the number of rows returned.
  - In some case you may want to return all rows. Then not using a WHERE clause is appropriate in this case.
  - However, if you don't need all the rows to be returned, use a WHERE clause to limit the number of rows returned.
    - Another negative aspect of a table scan is that it will tend to flush out data pages from the cache with useless data. This reduces ability of the Oracle to reuse useful data in the cache, which increases disk I/O and decreases performance.

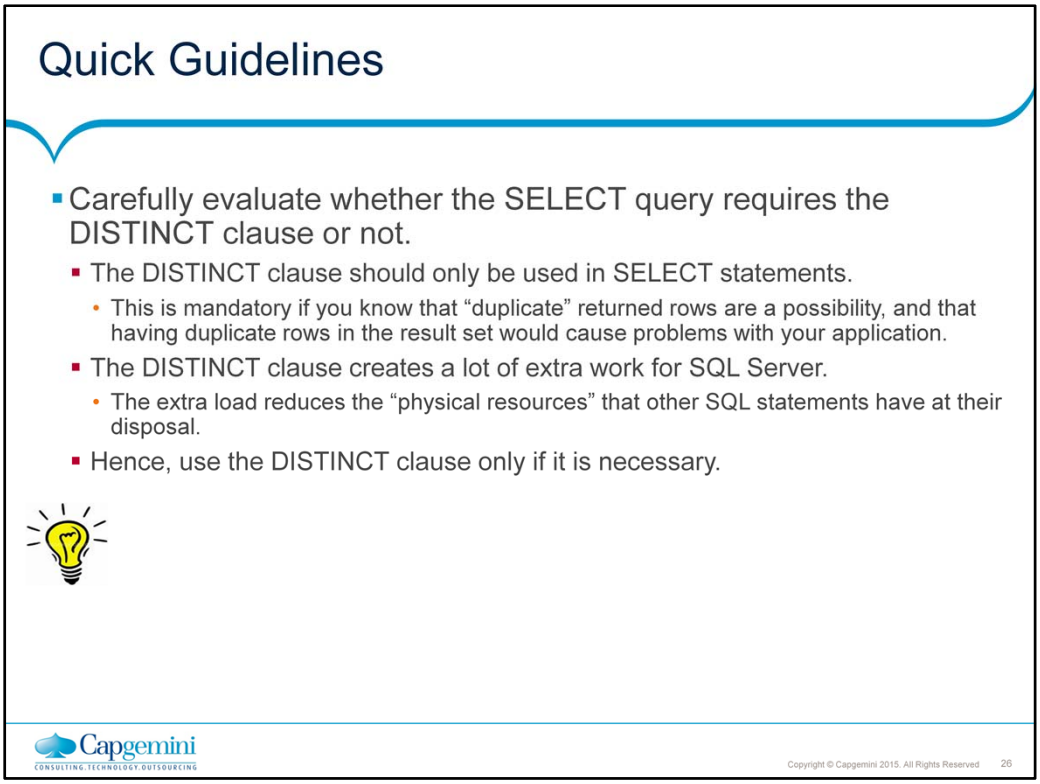
Quick Guidelines

- In addition, the above scenario increases network traffic, which can also lead to reduced performance.
- And if the table is very large, a table scan will lock the table during the time-consuming scan, preventing other users from accessing it, and will hurt concurrency.

- In your queries, do not return column data that is not required.
  - For example:
    - You should not use SELECT * to return all the columns from a table if all the data from each column is not required.
    - In addition, using SELECT * prevents the use of covered indexes, further potentially decreasing the query performance.

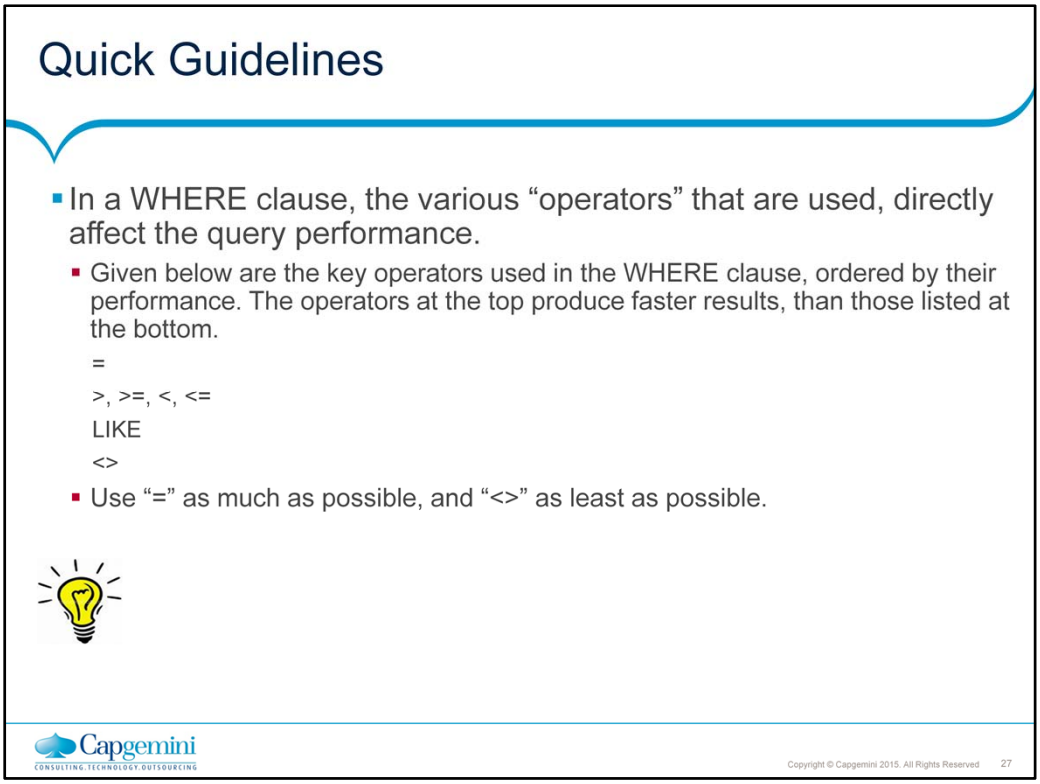
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25



Tips and Tricks in SELECT Statements (contd.):

- Some developers, as a habit, add the DISTINCT clause to each of their SELECT statements, even when it is not required.
  - ➢ This is a bad habit that should be stopped.



Tips and Tricks in SELECT Statements (contd.):

Use simple operands

- Some operators tend to produced speedy results than other operators. Of course, you may not have choice of using an operator in your WHERE clauses, but sometimes you do have a choice.
  - Using simpler operands, and exact numbers, provides the best overall performance.
  - If a WHERE clause includes multiple expressions, there is generally **no** performance benefit gained by ordering the various expressions in any particular order.
    - This is because the Query Optimizer does this for you, saving you the effort. There are a few exceptions to this, which are discussed further in the lesson.

contd.

Tips and Tricks in SELECT Statements (contd.):

Don't include code that does not do anything

- This may sound obvious. However, this scenario is seen in some off-the-shelf applications.
  - For example, you may see code which is given below:

```
SELECT column_name FROM table_name  WHERE 1 = 0
```

  - When this query is run, no rows will be returned. It is just wasting SQL Server resources.

- By default, some developers routinely include code, which is similar to the one given above, in their WHERE clauses when they make string comparisons.
  - For example:

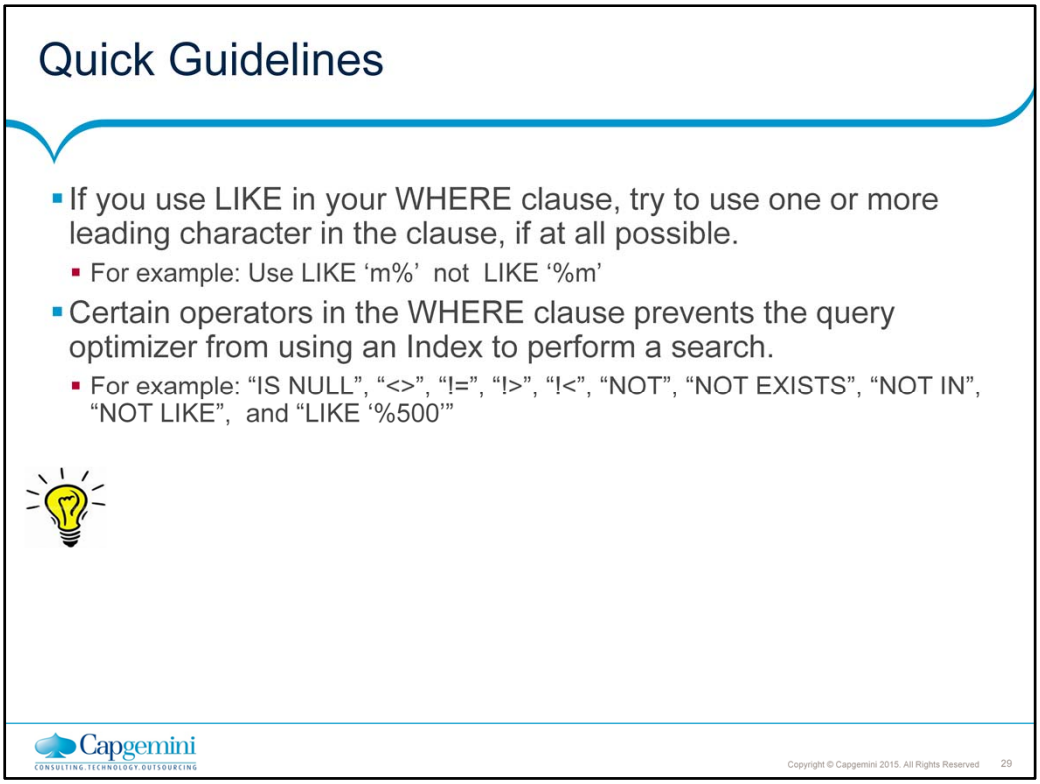
```
SELECT column_name FROM table_name
    WHERE LOWER(column_name) = 'name'
```

Any use of text functions in a WHERE clause decreases performance.

- If your database has been configured to be case-sensitive, then using text functions in the WHERE clause does decrease performance. However, in such a case, use the technique described below, along with appropriate indexes on the column in question:

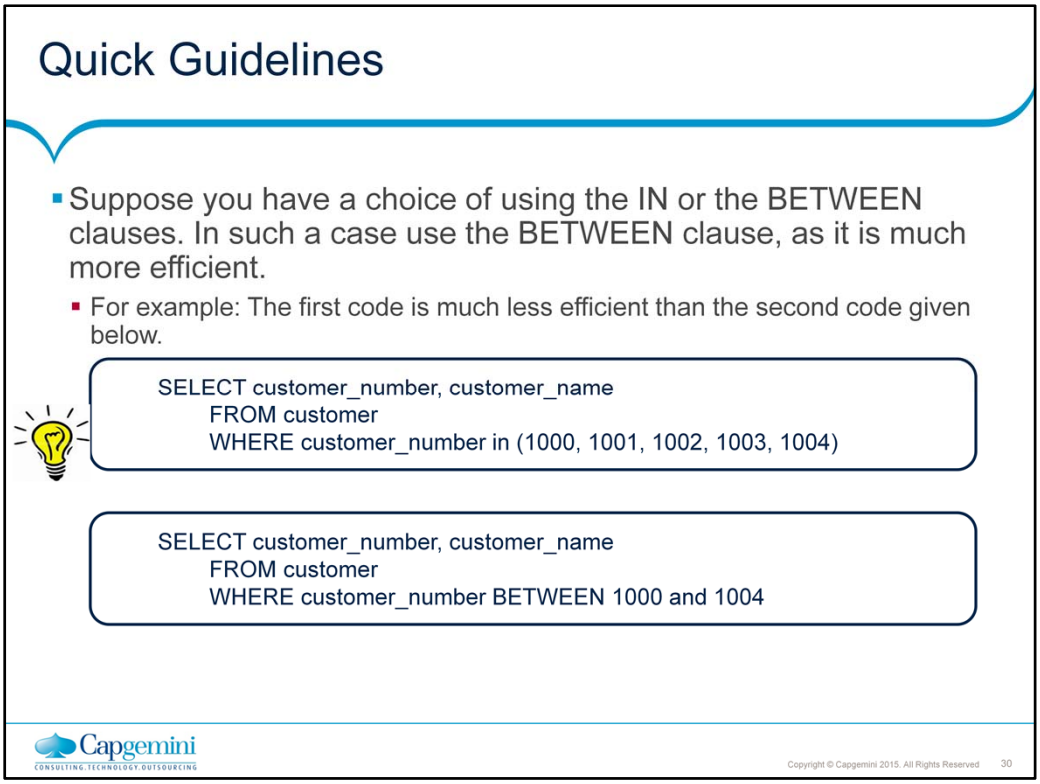
```
SELECT column_name FROM table_name
WHERE column_name = 'NAME' or column_name = 'name'
```

- This code will run much faster than the first example.



Tips and Tricks in SELECT Statements (contd.):

- If you use a leading character in your LIKE clause, then the Query Optimizer has the ability to potentially use an Index to perform the query. Thus speeding performance and reducing the load on SQL engine.
  - However, if the leading character in a LIKE clause is a "wildcard", then the Query Optimizer will not be able to use an Index. Here a table scan must be run, thus reducing performance and taking more time.
- The more leading characters you use in the LIKE clause, it is more likely that the Query Optimizer will find and use a suitable Index.

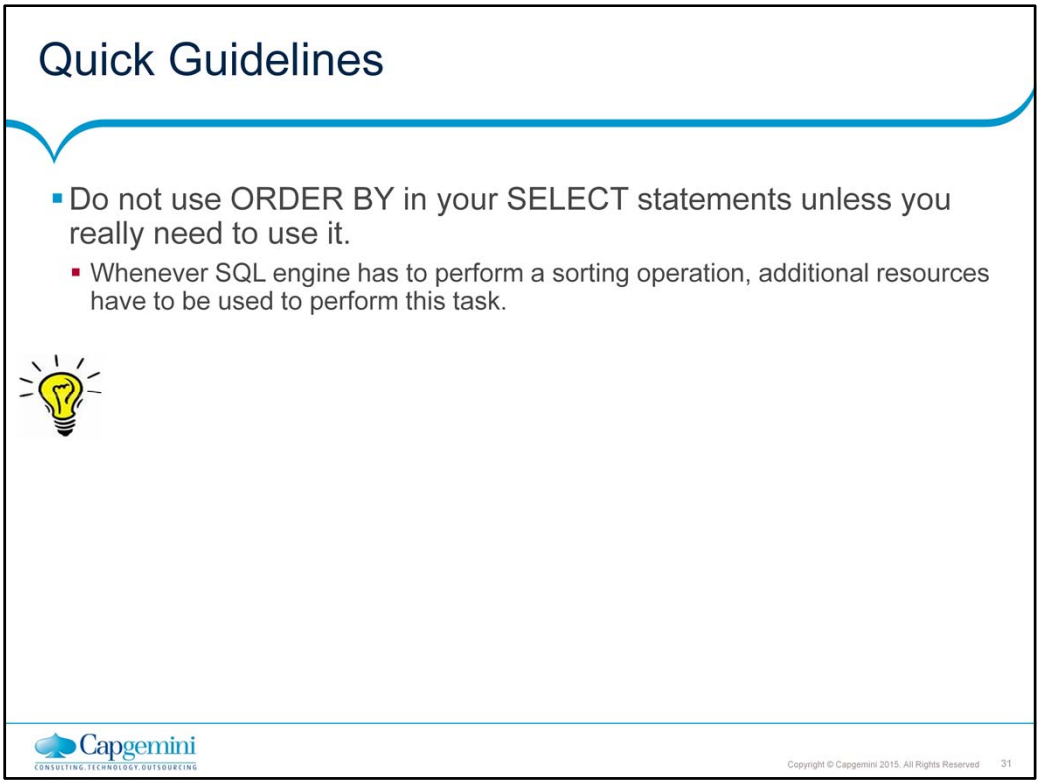


Tips and Tricks in SELECT Statements (contd.):

- Assuming there is a useful Index on customer_number, the Query Optimizer can locate a range of numbers much faster by using BETWEEN clause.
  - This is much faster than it can find a series of numbers by using the IN clause (which is really just another form of the OR clause).

Using Efficient Non-index WHERE clause sequencing:

- Oracle evaluates un-indexed equations, linked by the AND verb in a bottom-up fashion. This means that the first clause (last in the AND list) is evaluated, and if it is found TRUE, then the second clause is tested.
- Always try to position the most expensive clause first in the WHERE clause sequencing.
- Oracle evaluates un-indexed equations, linked by the OR verb in a top-down fashion. This means that the first clause (first in the OR list) is evaluated, and if it is found FALSE, then the second clause is tested.
- Always try to position the most expensive OR clause last in the WHERE clause sequencing.



Tips and Tricks in SELECT Statements (contd.):

Don't use ORDER BY in your SELECT statements unless you really need to:

- The ORDER BY clause adds a lot of extra overhead.
  For example: Sometimes it may be more efficient to sort the data at the client than at the server. In other cases, the client does not even need sorted data to achieve its goal. The key here is to remember that you should not automatically sort data, unless you know it is necessary.
- Whenever SQL Server has to perform a sorting operation, additional resources have to be used to perform this task. Sorting often occurs when any of the following Transact-SQL statements are executed:
  - ORDER BY
  - GROUP BY
  - SELECT DISTINCT
  - UNION
  - CREATE INDEX (generally not as critical as happens much less often)
- In many cases, these commands cannot be avoided. On the other hand, there are few ways in which sorting overhead can be reduced, like:
  - Keep the number of rows to be sorted to a minimum. Do this by only returning those rows that absolutely need to be sorted.
  - Keep the number of columns to be sorted to the minimum. In other words, do not sort more columns than required.
  - Keep the width (physical size) of the columns to be sorted to a minimum.
  - Sort column with number datatypes instead of character datatypes.

Summary

- In this lesson, you have learnt:
  - What is SELECT statement?
  - Usage of the following:
    - The WHERE clause
    - The Mathematical, Comparison, and Logical operators
    - The AND or OR clause
    - The NOT clause
    - The DISTINCT clause
    - The ORDER BY clause

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 32

Review – Questions

- Question 1: The ___ table consists of exactly one column, whose name is “dummy”.
- Question 2: The LIKE operator comes under the ___ category.
  - Option 1: mathematical
  - Option 2: comparison
  - Option 3: logical
- Question 3: The ___ specifies the order in which the operators should be evaluated.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 33

Review – Questions

- Question 4: The NOT NULL operator finds rows that do not satisfy a condition.
  - True / False
- Question 5: More than one column can also be used in the ORDER BY clause.
  - True / False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 34

DBMS/SQL

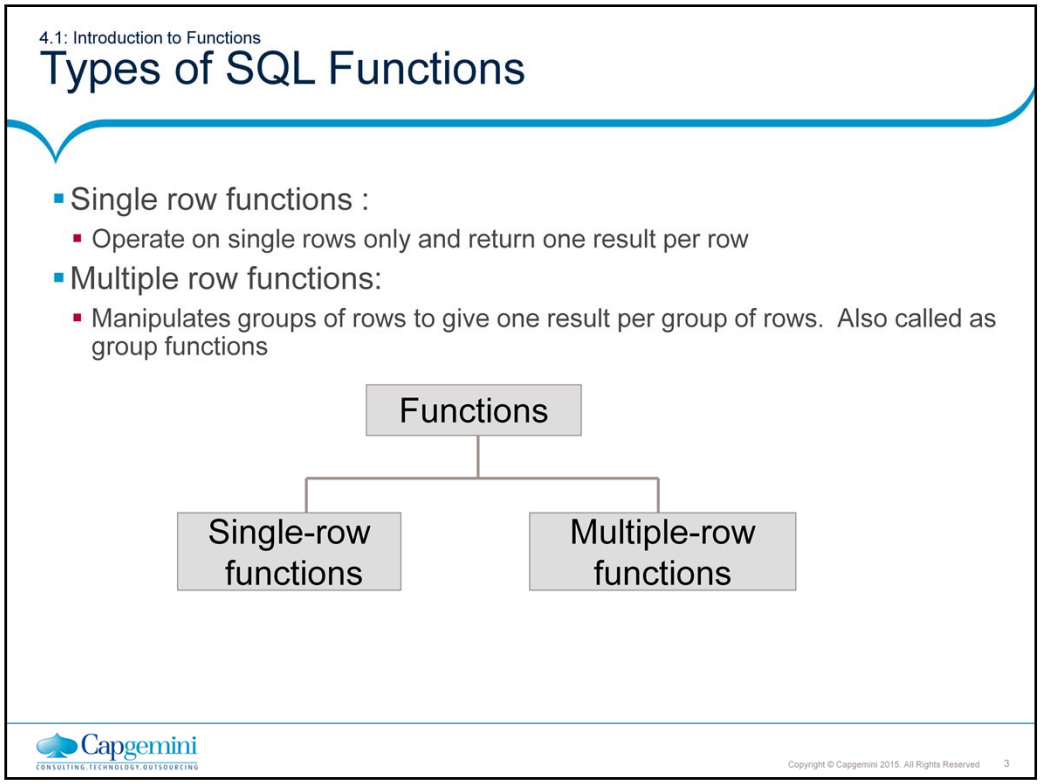
Lesson 04: Aggregate (GROUP) Functions

Lesson Objectives

- To understand the following topics:
  - Introduction to Functions
  - Aggregate (Group) functions:
    - GROUP BY clause
    - HAVING clause

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



Functions:

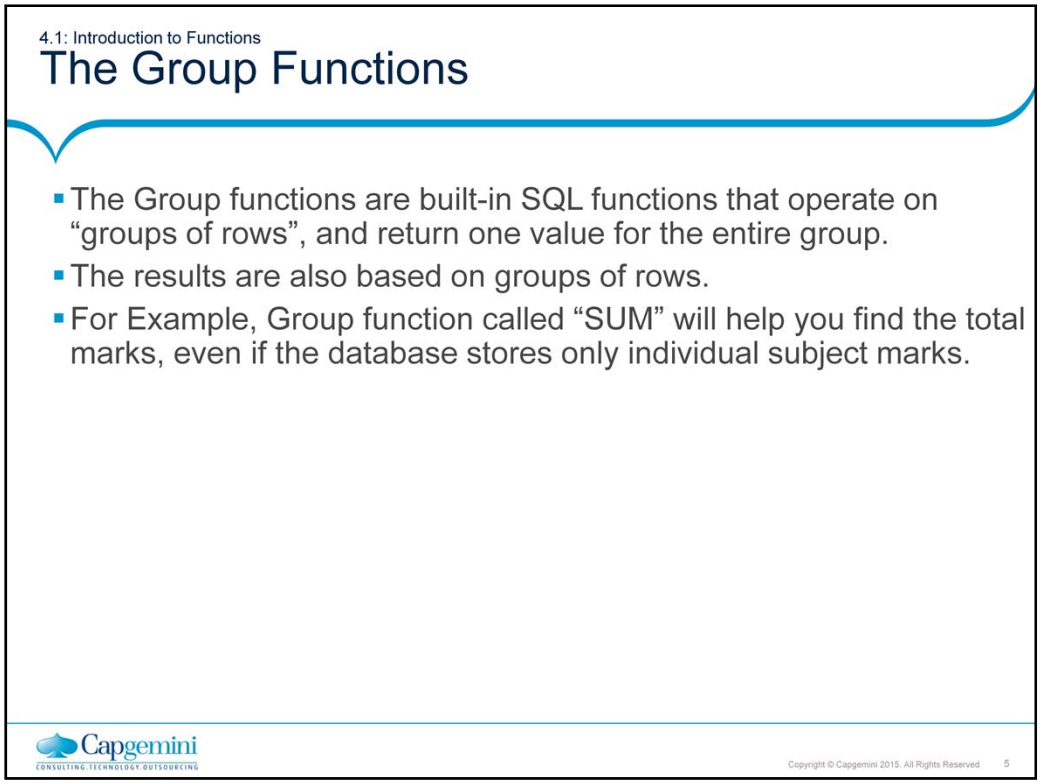
- Functions can be used to manipulate data values in a variety of ways.
  - Functions help in making the basic query block more powerful.
  - Functions may be used to perform calculations on data, alter data formats for display, convert data types, etc.
- Functions are similar to operators in that they manipulate data items and return a result.
- Functions differ from operators in the format in which they appear with their arguments. This format allows them to operate on zero, one, two, or more arguments:
  - function(argument, argument, ...)
- There are two types of functions: SQL functions and User-defined functions.
  1. **SQL functions**: SQL functions are built into most DBMS and are available for use in various appropriate SQL statements. SQL functions are further categorized as:
     a) **Single-Row functions**: Single-row functions return a single result row for every row of a queried Table or View.
     For example: ABS, ROUND etc.

b) **Aggregate functions**: Aggregate functions return a single row based on groups of rows, rather than on single rows. **For example**: MAX, MIN, COUNT, etc.

2. **User-defined functions**: You can write user-defined functions in PL/SQL or Java to provide functionality that is not available in SQL or SQL functions. User-defined functions can appear in a SQL statement wherever SQL functions can appear, that is, wherever an expression can occur.

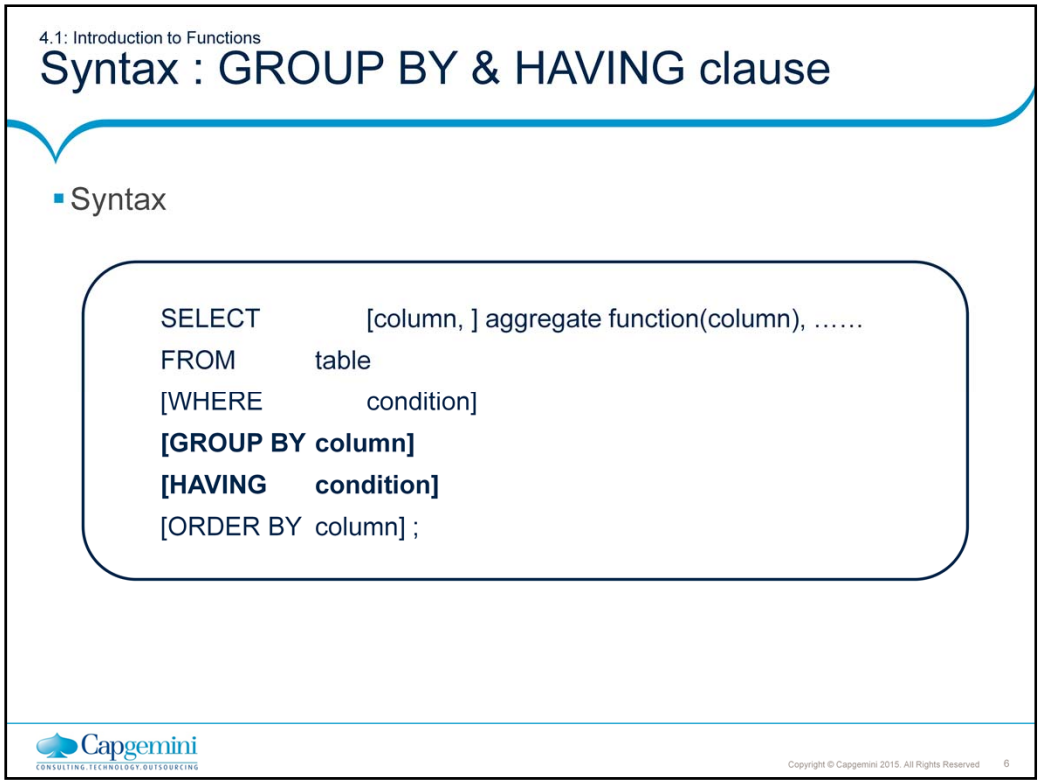
For example: User-defined functions can be used in the following:

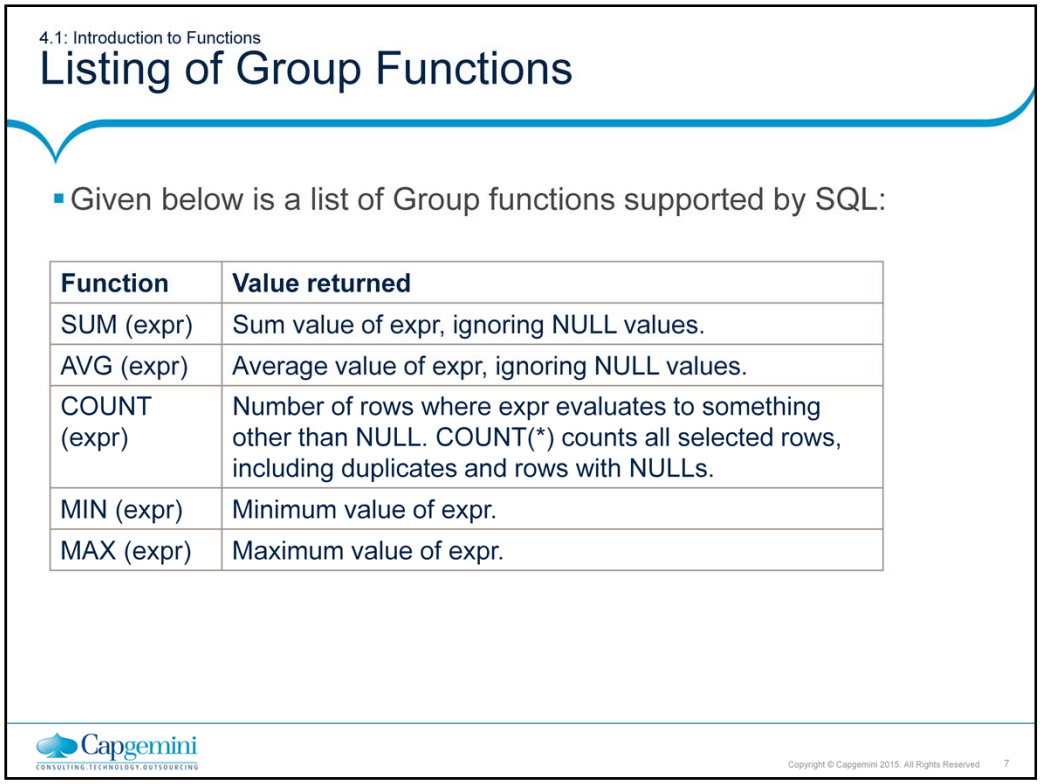
a) The select list of a SELECT statement
b) The condition of a WHERE clause
c) ORDER BY and GROUP BY clauses
d) The VALUES clause of an INSERT statement
e) The SET clause of an UPDATE statement



Aggregate (Group) Functions:

- SQL provides a set of "built-in" functions for producing a single value for an entire group. These functions are called as "Set functions" or "Aggregate (Group) functions".
- These functions can work on a "normal result table" or a "grouped result table".
  - If the result is not grouped, then the aggregate will be taken for the whole result table.


4.1: Introduction to Functions
Syntax : GROUP BY & HAVING clause
Syntax
SELECT [column,] aggregate function(column), ……
FROM table
[WHERE condition]
[GROUP BY column]
[HAVING condition]
[ORDER BY column] ;
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 6



Aggregate (Group) Functions supported by SQL:

- All the above functions operate on a number of rows (for example, an entire table), and are therefore known as “Group (or Aggregate) functions”.
- A Group function can be used on a subset of the rows in a table by using the WHERE clause.
- The Aggregate functions ignore NULL values in the column.
  - To include NULL values, NVL function can be used with Aggregate functions.

Note :

- **Count(*)**: Returns the number of rows in the table, including duplicates and those with NULLs.
- **Count(<Expression>)**: Returns the number of rows where expression is NOT NULL.

Aggregate (Group) Functions supported by SQL (contd.):

SUM(COL_NAME | EXPRESSION)

- SUM returns the total of values present in a particular "column" or a "number of columns" that are linked together in the expression. All the columns, which form the argument to SUM, must be numeric only.
- To find the sum of one subject for all students following query is used:

```
SELECT SUM(subject1)
        FROM student_marks;
```

- To find the yearly compensation of staff from department 20, the following query is used:

```
SELECT SUM(12*staff_sal)
        FROM staff_master
        WHERE dept_code = 20;
```

AVG(COL_NAME | EXPRESSION)

- AVG is similar to SUM. AVG returns the average of a NUMBER of values. The restrictions, which apply on SUM also, apply on AVG.
- To find the average salary the staff, the following query is used:

```
SELECT AVG(sal)
        FROM staff_master;
```

- To find the average yearly compensation of staff from department 20, the following query is used:

```
SELECT AVG(12*staff_sal)
        FROM staff_master
        WHERE dept_code = 20;
```

Aggregate (Group) Functions supported by SQL (contd.):

COUNT(*)

- COUNT returns the number of rows.
- To find the total number of staff members, the following query is used:

```
SELECT COUNT(*)
    FROM staff_master;
```

  - It is possible to restrict the rows for the operation of COUNT.
- To find the total number of staff members in department 10, the following query is used:

```
SELECT COUNT(*)
    FROM staff_master
    WHERE dept_code = 10 ;
```

- To find the total number of staff members hired after '01-JAN-02', the following query is used:

```
SELECT COUNT(*)
    FROM staff_master
    WHERE hiredate >'01-JAN-02' ;
```

  - When an aggregate function is used in a SELECT statement, column names cannot be used in SELECT unless GROUP BY clause is used.

MIN(COL_NAME | EXPRESSION)

- MIN returns the lowest of the values from the column. MIN accepts columns, which are NON-NUMERIC too.
- To find the minimum salary paid to a staff member, the following query is used:

```
SELECT MIN(staff_sal)
    FROM staff_master;
```

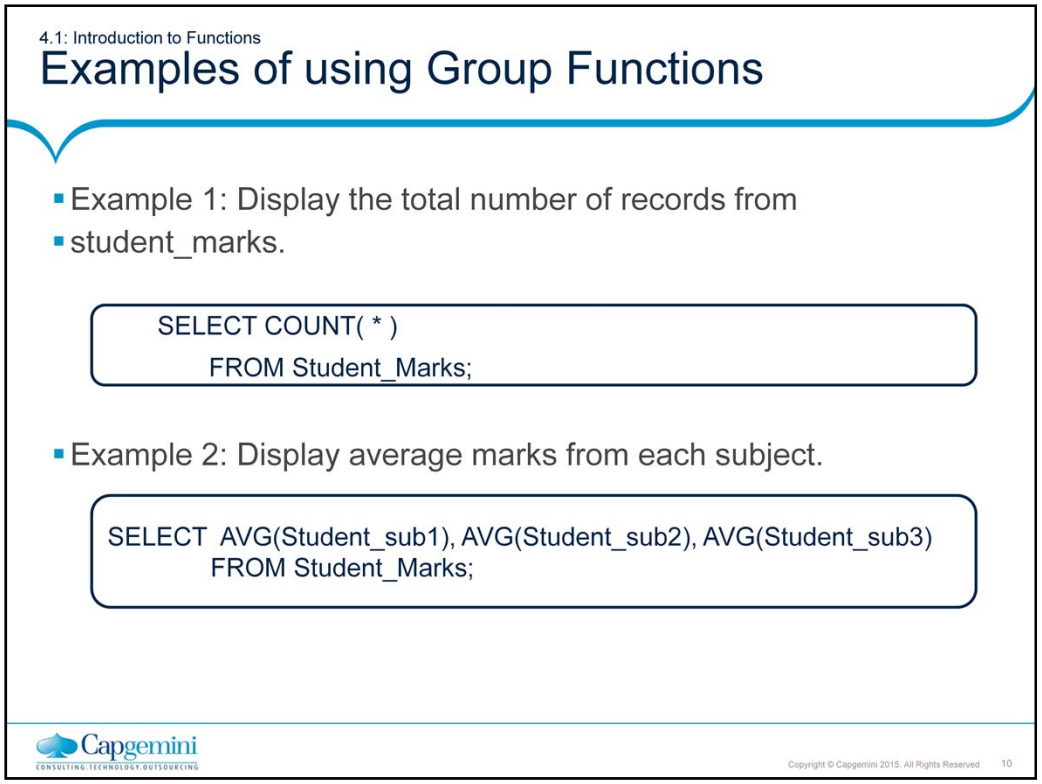
- To list the staff member who alphabetically heads the list, the following query is used:

```
SELECT MIN(staff_name)
    FROM staff_master;
```

```
SELECT MAX(subject2)
    FROM student_marks ;
```

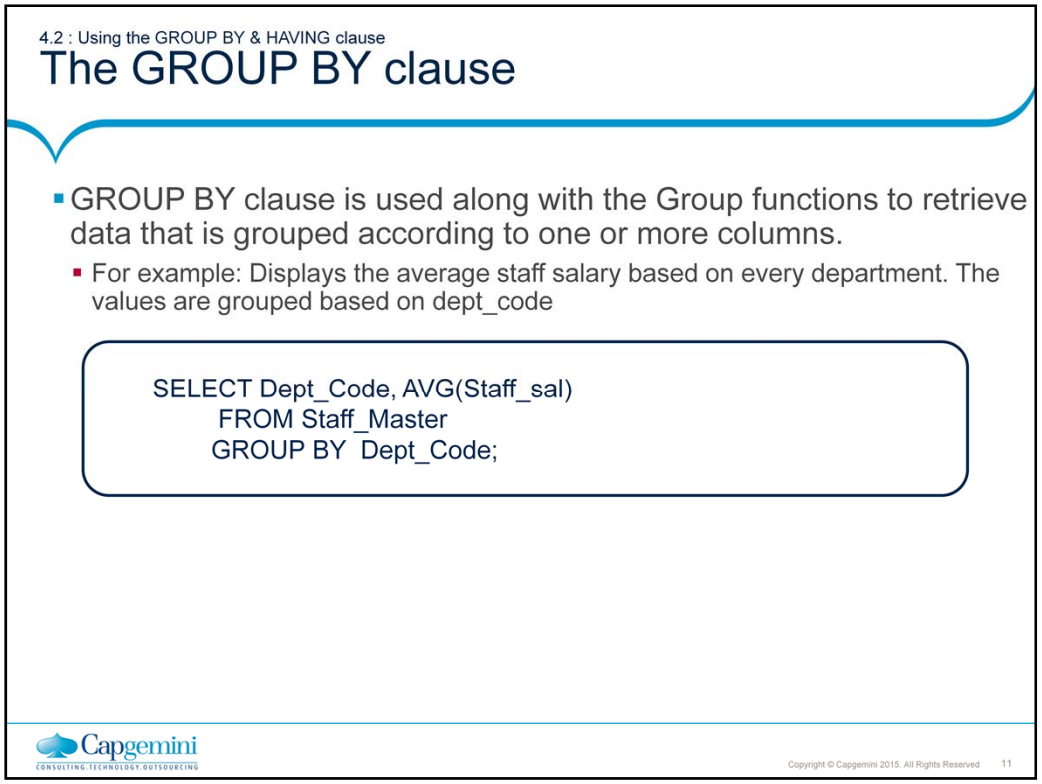
MAX(COL_NAME | EXPRESSION)

- MAX is the reverse of MIN. MAX returns the maximum value from among the list of values.
- To find the maximum marks for a student in subject2, the following query is used:



Note:

- The first query returns the value, which counts the number of rows fetched from the student_marks table. All the rows in thetable are treated as one group.Group Functions operate on sets of rows to give one result per group. Can be used on the whole table or certain set of rows



Usage of GROUP BY and HAVING clauses:

- All the SELECT statements we have used until now have acted on data as if the data is in a "single group". But the rows of data in some of the tables can be thought of as being part of "different groups".
  For example: The staff_master table contains staff information allocated to various departments identified by dept_code. If we wish to find the minimum salary of each group of employees in respective department, then none of the clauses that we have seen until now are of any use.
- The GROUP BY clause is used to group the result table derived from earlier FROM and WHERE clauses. Further, a HAVING clause is used to apply search condition on these groups.
  - When a GROUP BY clause is used, each row of the resulting table will represent a group having same values in the column(s) used for grouping.
  - Subsequently, the HAVING clause acts on the resulting grouped table to remove the row that does not satisfy the criteria in the HAVING search condition

The GROUP BY Clause:

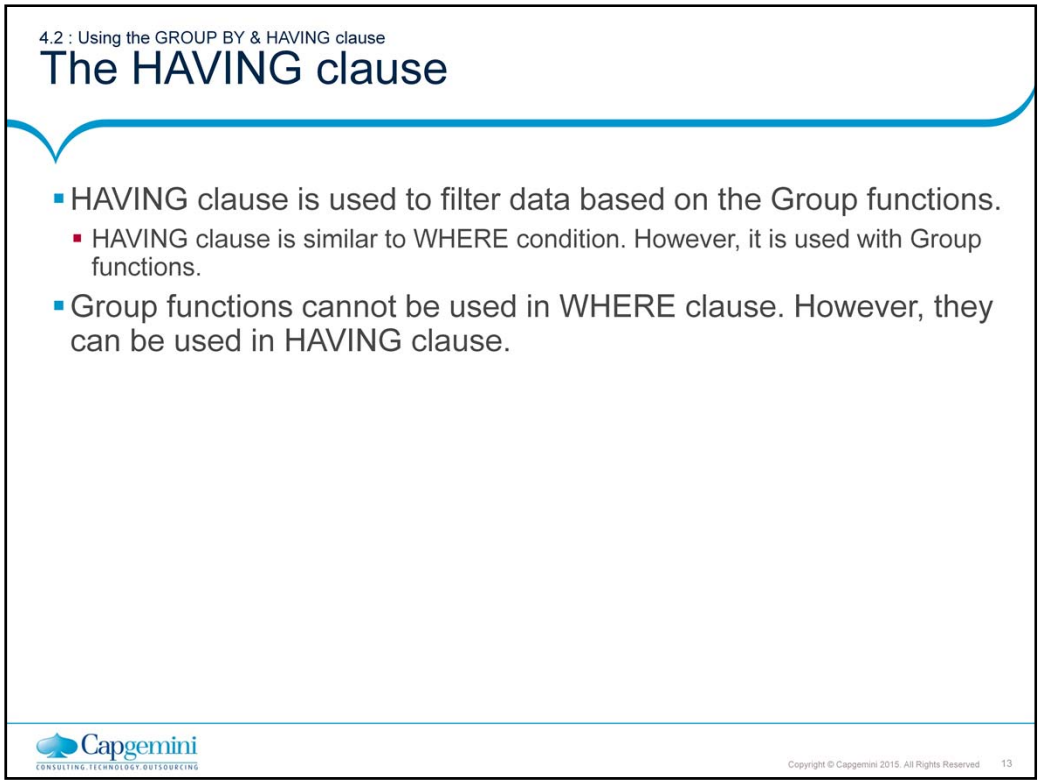
If you have to fetch columns other than group functions or if you want your results to be grouped on certain criteria use “GROUP BY” clause

- The GROUP BY clause is of the form:

```
GROUP BY <column  list>
```

- The columns specified must be selected in the query. If a table is grouped, the SELECT list should have columns and expressions, which are single-valued for a group. These can be:
  - columns on which grouping is done
  - constants
  - aggregate functions on the other columns on which no grouping is done

1. The GROUP BY clause should contain all the columns in the SELECT list, except those used along with the Group functions. Only the column names which have been used in GROUP BY clause and aggregate columns can be used in SELECT clause
2. When an Aggregate (Group) function is used in a SELECT statement, the column names cannot be used in SELECT, unless GROUP BY clause is used.
3. Grouping can be done on multiple columns, as well.

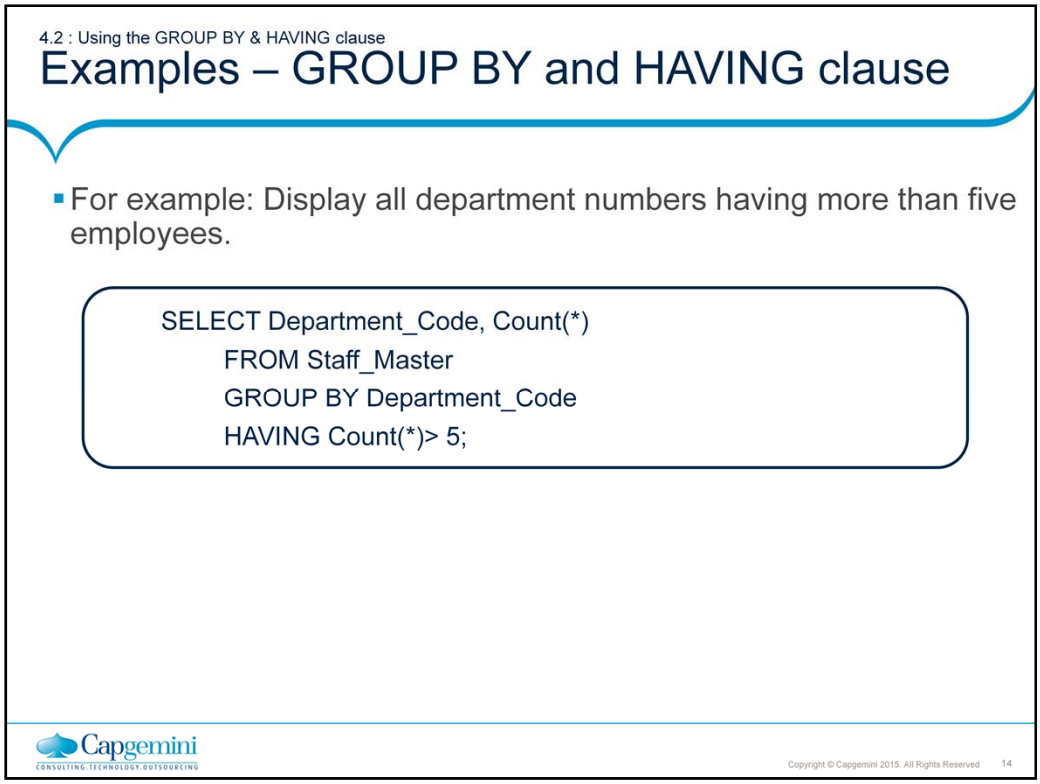


The HAVING Clause:

- A HAVING clause is of the form:

  HAVING <search condition>

- The HAVING search condition applies to "each group". It can be:
  - formed using various predicates like between, in, like, null, comparison, etc
  - combined with Boolean operators like AND, OR, NOT
- Since the search condition is for a "grouped table". The predicates should be:
  - on a column by which grouping is done.
  - on a set function (Aggregate function) on other columns.
- The aggregate functions can be used in HAVING clause. However, they cannot be used in the WHERE clause.
- When WHERE, GROUP BY, and HAVING clauses are used together in a SELECT statement:
  1. The WHERE clause is processed first in order.
  2. Subsequently, the rows that are returned after the WHERE clause is executed are grouped based on the GROUP BY clause.
  3. Finally, any conditions, on the Group functions in the HAVING clause, are applied to the grouped rows before the final output is displayed.



The HAVING clause (contd.):

- To find out Average, Maximum, Minimum salary of departments, where average salary is greater than 2000.

```
SELECT dept_code,AVG(staff_sal),MIN(staff_sal),
MAX(staff_sal)
FROM staff_master
GROUP BY dept_code HAVING AVG(staff_sal) >
2000;
```

- To find out average salary of all staff members who belong to department 10 or 20 and their average salary is greater than 10000

```
SELECT design_code,dept_code,avg(staff_sal)
FROM staff_master where dept_code in(10,20)
GROUP BY  design_code,dept_code
HAVING avg(staff_sal) >10000;
```

4.3: Tips and Tricks on using Group Functions, GROUP BY & HAVING clause

Quick Guidelines

- All group functions except COUNT(*) ignores NULL values.
- To substitute a value for NULL values use NVL functions.
- DISTINCT clause makes the function consider only non duplicate values.
- The AVG and SUM are used with numerica data.
- The MIN and MAX functions used with any data type.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

NVL function is covered in the next lesson

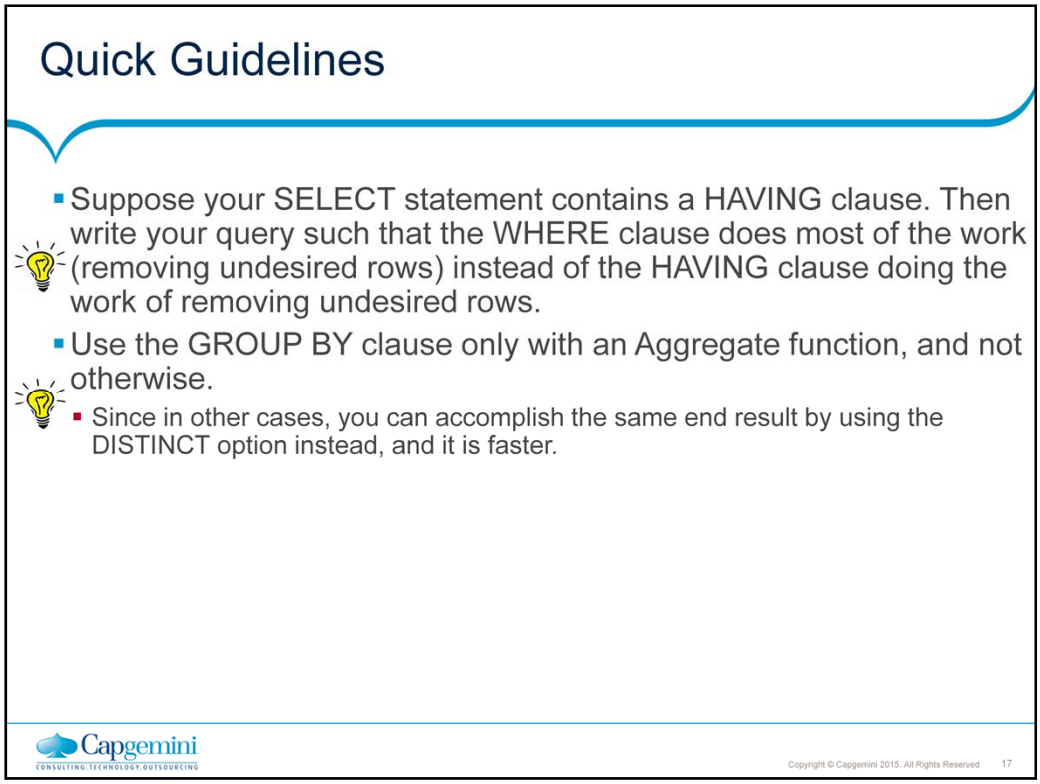
Quick Guidelines

- All individual columns included in the SELECT clause other than group functions must be specified in the GROUP BY clause.
- Any column other than selected column can also be placed in GROUP BY clause.
- By default rows are sorted by ascending order of the column included in the GROUP BY list.
- WHERE clause specifies the rows to be considered for grouping.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

NVL function is covered in the next lesson



Tips and Tricks:

- By appropriately using the WHERE clause, you can eliminate unnecessary rows before they reach the GROUP BY and HAVING clause. Thus saving some unnecessary work, and boosting performance.
  For example: In a SELECT statement with WHERE, GROUP BY, and HAVING clauses, the query executes in the following sequence.
  - First, the WHERE clause is used to select the appropriate rows that need to be grouped.
  - Next, the GROUP BY clause divides the rows into sets of grouped rows, and then aggregates their values.
  - And last, the HAVING clause then eliminates undesired aggregated groups.
    - If the WHERE clause is used to eliminate as many of the undesired rows as possible, then the GROUP BY and the HAVING clauses will have to do less work. Thus boosting the overall performance of the query.

contd.

Tips and Tricks (contd.):

- The GROUP BY clause can be used with or without an Aggregate function. However, if you want optimum performance, do not use the GROUP BY clause without an Aggregate function. This is because you can accomplish the same end result by using the DISTINCT option instead, and it is faster.
  For example: You can write your query two different ways:

```
SELECT OrderID
FROM [Order Details]
WHERE UnitPrice > 10
GROUP BY OrderID
```

or

```
SELECT DISTINCT OrderID
FROM [Order Details]
WHERE UnitPrice > 10
```

Both of the above queries produce the same results, but the second one will use less resources and perform faster.

Summary

- In this lesson, you have learnt about:
  - Aggregate (Group functions)
    - GROUP BY clause
    - HAVING clause

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

Review – Questions

- Question 1: Identify the various group functions from the list given below:
  - Option 1: maximum
  - Option 2: sum
  - Option 3: count
  - Option 4: minimum

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

Review – Questions

- Question 2: The AVG function ignores NULL values in the column.
  - True / False
- Question 3: Count(*) returns the number of rows in the table, including duplicates and those with NULLs.
  - True / False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

DBMS/SQL

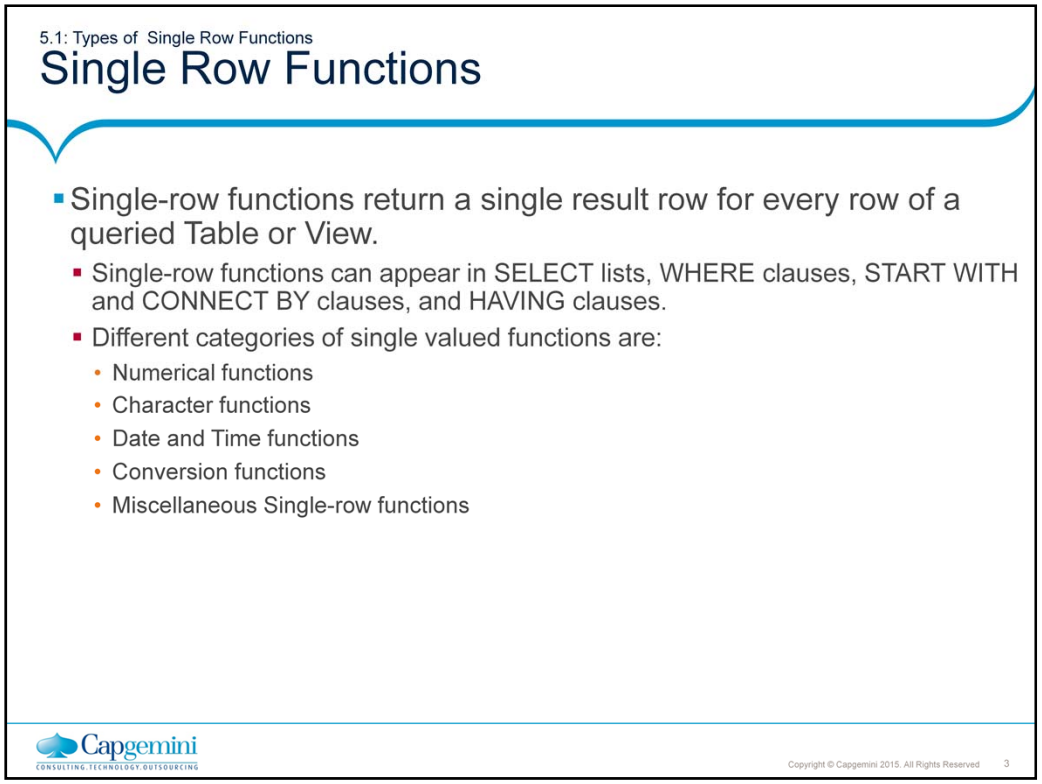
Lesson 05 SQL (Single-row) Functions

Lesson Objectives

- To understand the following topics:
  - SQL (single-row) functions
    - Number functions
    - Character functions
    - Date functions
    - Conversion functions
    - Miscellaneous Single-row functions

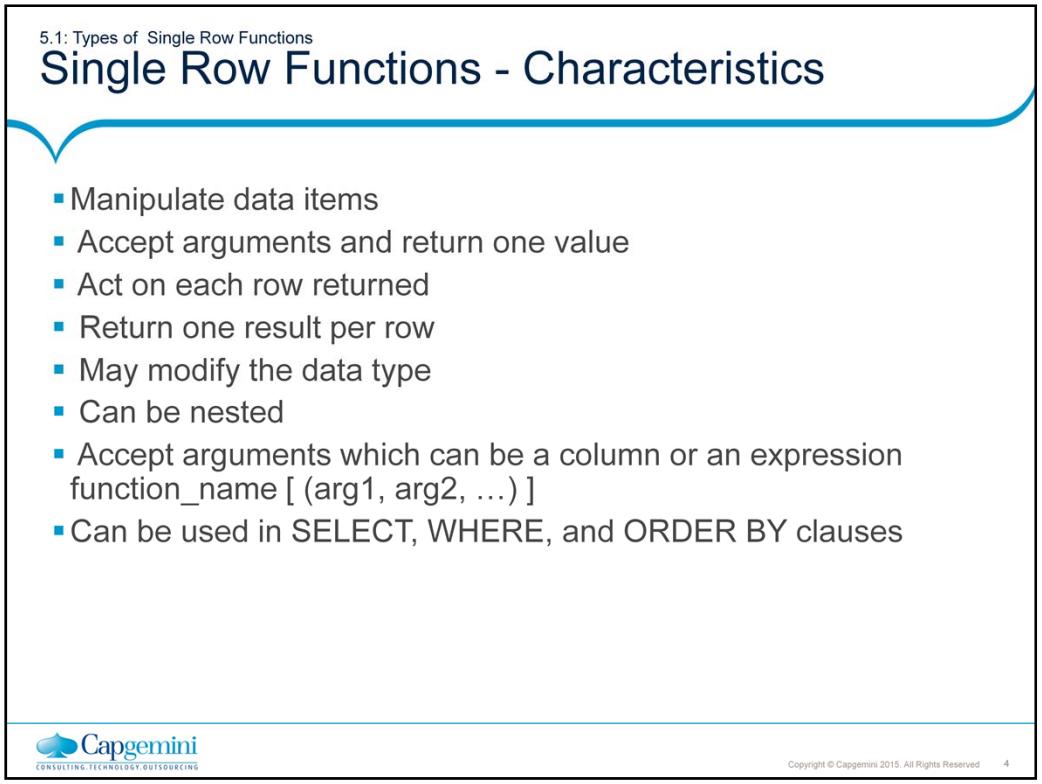
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

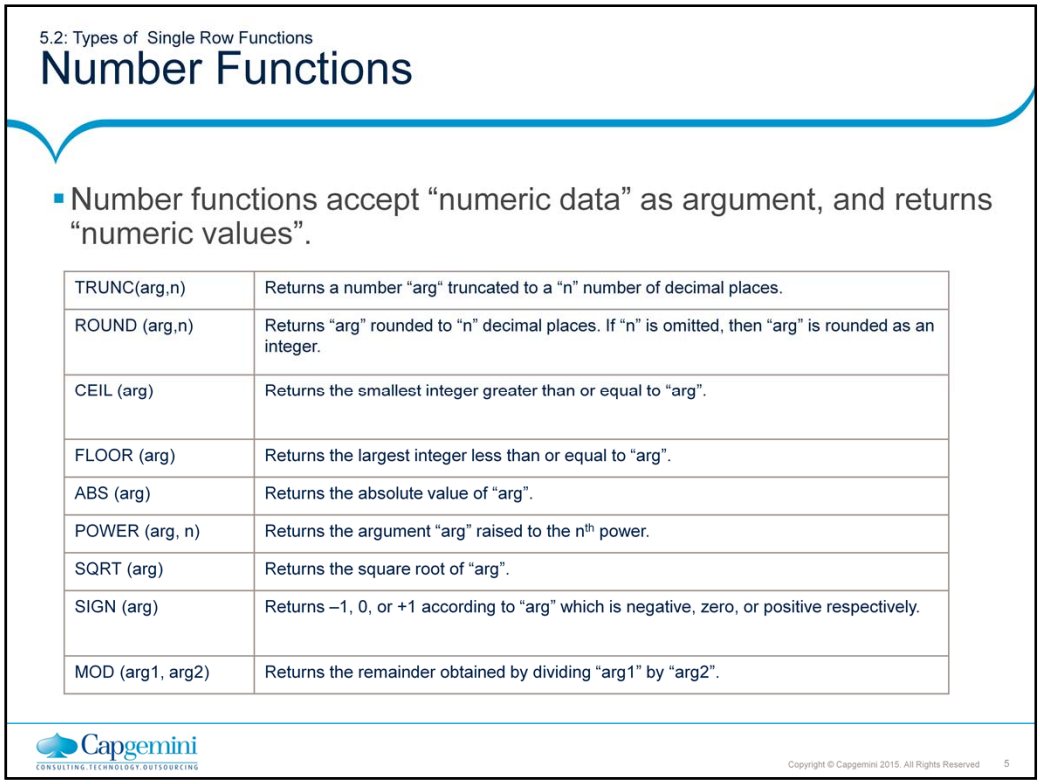


SQL Functions:

- So far, we have seen “Aggregate functions”, which operate against a “collection of values”, however return a “single value”.
- Now we shall see “scalar functions” which operate against a “single value”, and return a “single value” based on the input value.
- The functions can be broadly classified into:
  - Numerical functions- Accept numeric input and return numeric values
  - For example: ABS, CEIL, TRUNC, ROUND, POWER, etc.
  - Character functions - Accept character input and can return both characters and number values
  - For example: CONCAT, LPAD, RPAD, TRIM, SUBSTR, etc.
  - Date and Time functions- Operate on values of the DATE data type
  - For example: SYSDATE, ADD_MONTHS, LAST_DAY, etc.
  - Conversion functions- Convert a value from one data type to another
  - For example: CAST, ASCIISTR, ROWIDTOCHAR, etc.
  - Miscellaneous Single-row functions
  - For example: BFILENAME, DECODE, NVL, NULLIF, USERENV, etc.



Let us now have a look at each of these function categories



Numeric Functions:

TRUNC(n,m)

The trunc function returns a number truncated to a certain number of decimal places. The syntax for the trunc function is:

```
trunc( number, [ decimal_places ] )
```

where:
number = the number to truncate.
decimal_places = the number of decimal places to truncate to. This value must be an integer. If this parameter is omitted, the trunc function will truncate the number to 0 decimal places.

For example:

trunc(125.815) would return 125
trunc(125.815, 1) would return 125.8
trunc(125.815, -1) would return 120
trunc(125.815, -2) would return 100

ROUND(n,m)

The round function returns a number rounded to a certain number of decimal places. The syntax for the round function is:

```
round( number, [ decimal_places ] )
```

Numeric Functions (contd.):
where:
number = the number to round.
decimal_places = the number of decimal places rounded to. This value must be an integer. If this parameter is omitted, the round function will round the number to 0 decimal places.
For example:
round(125.315) would return 125
round(125.315, 0) would return 125
round(125.315, 1) would return 125.3
round(125.315, 2) would return 125.32
round(-125.315, 2) would return -125.32
CEIL(n)
It returns smallest integer greater than or equal to n.

```
SELECT CEIL(15.7) "Ceiling" FROM DUAL;
```

Ceiling
16

FLOOR(n)
•It returns largest integer equal to or less than n.

```
SELECT FLOOR(15.7) "Floor" FROM DUAL;
```

Floor
15

ABS(n)
•It returns the absolute value of n.

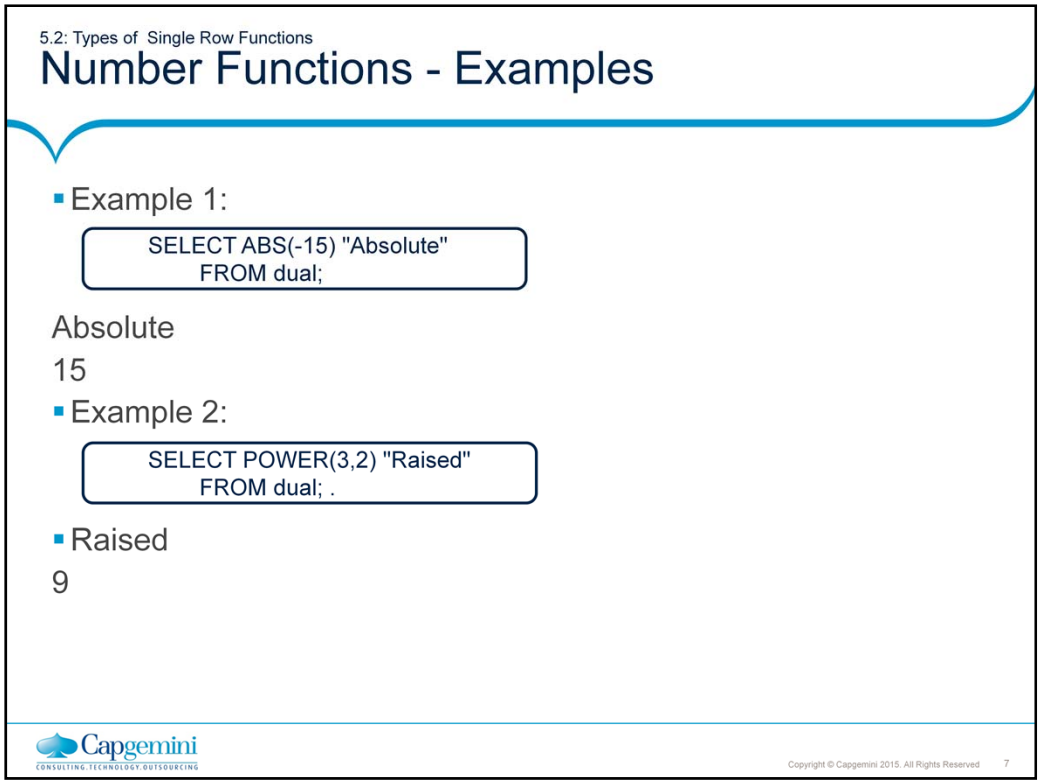
```
SELECT ABS(20) "Absolute" FROM DUAL;
```

Absolute
20

POWER function
•It returns m raised to n^{th} power .It is of the form: Power(m,n)

```
SELECT POWER (3,3) “Raised” FROM DUAL;
```

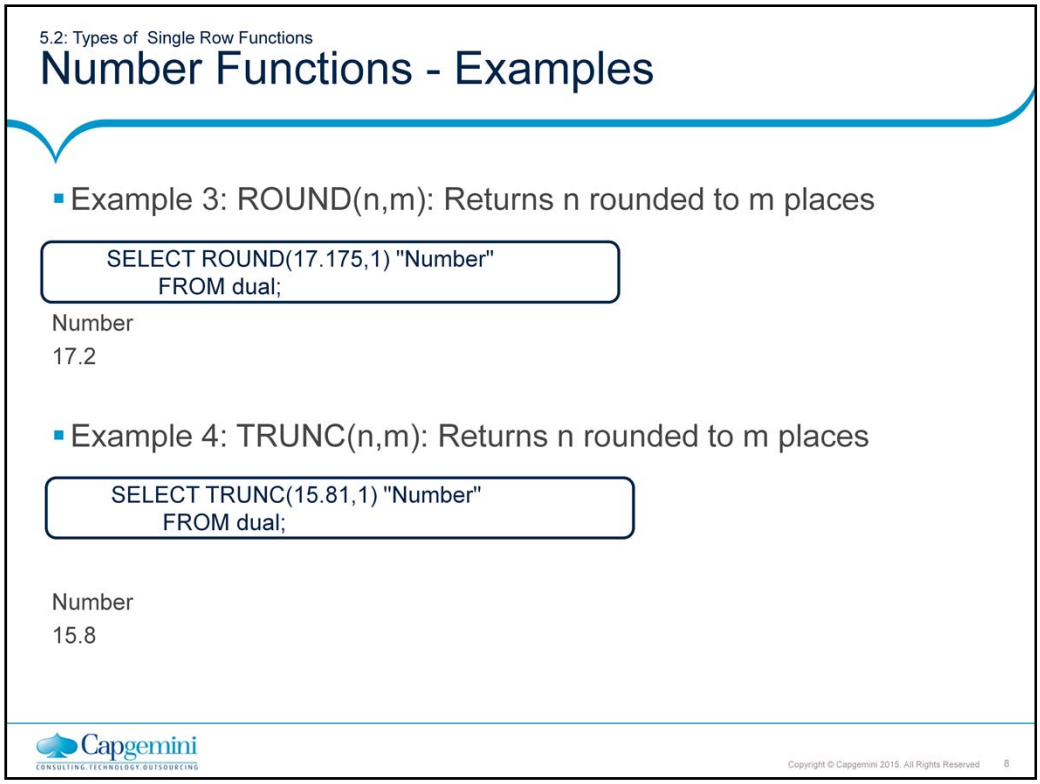
Raised
27



Examples of Number Functions:

- DUAL table, which is shown in the slide, is a table owned by SYS.
  - SYS owns the “data dictionary”, and DUAL is part of the data dictionary.
  - DUAL is a small work-table, which consists of only one row and one column, and contains the value “x” in that column. Besides arithmetic calculations, it also supports “date retrieval” and it’s “formatting”.
  - Often a simple calculation needs to be done. A SELECT must have a table name in it’s FROM clause, else it fails.
  - To facilitate such calculations via a SELECT, the DUAL dummy table is provided.
  - The structure of the DUAL table can be viewed by using the SQL statement:

```
DESC DUAL;
```



Examples of Number (numeric) Functions (contd.):

- **Round(n,m)**:

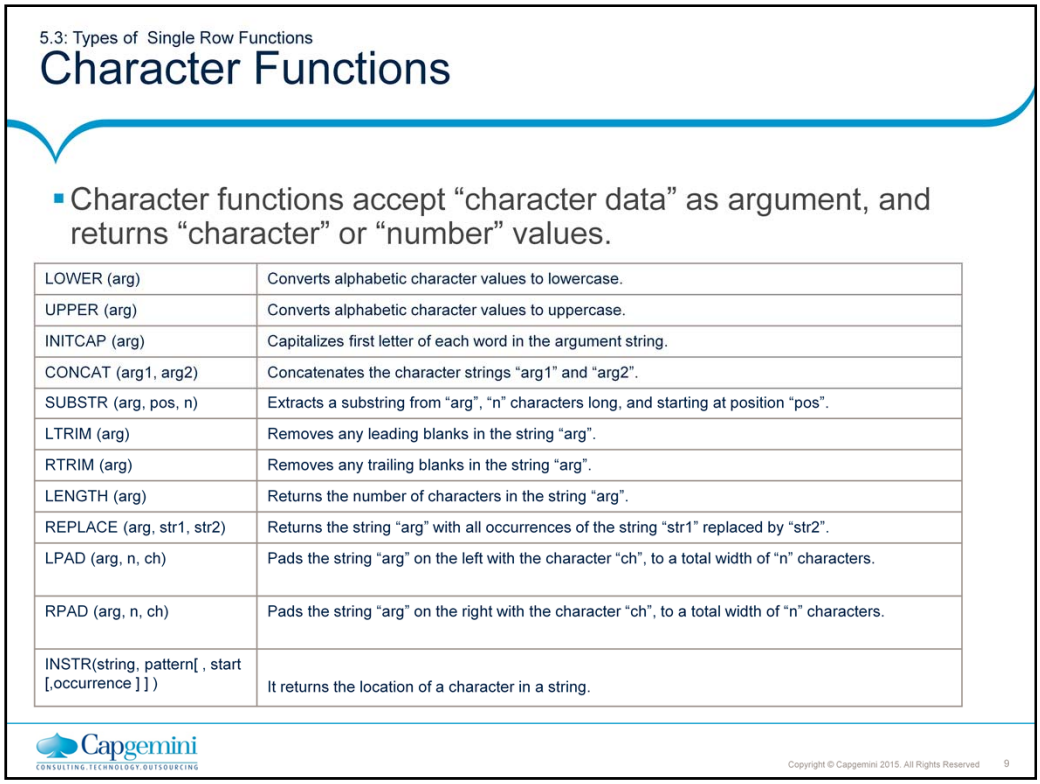
```
SELECT ROUND(17.175,-1) "Number"
FROM dual;
```

O/P : 20

- **TRUNC(n,m)**:

```
SELECT TRUNC(15.81,-1) "Number“
FROM dual;
```

O/P : 10



5.3: Types of Single Row Functions

Character Functions

- Character functions accept "character data" as argument, and returns "character" or "number" values.

LOWER (arg)	Converts alphabetic character values to lowercase.
UPPER (arg)	Converts alphabetic character values to uppercase.
INITCAP (arg)	Capitalizes first letter of each word in the argument string.
CONCAT (arg1, arg2)	Concatenates the character strings "arg1" and "arg2".
SUBSTR (arg, pos, n)	Extracts a substring from "arg", "n" characters long, and starting at position "pos".
LTRIM (arg)	Removes any leading blanks in the string "arg".
RTRIM (arg)	Removes any trailing blanks in the string "arg".
LENGTH (arg)	Returns the number of characters in the string "arg".
REPLACE (arg, str1, str2)	Returns the string "arg" with all occurrences of the string "str1" replaced by "str2".
LPAD (arg, n, ch)	Pads the string "arg" on the left with the character "ch", to a total width of "n" characters.
RPAD (arg, n, ch)	Pads the string "arg" on the right with the character "ch", to a total width of "n" characters.
INSTR(string, pattern[, start [,occurrence]])	It returns the location of a character in a string.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

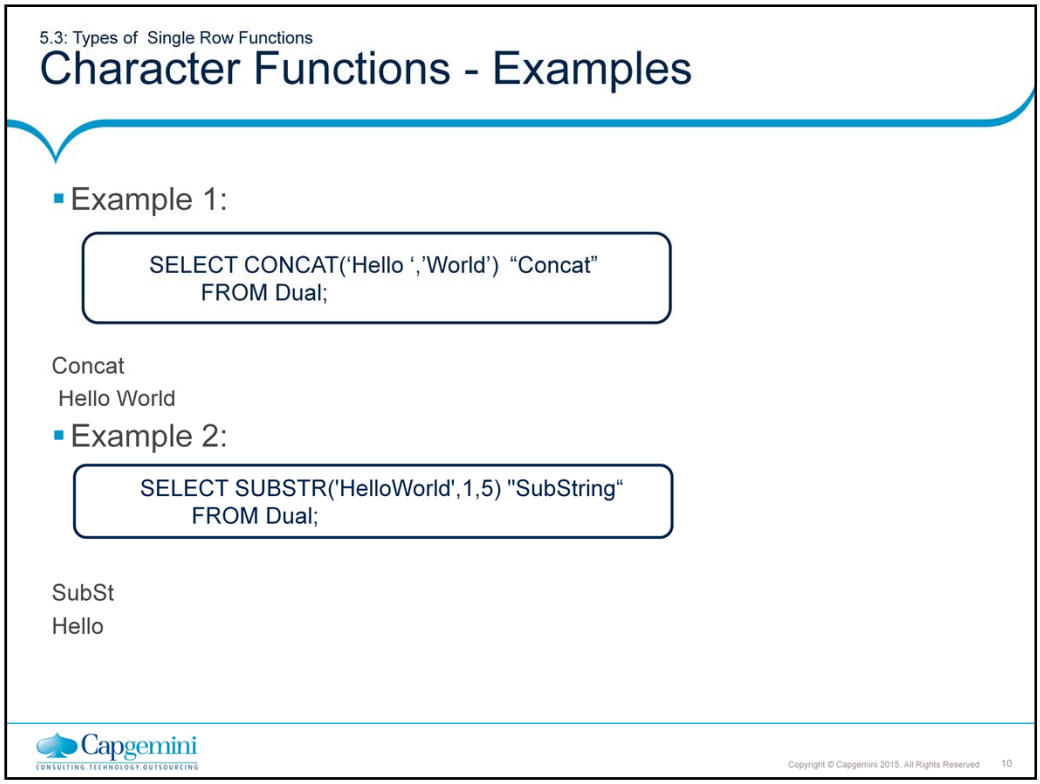
Character Functions:

Example1:

```
SELECT Upper('Hello'), Lower('WORLD')
FROM Dual;
```

Note: Functions can be nested to any depth. Evaluation starts with the "inner most functions", and proceeds outwards.

For example: LENGTH(LTRIM(RTRIM(name)))



Examples of Character Functions:

- **REPLACE function** returns char with every occurrence of search_string replaced with replacement_string.
  - If replacement_string is omitted or NULL, then all occurrences of search_string are removed.
  - If search_string is NULL, then char is returned.

```
SELECT REPLACE('JACK and JUE','J','BL') "Changes"
FROM DUAL;
```

Changes
BLACK and BLUE

- **CONCAT function** returns "char1" concatenated with "char2". Both "char1" and "char2" can be any of the datatypes CHAR, VARCHAR2, NCHAR, NVARCHAR2, CLOB, or NCLOB. The string returned is in the same character set as char1. It's datatype depends on the datatypes of the arguments.
  - In concatenations of two different datatypes, the Oracle Database returns the datatype that results in a "lossless conversion". Therefore:
    - If one of the arguments is a LOB, then the returned value is a LOB.
    - If one of the arguments is a national datatype, then the returned value is a national datatype.

String Functions:

- The CONCAT function is equivalent to the “concatenation operator (||)”. The function is useful when there are spaces in the values to be concatenated. The “concatenation operator” does not permit spaces.

UPPER(string)

- This function converts all characters in string to uppercase.

```
SELECT UPPER(staff_name),staff_code
    FROM staff_master;
```

LOWER(string)

- This function converts all characters in string to lowercase.

```
SELECT LOWER(student_name)
    FROM student_master;
```

INITCAP(string):

- This function converts the first character of each word in string to uppercase and the rest of the characters to lowercase.

```
SELECT INITCAP(staff_name)
    FROM staff_master;
```

LPAD(string1,n,string2)

- This function adds string2 before string1 as many times as required to make the string1 length equal to “n” chars.
- To right align the names of staff_members:

```
SELECT LPad(staff_name,30)
    FROM staff_master;
```

LTRIM(string,CHAR set)

- This function removes chars from beginning of string as long as the character matches one of the chars in the CHAR set.

```
SELECT student_name,LTRIM(student_name,'MALICE')
    FROM student_master;
```

RPAD(string1,n,string2)

- This function is similar to LPAD. It adds chars to the right end.

RTRIM(string,CHAR set)

- This function is similar to LTRIM. It removes chars from the right end.

SUBSTR(CHAR,m,n)

- This function returns the string from the m^{th} character of the string to the n^{th} character

String Functions (contd.):

- **LENGTH function**
  It returns the length of char in characters
  It is of the form:
  Length (string)

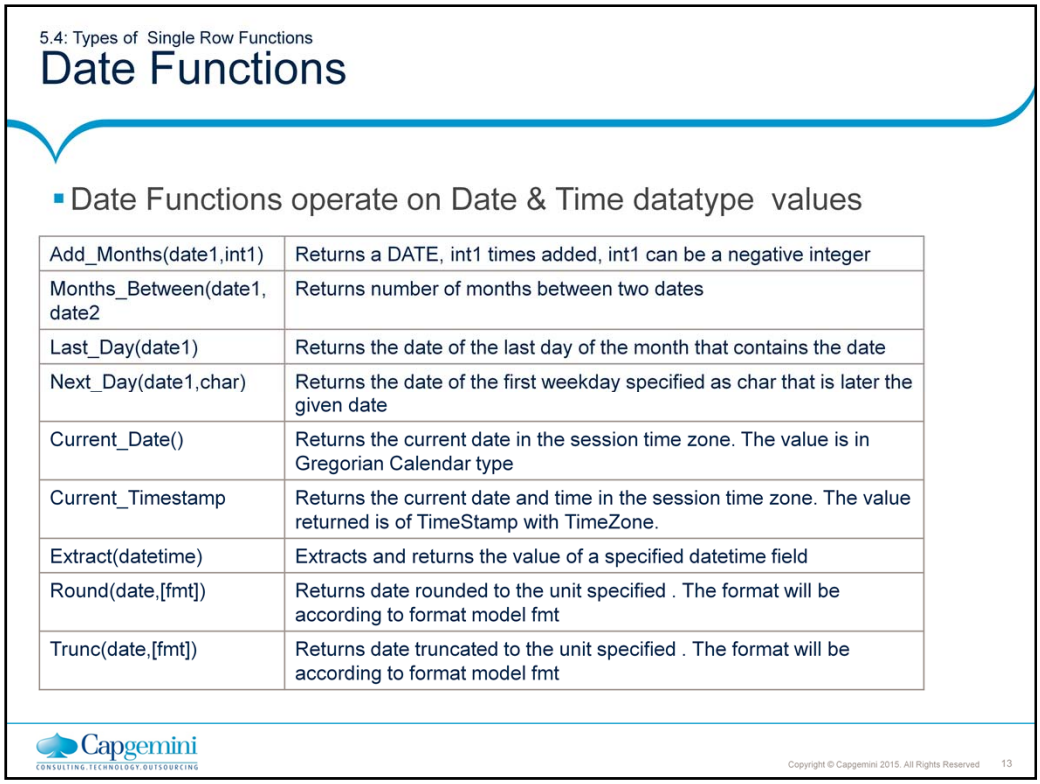
```
Select length ('candide') " length in characters"
    FROM dual;
```

INSTR(string, pattern[, start [,occurrence]])

- It returns the location of a character IN a STRING.

```
SELECT INSTR ('CORPORATEFLOOR','R',3,2) "Instring"
    FROM DUAL
```

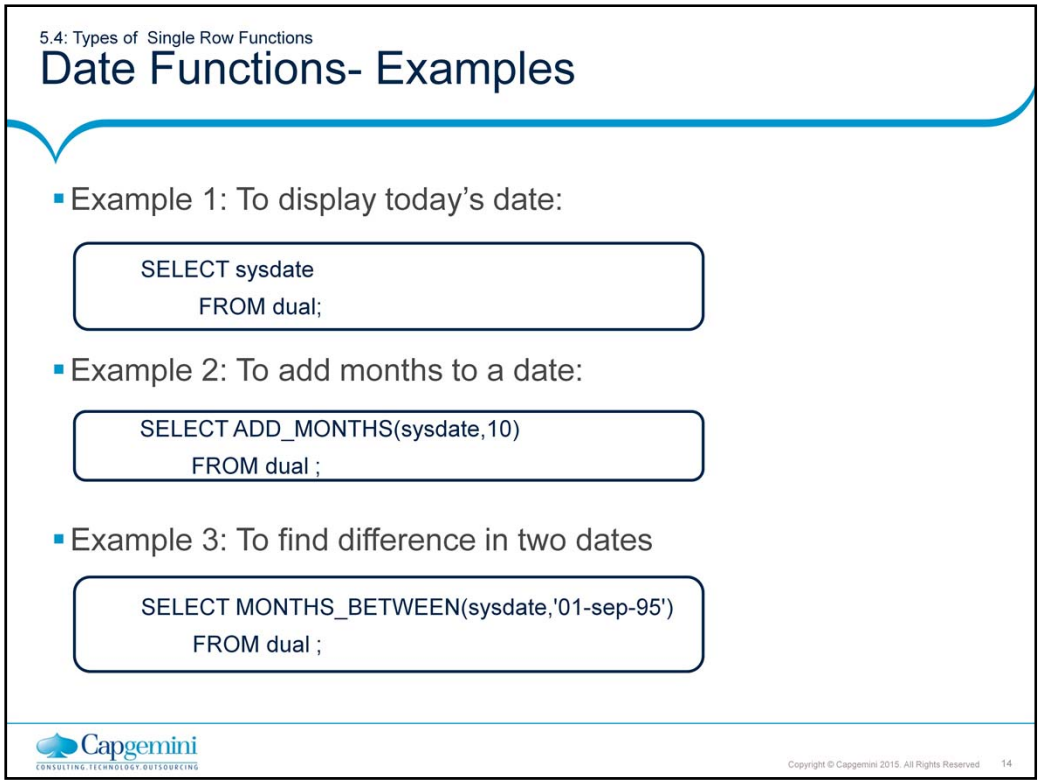
Instring
6



DATE Functions

- Datetime functions operate on date (DATE), timestamp, and interval values.
- Some of the datetime functions were designed for the Oracle DATE datatype (ADD_MONTHS, CURRENT_DATE, LAST_DAY, and NEXT_DAY).
  - ➢ If you provide a timestamp value as their argument, Oracle Database internally converts the input type to a DATE value and returns a DATE value.
  - ➢ The exceptions are:
    - the MONTHS_BETWEEN function, which returns a number, and
    - the ROUND and TRUNC functions, which do not accept timestamp or interval values at all.

•The remaining datetime functions are designed to accept any of the three types of data (date, timestamp, and interval), and to return a value of one of these types.



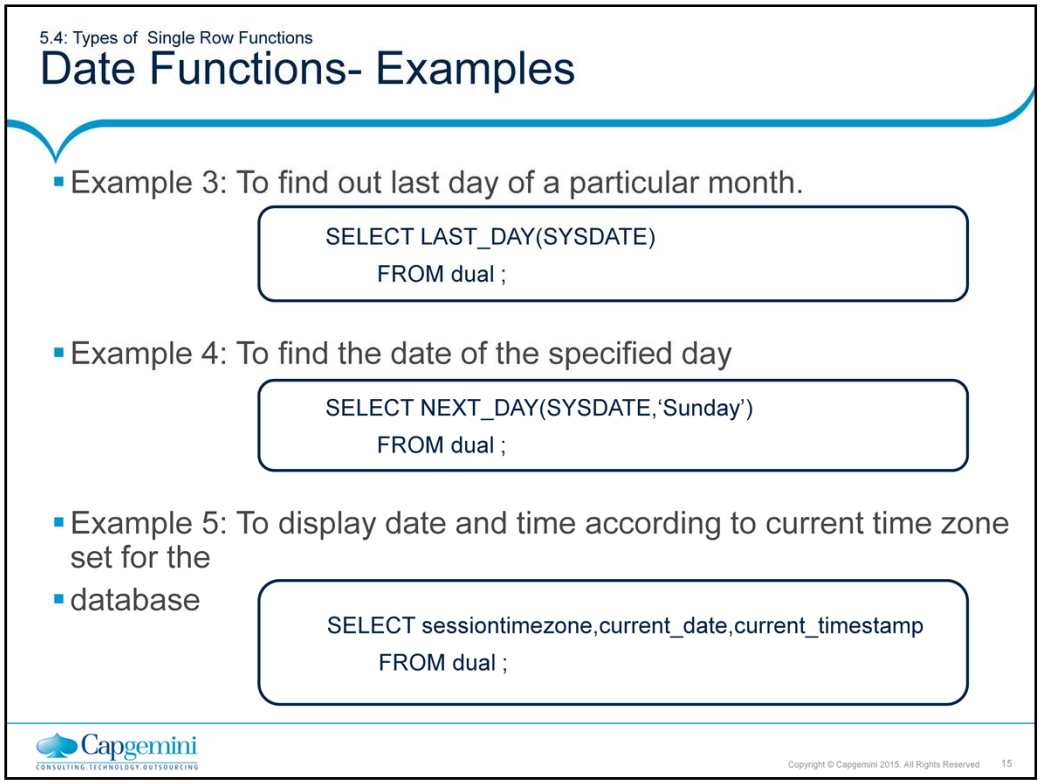
Date Functions:

ADD_MONTHS(DATE1,int1) returns the date as addition of "date" and "integer" months. The first argument can be a datetime value or any value that can be implicitly converted to DATE. The second argument can be an integer or any value that can be implicitly converted to an integer. The return type is always DATE.

```
SELECT book_code,ADD_MONTHS(book_issue_date,1)
FROM book_transaction;
```

MONTHS_BETWEEN (DATE1,DATE2) This function returns number of months between the two DATEs. The result is positive if date1 is later than date2 and result is negative if date2 is later than date2.

```
SELECT staff_code,
MONTHS_BETWEEN(TRUNC(sysdate),hiredate)
FROM staff_master;
```



Date Functions (contd.):

LASTDAY(date1) returns the date of the last day of the month that contains the date. The return type is always DATE.

- To display the date of the last day in the month the books were issued

```
SELECT book_issue_date, LAST_DAY(book_issue_date)
    FROM book_transaction;
```

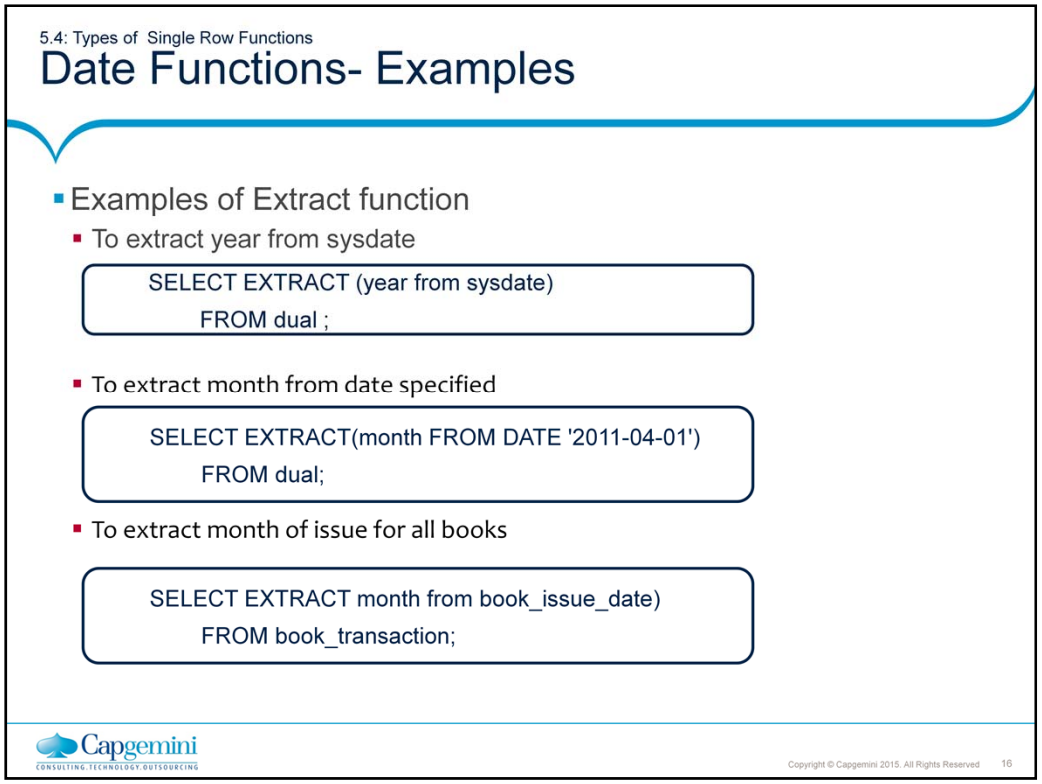
NEXT_DAY(date1, char) Returns the date of the first weekday specified as char that is later the given date

- To display the date on coming Friday in the week that books were issued

```
SELECT book_issue_date,
NEXT_DAY(book_issue_date,'Friday')
    FROM book_transaction
```

CURRENT_DATE & CURRENT_TIMESTAMP return current date and time respectively base on the timezone set for the database.

The query given on the slide above shows the date and time based on the offset time base on GMT which is according to the timezone set for the database



Date Functions (contd.):

EXTRACT(datetime) extracts the value of a specified datetime field. This function is useful for manipulating datetime field values. For example you can extract only year, month or day from a given date value.

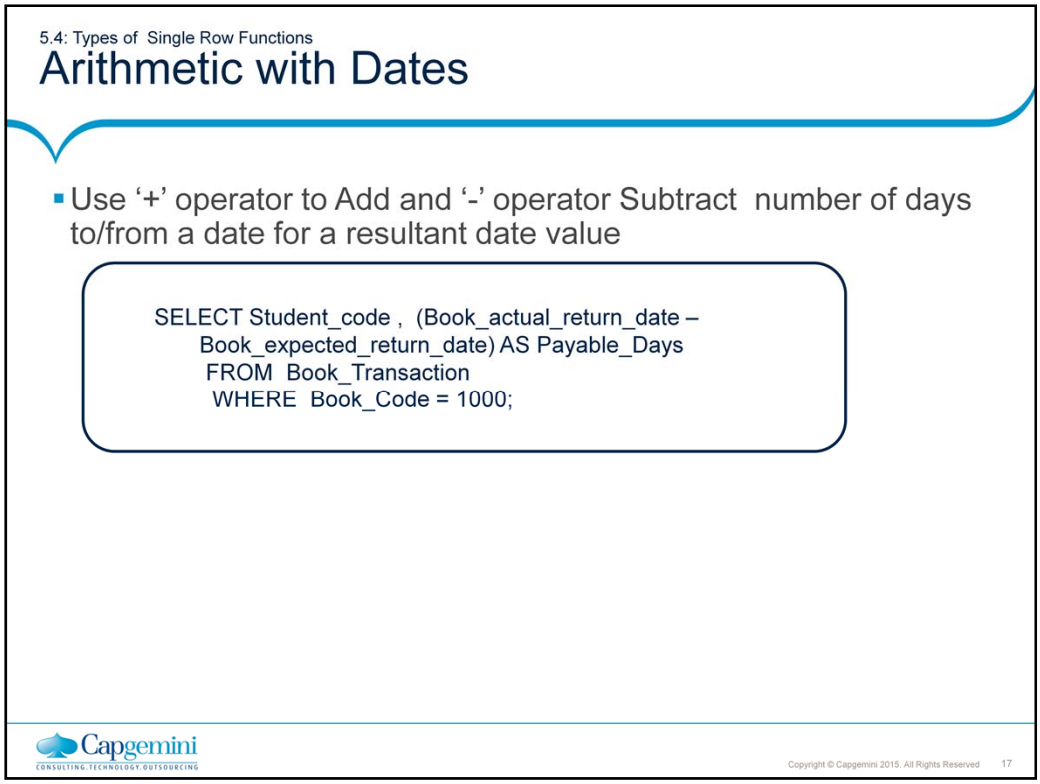
- To display year of birth for all students

```
SELECT EXTRACT(year from student_dob)
FROM student_master
```

TRUNC (DATE1)

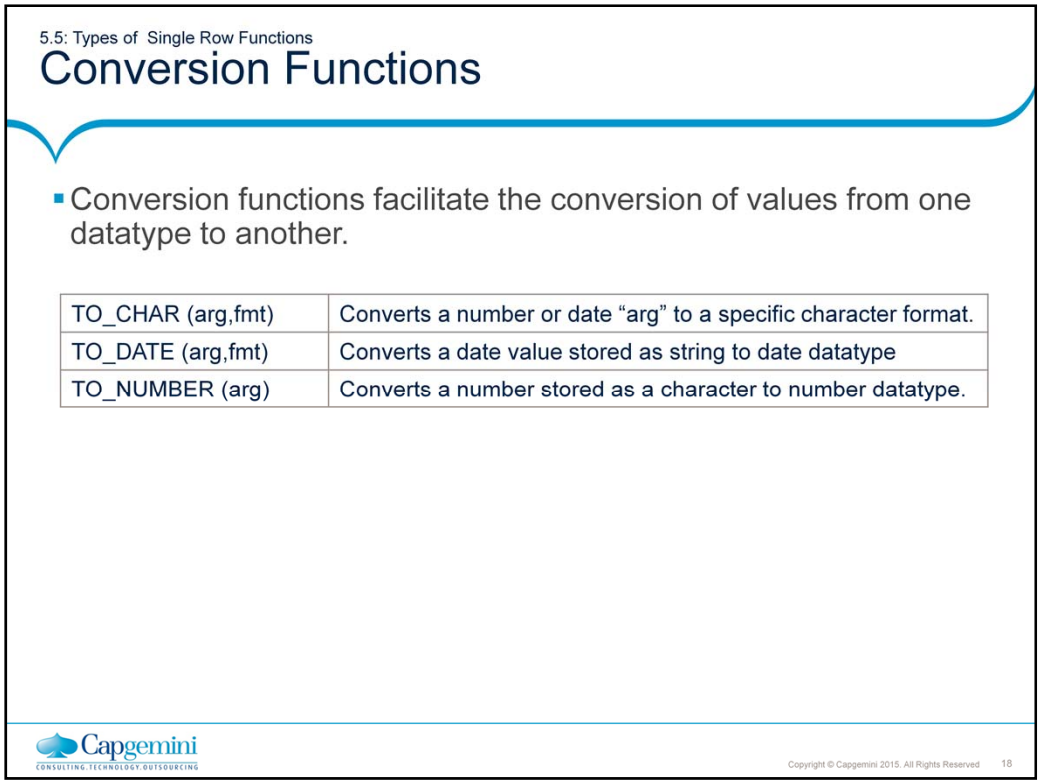
- This function truncates the time part from the DATE. This is required when we do DATE calculations.

```
SELECT staff_name, TRUNC(sysdate) -
TRUNC(HIREDATE)  FROM staff_master;
```



- Given below are formats of Date Functions

Format	Meaning
YYYY	Four digit year
YY	Last two digits of the year
MM	Month (01-12 where JAN = 01 …)
MONTH	Name of the month stored as a length of nine characters.
MON	Name of the month in three letter format.
DD	Day of the month (01-31).
D	Day of the week.
DAY	Name of the day stored as a length of nine characters.
DY	Name of the day in three letter format.
FM	This prefix can be added to suppress blank padding.
HH	Hour of the day.
HH24	Hour in the 24 hour format.
MI	Minutes of the hour.
SS	Seconds of the minute.
TH	The suffix used with the day.

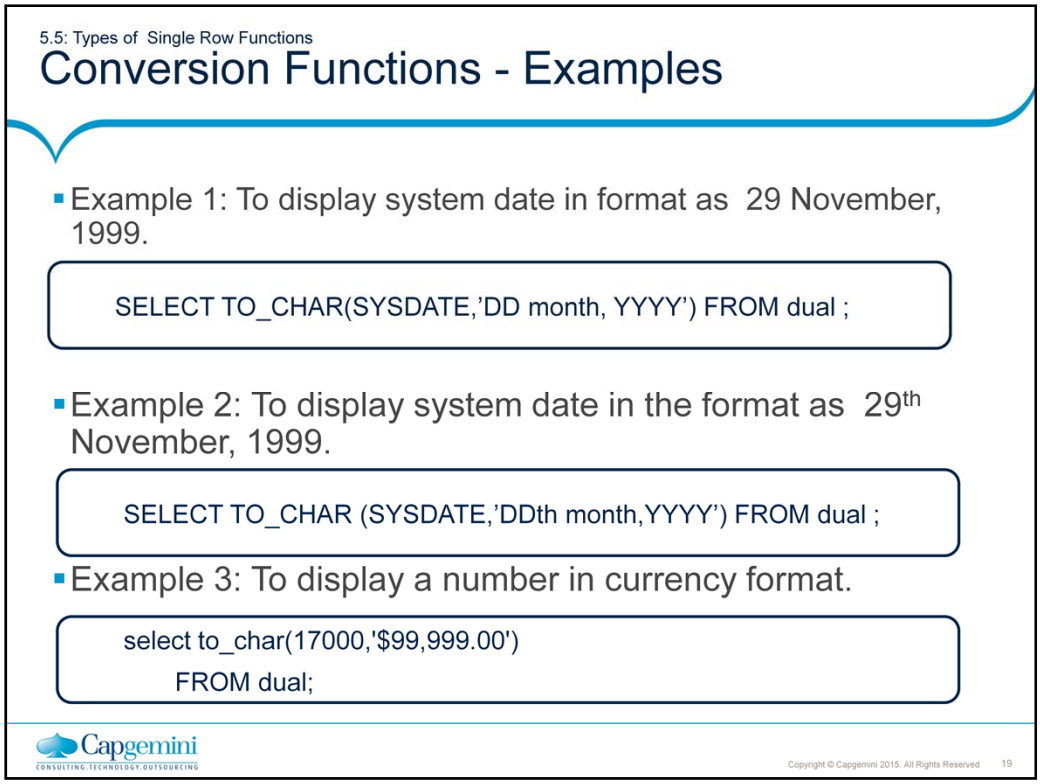


Conversion Functions

Conversion functions convert values of one datatype to another. Usually the conversion function accepts two arguments wherein first is the input type and second is the output type.

Oracle takes care of implicit datatype conversion. Explicit datatype conversions are done using the conversion functions.

Although Oracle does provide implicit type conversion to ensure reliability of SQL statements you should use conversion functions



TO_CHAR(DATE1,format)

- This function converts the DATE given to the format specified.

```
SELECT TO_CHAR(TRUNC(sysdate), 'ddth fmMonth yy')
'DATE'
FROM dual;
```

DATE
01st January 95

```
SELECT TO_CHAR(TRUNC(sysdate),'fmMonth') "DATE"
FROM dual;
```

DATE
July

- Example: To display the quarter which has the specified date.

```
SELECT to_char ( sysdate , 'Q')
FROM dual ;
```

5.5: Types of Single Row Functions

Conversion Functions - Examples

- Example 3: To display employees whose hire date is September 08, 1981.

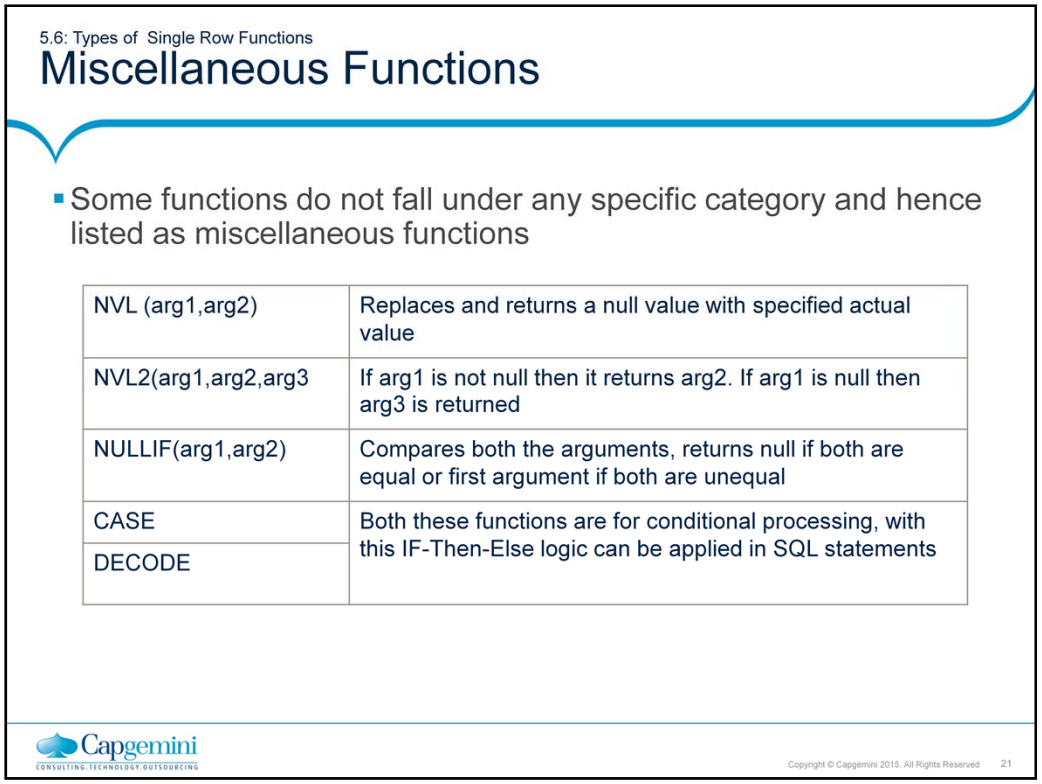
```
SELECT staff_code, hiredate
    FROM staff_master
    WHERE
    hiredate = TO_DATE ('September 08,1981','Month DD, YYYY');
```

- Example 4: To display the value in timestamp format

```
SELECT TO_TIMESTAMP(sysdate,'DD-MM-YY')
    from dual;
```

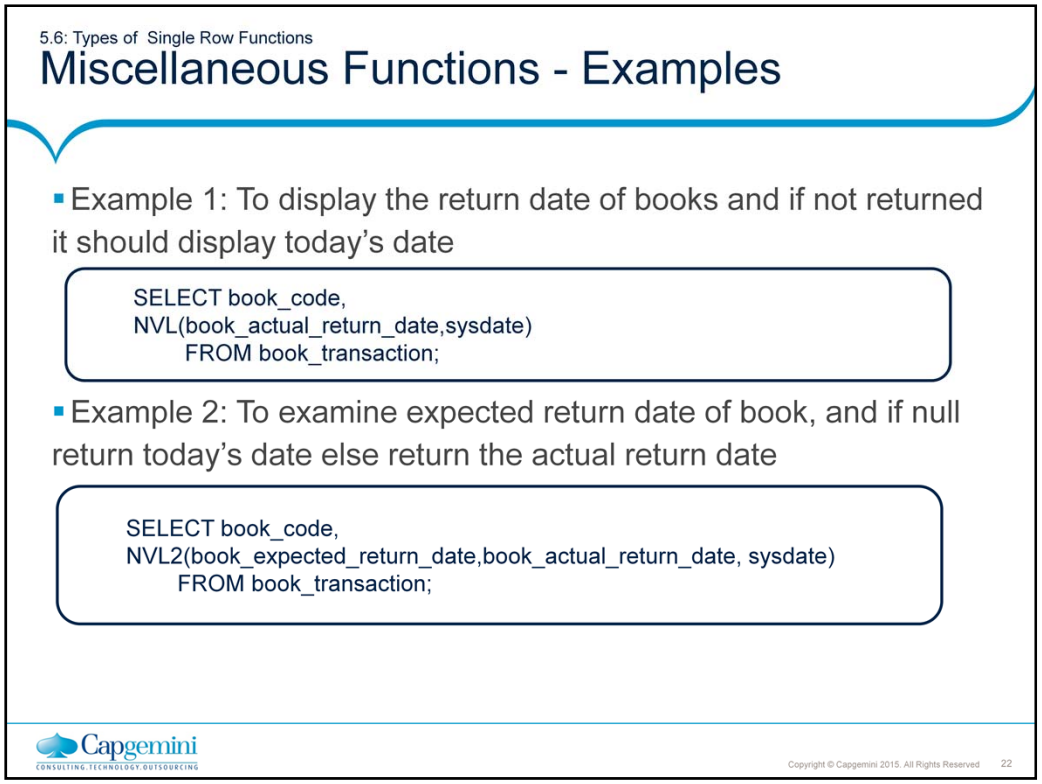
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20



Miscellaneous Functions

These functions are sometimes also called as General Functions. These functions work with any datatype values



Miscellaneous Single-row Functions:

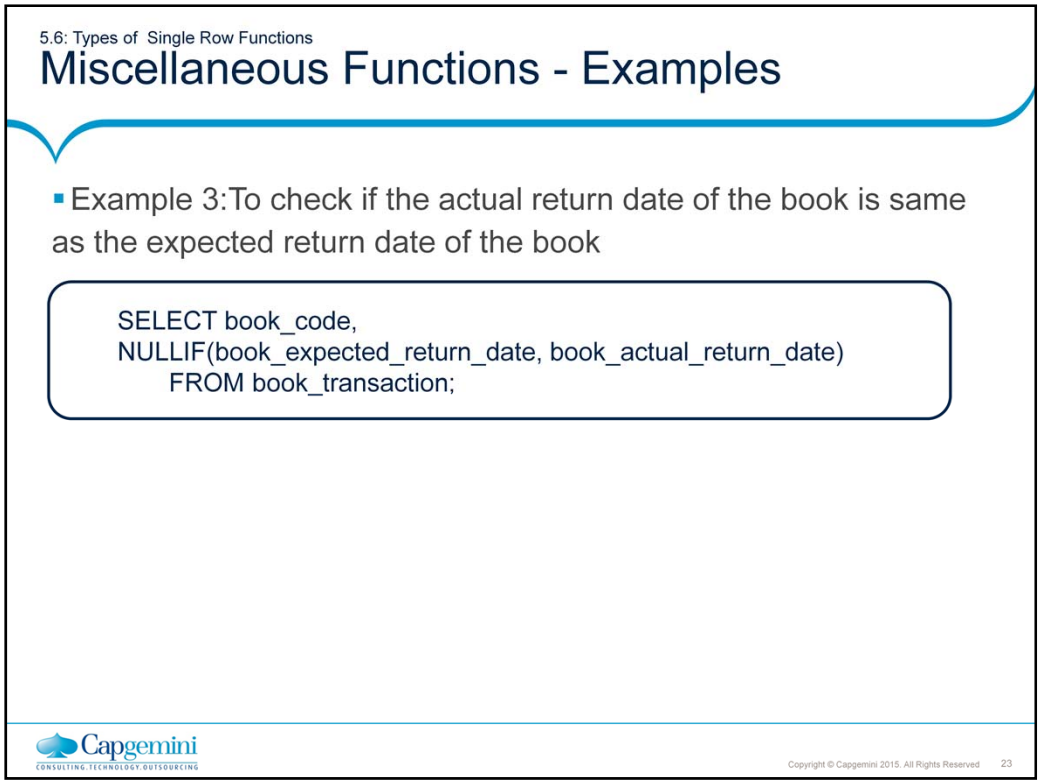
NVL ()

- Many times there are records holding NULL values in a table. When an output of such a table is displayed, it is difficult to understand the reason of NULL or BLANK values shown in the output.
- The only way to overcome this problem is to replace NULL values with some other meaningful value while computing the records. This can be done using the NVL function.

NVL2()

- The NVL2 function can be thought of as an extension to NVL function. But this function examines the first value. If the value is not null then returns the second value and if null then returns the third value.

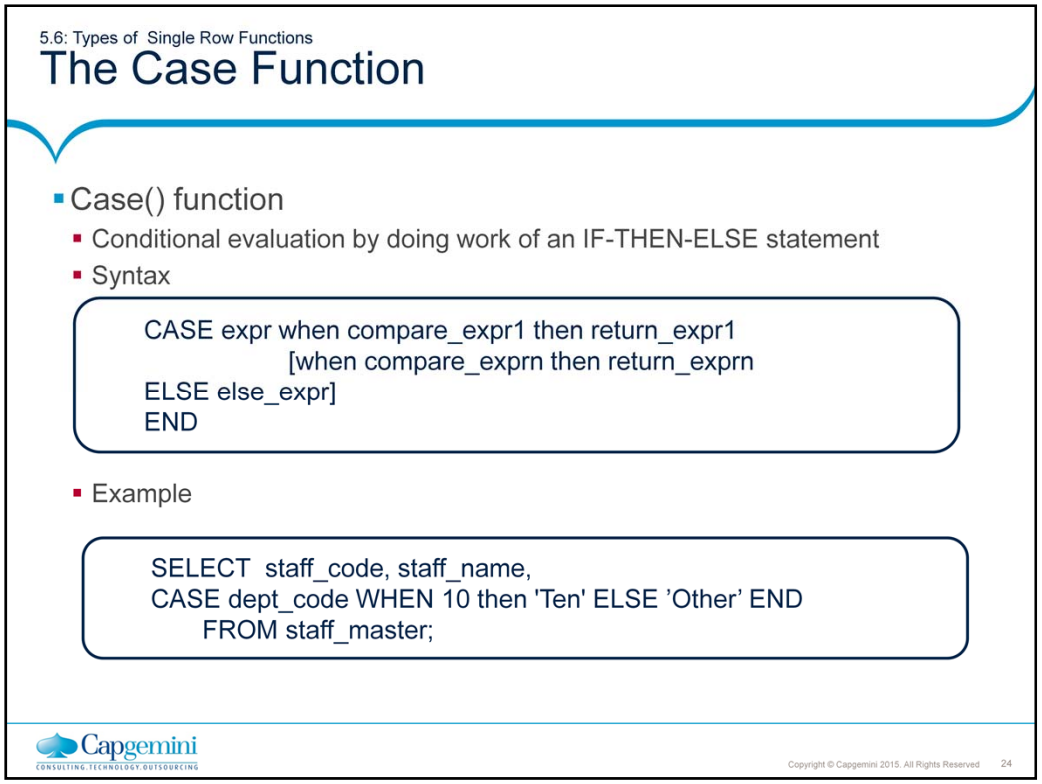
Examples for both functions are shown on the slide



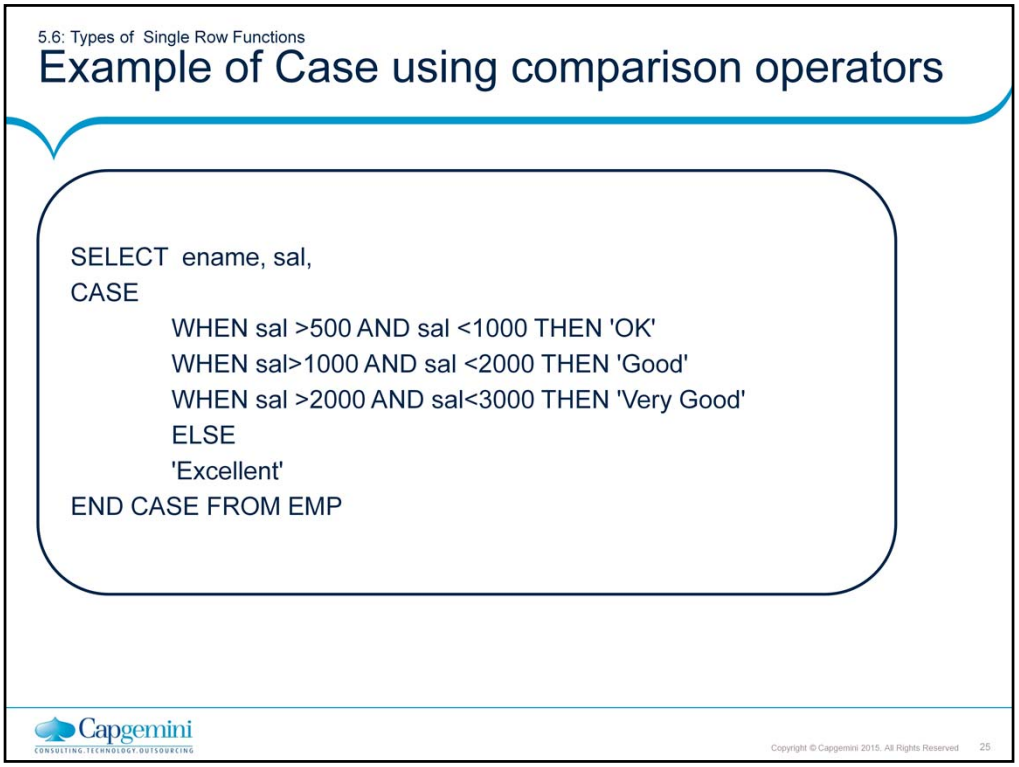
NullIF ()

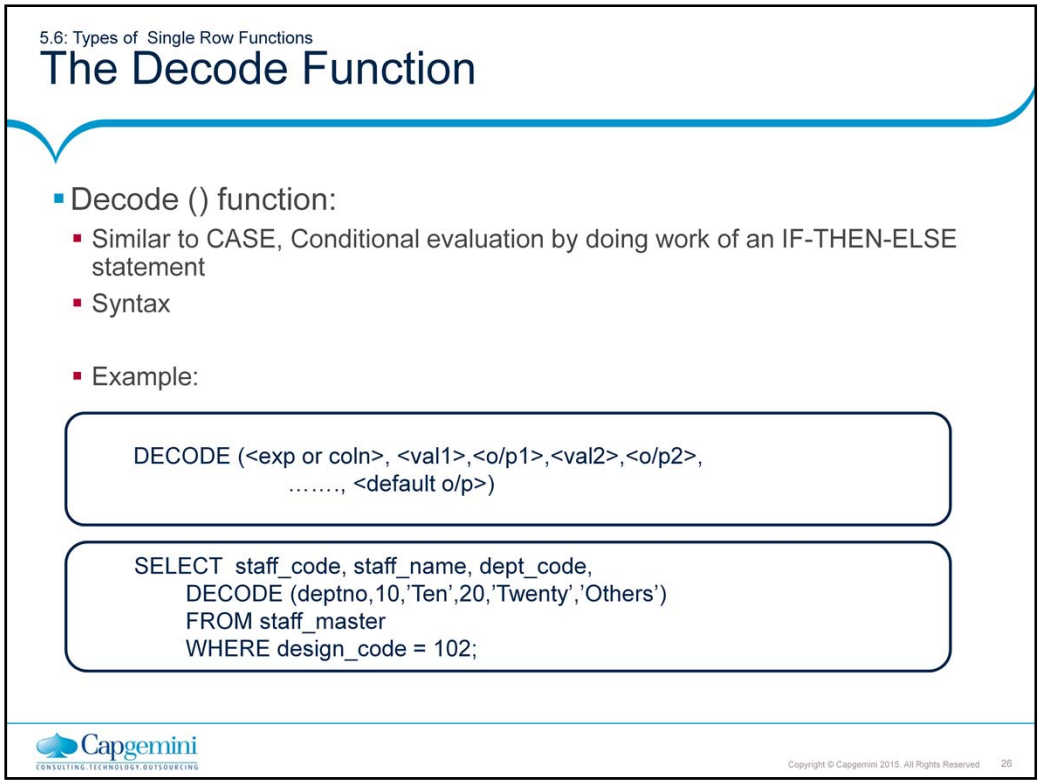
- This function compares the arguments provided. If they are equal, the function returns null and if not then it returns the first argument.
- You cannot specify the literal NULL as the first argument.

Example shown on the slide



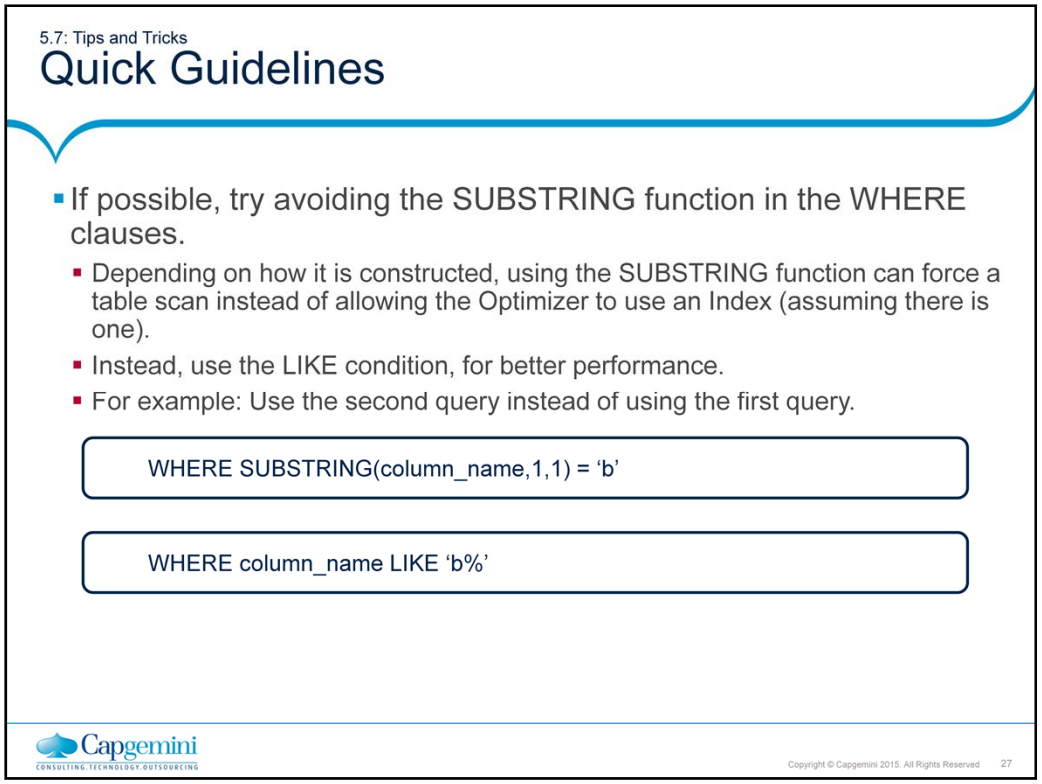
CASE()
In a simple case expression, Oracle searches for the first WHEN…THEN pair for which expr is equal to comparison expr and returns return_expr.
The expressions used should be of same datatype.
CASE is capable of more logical comparisons like <> etc..
Also CASE can work with predicates and subqueries


5.6: Types of Single Row Functions
Example of Case using comparison operators
SELECT ename, sal,
CASE
WHEN sal >500 AND sal <1000 THEN 'OK'
WHEN sal>1000 AND sal <2000 THEN 'Good'
WHEN sal >2000 AND sal<3000 THEN 'Very Good'
ELSE
'Excellent'
END CASE FROM EMP
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
25



DECODE () function

- This function decodes the expression after comparing it to each search value. If the expression is the same as search, result is returned. If the default value is omitted, a null value is returned where a search value does not match any of the result values.
- As compared to CASE, DECODE can do an equality check only. The expressions in DECODE can work only on scalar values.
- DECODE can work as a function inside SQL only but CASE can be more efficient substitute in PL/SQL blocks.



Tips and Tricks:

- If possible, try avoiding the SUBSTRING function in your WHERE clauses.
- Depending on how it is constructed, using the SUBSTRING function can force a table scan instead of allowing the Optimizer to use an Index (assuming there is one).
  - ➢ If the substring you are searching for does not include the first character of the column you are searching for, then a table scan is performed.
- ➢ In case of conversion functions, If value to be converted is not in right format, then Oracle will throw an error

Summary

- In this lesson, you have learnt:
  - SQL (single-row) functions
  - Character functions
  - Number functions
  - Date functions
  - Conversion functions
  - Miscellaneous functions

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28

Review – Questions

- Question 1: Single row functions can be broadly classified as _____.
  - Option 1: character functions
  - Option 2: numeric functions
  - Option 3: Date functions
  - Option 4: all the above

- Question 2: The function which returns the value after capitalizing the first character is ___.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 29

Review – Questions

- Question 3: The output of the following function will be ___.
  - select to_char(17000,'$99,999.00') 'Amount' from dual;
- Question 4: The decode function can convert NULL values to required type.
  - True / False

Review – Questions

- Question 5: The function which returns the last date of the month is ___.
  - Option 1: LAST_DATE
  - Option 2: LAST_DAY
  - Option 3: MONTH_LAST_DATE
  - Option 4: MONTH_LAST_DAY

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31

DBMS/SQL

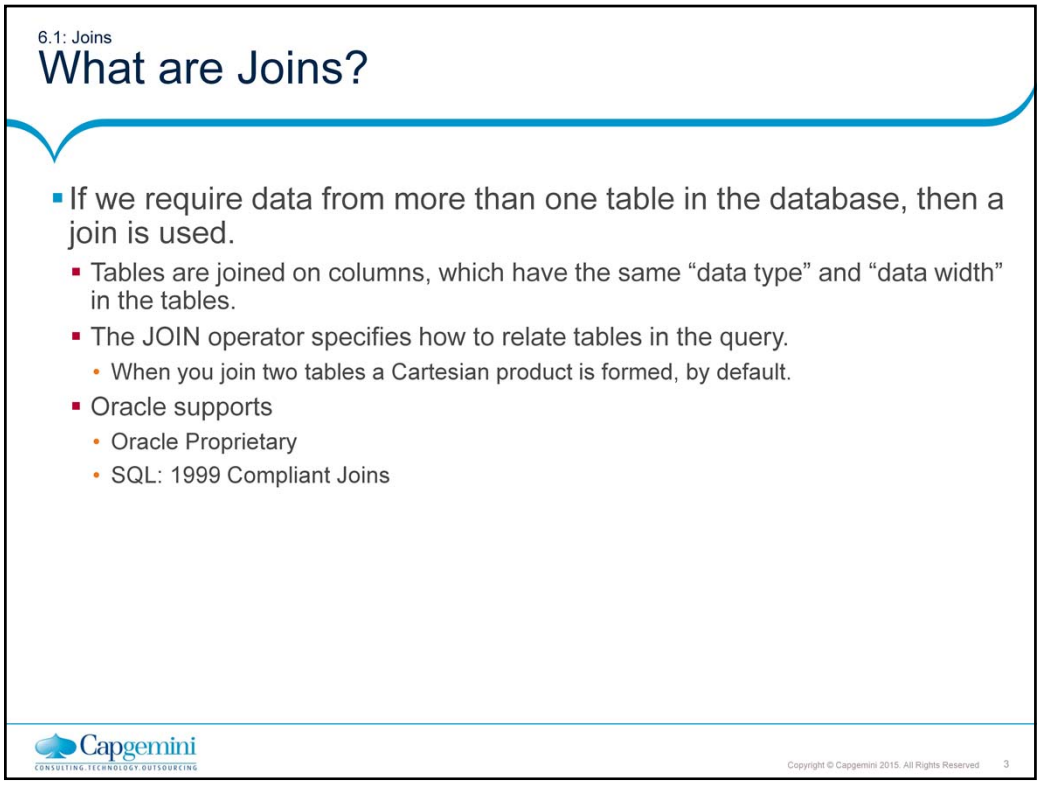
Lesson 06 Joins and Subqueries

Lesson Objectives

- To understand the following topics:
  - Joins
    - Oracle Proprietary Joins
    - SQL: 1999 Compliant Joins
  - Types of joins
  - Sub-queries
  - Tips and Tricks

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



Joins:

- JOINS make it possible to select data from more than one table by means of a single statement.
- The joining of tables is done in SQL by specifying the tables to be joined in the FROM clause of the SELECT statement.
- When you join two tables a Cartesian product is formed.
- The conditions for selecting rows from the product are determined by the predicates in the WHERE clause.
- All the subsequent WHERE, GROUP BY, HAVING, ORDER BY clauses work on this product.
- If the same table is used more than once in a FROM clause then “aliases” are used to remove conflicts and ambiquities. They are also called as “co-relation names” or “range variables”.

Joins (contd.):
Assume two tables:
TABLE F1

COL1	COL2
A	1
B	2
C	3

Table F2

COL1	COL2
X	100
Y	200

The statement SELECT * FROM F1,F2; results in:

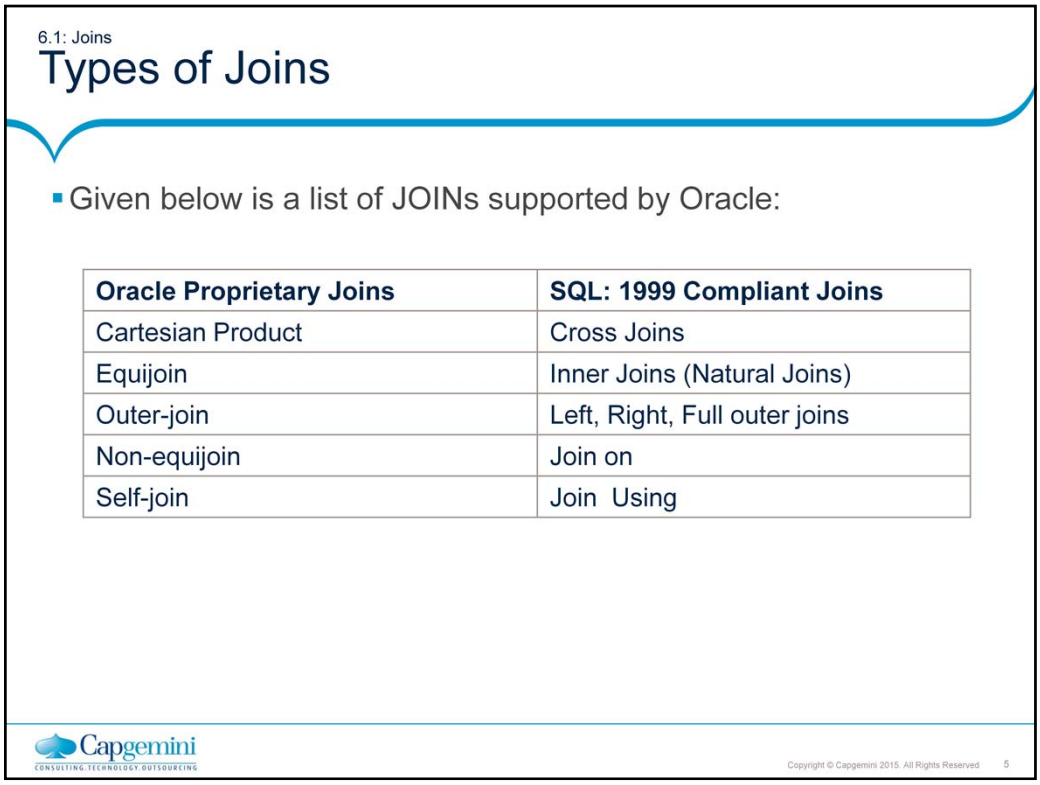
COL1	COL2	COL1	COL2
A	1	X	100
B	2	X	100
C	3	X	100
A	1	Y	200
B	2	Y	200
C	3	Y	200

6 rows selected

The statement SELECT * FROM F1 First_T, F1 Second_T; results in:

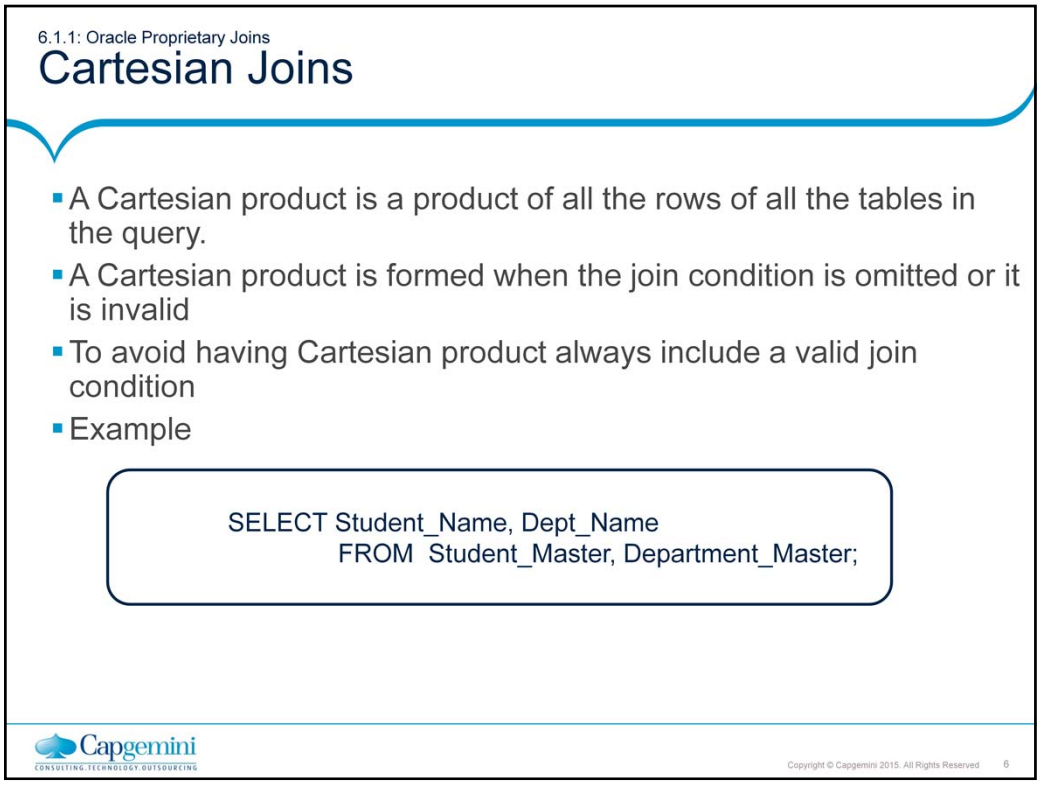
COL1	COL2	COL1	COL2
A	1	A	1
B	2	A	1
C	3	A	1
A	1	B	2
B	2	B	2
C	3	B	2
A	1	C	3
B	2	C	3
C	3	C	3

9 rows selected



Note:

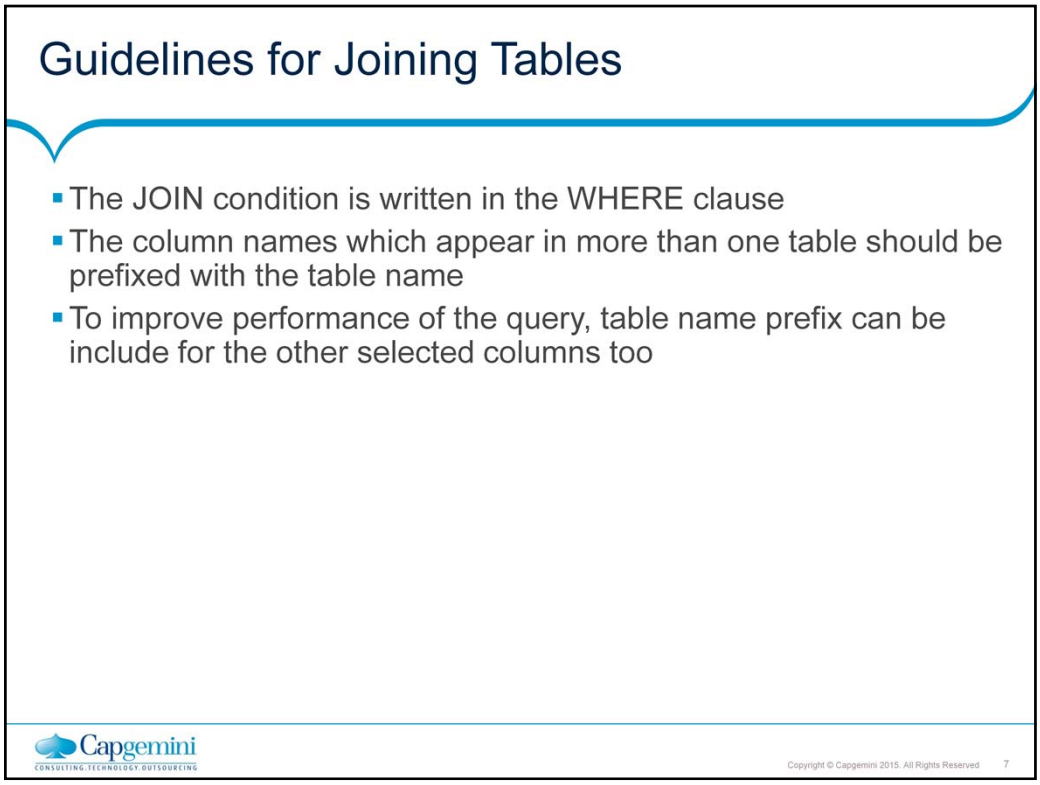
- Oracle9i onwards offers JOIN syntax that is SQL: 1999 compliant.
- Prior to the 9i release, the JOIN syntax was different from the ANSI standards.
- The new SQL: 1999 compliant JOIN syntax does not offer any performance benefits over the Oracle proprietary JOIN syntax that existed in prior releases.
- As we go ahead we will see both variations of joins supported by Oracle.



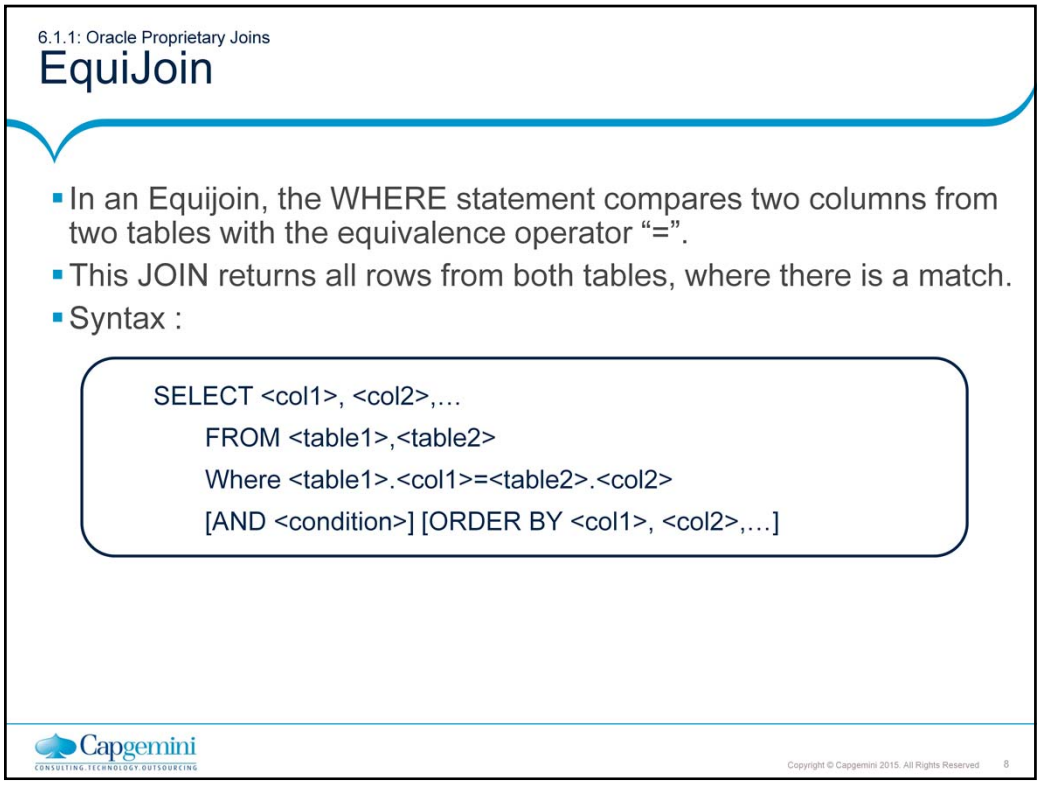
Cartesian Product

Whenever a join condition is completely omitted or is invalid a Cartesian product results. It displays all combinations of rows. A Cartesian product tends generates a large number of rows. Unless there is some specific need to combine all rows avoid a Cartesian product by including a valid join condition in the query

The example shown on the slide joins all rows of Student_Master and Department Master resulting in a Cartesian Join



Before we get on to Joins let us understand some basic guidelines to write Join Queries



Equi Join

• Equi Join which is sometimes also referred to as Inner Join or simple join is done by writing a join condition using the "=" operator

• Typically the tables are joint to get meaningful data.

• The join is based on the equality of column values in the two tables and therefore is called an Equijoin.

• To join together "n" tables, you need a minimum of "n-1" JOIN conditions.

For example: To join three tables, a minimum of two joins is required.

• In the syntax given in the slide:

Column1 in Table1 is usually the Primary key of that table.

Column2 in Table2 is a Foreign key in that table.

Column1 and Column2 must have the same data type, and for certain data types, they should have same size, as well.

6.1.1: Oracle Proprietary Joins

EquiJoin - Example

- Example 1: To display student code and name along with the
- department name to which they belong

```
SELECT Student_Code,Student_name,Dept_name
  FROM Student_Master ,Department_Master
  WHERE Student_Master.Dept_code =Department_Master.Dept_code;
```

- Example 2: To display student and staff name along with the department name to which they belong

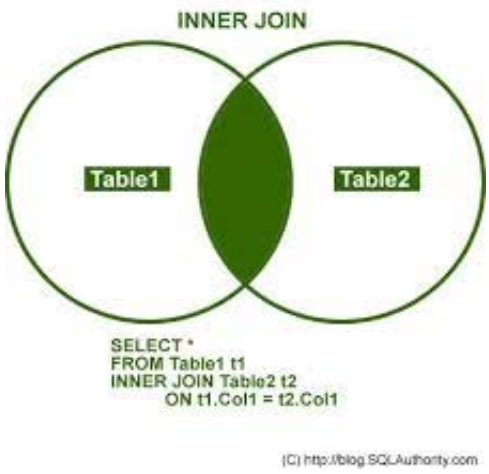
```
SELECT student_name,staff_name, dept_name
        FROM student_master, department_master,staff_master
        WHERE student_master.dept_code=department_master.dept_code
         and staff_master.dept_code=department_master.dept_code;
```

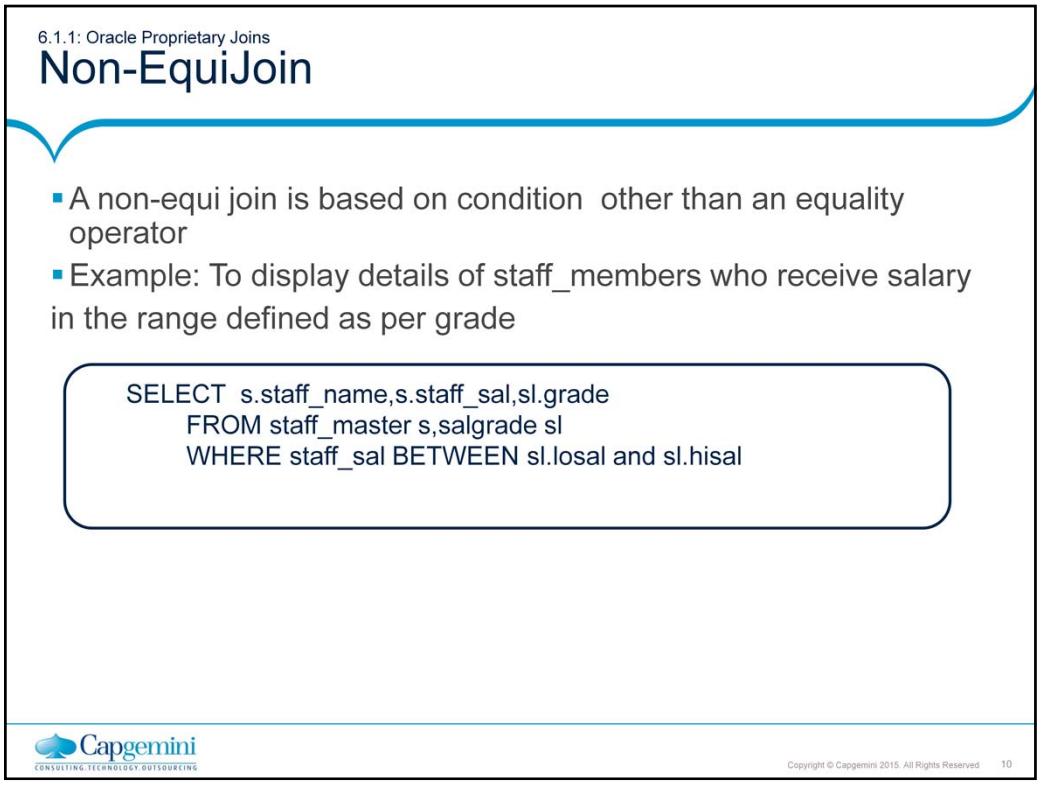
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Equi Join

- Frequently, these type of JOIN involves PRIMARY and FOREIGN key complements.
- You can also use table aliases to qualify column names in the SELECT and Join Condition





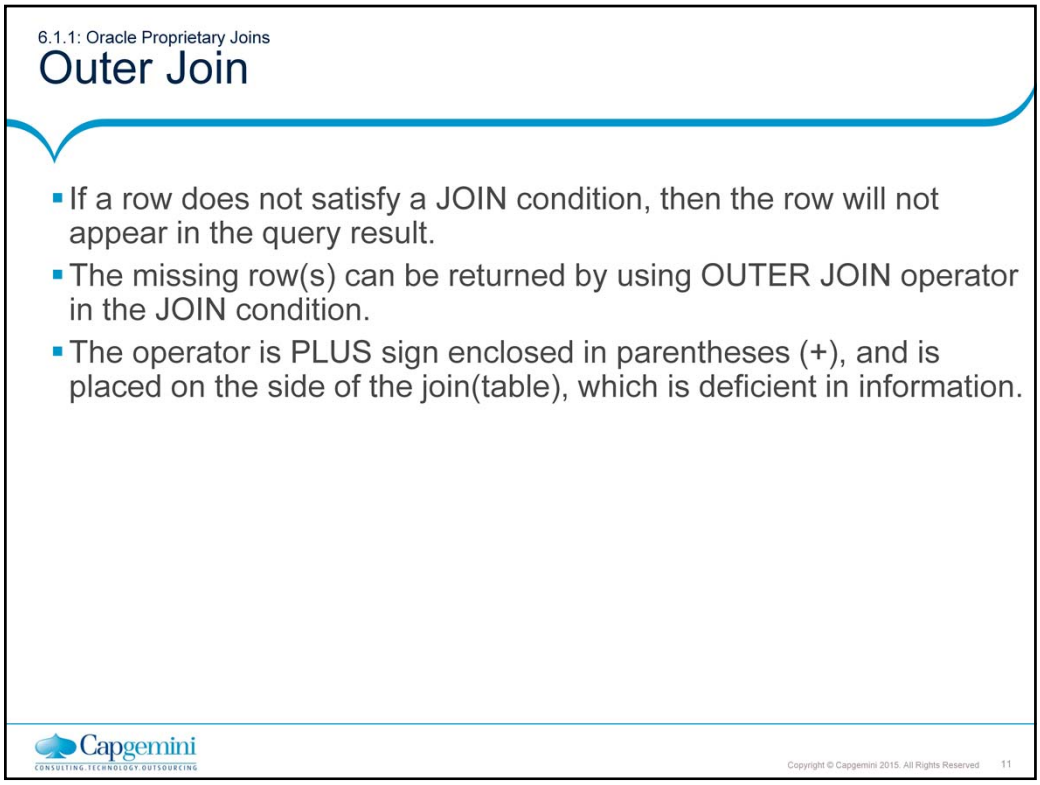
Non-Equijoin

- A Non-equijoin is a JOIN condition containing something other than an equality operator.
- The example on the slide shows a non-equijoin operation

Assume that we have a Salgrade table which is used to determine the range of salary for all staff member. The structure of the table is as follows:

Name	Type
GRADE	NUMBER
LOSAL	NUMBER
HISAL	NUMBER

So to display all the staff members who receive salary between the ranges specified in the salgrade table we will use a non-equijoin



Outer Join

- If a row does not satisfy the join condition, the row will not appear in the query result. In this situation outer join can be used
- Outer Joins are similar to Inner Joins. However, they give a bit more flexibility when selecting data from related tables. This type of join can be used in situations where it is desired to select “all rows from the table on the left or right ”, regardless whether they match the join condition
- Outer Join is an exclusive “union” of sets (whereas normal joins are intersection). OUTER JOINs can be simulated using UNIONS.
    - In a JOIN of two tables an Outer Join may be for the first table or the second table. If the Outer Join is taken on, say the DEPARTMENT_MASTER table, then each row of this table will be selected at least once whether or not a JOIN condition is satisfied.
- An Outer Join does not require each record in the two joint tables to have a matching record in the other table. The joint table retains each record — even if there is no other matching record.

6.1.1: Oracle Proprietary Joins

Outer Join

- Syntax
- Table1.column = table2.column (+) means OUTER join is taken on table1.
- The (+) sign must be kept on the side of the join that is deficient in information
- Depending on the position of the outer join (+), it can be denoted as Left Outer or Right outer Join

```
WHERE table1 <OUTER JOIN INDICATOR> = table 2
```

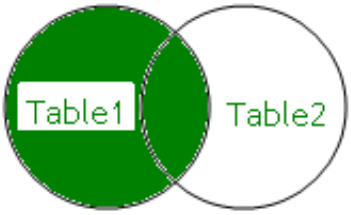
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

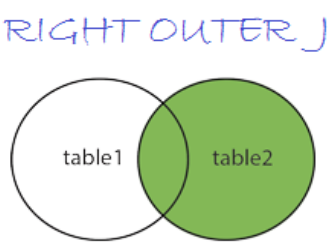
 12

Outer Join (contd.):

- The plus(+) operator can appear only on one side of the expression. It returns those rows from one table that have no direct match in the other table.
- One restriction on outer join is that you cannot use IN operator or the OR operator to create a complex condition

- Left and right outer join diagrams :





6.1.1: Oracle Proprietary Joins

Outer Join - Example

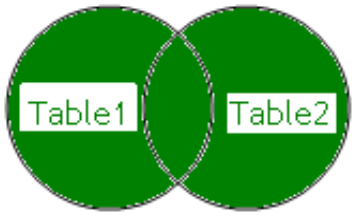
- To display Department details which have staff members and also display department details who do not have any staff members

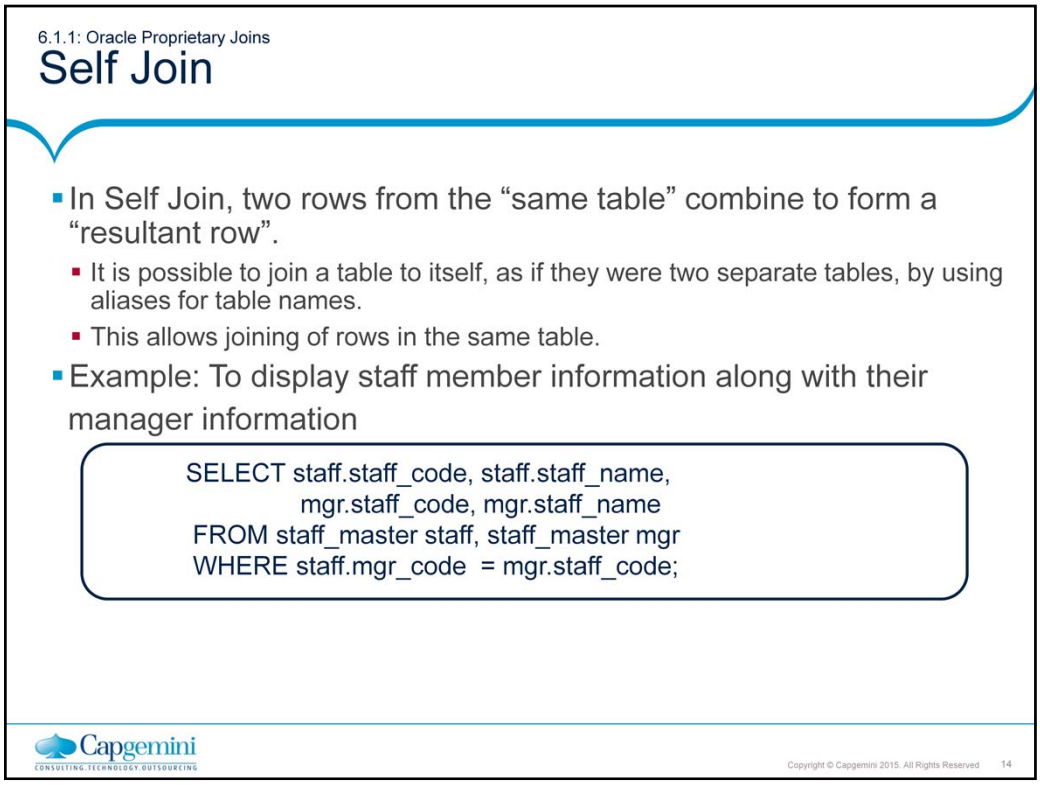
```
SELECT staff.staff_code,staff.Dept_Code,dept.Dept_name
  FROM Staff_master staff, Department_Master dept
  WHERE staff.Dept_Code(+) = dept.Dept_Code
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

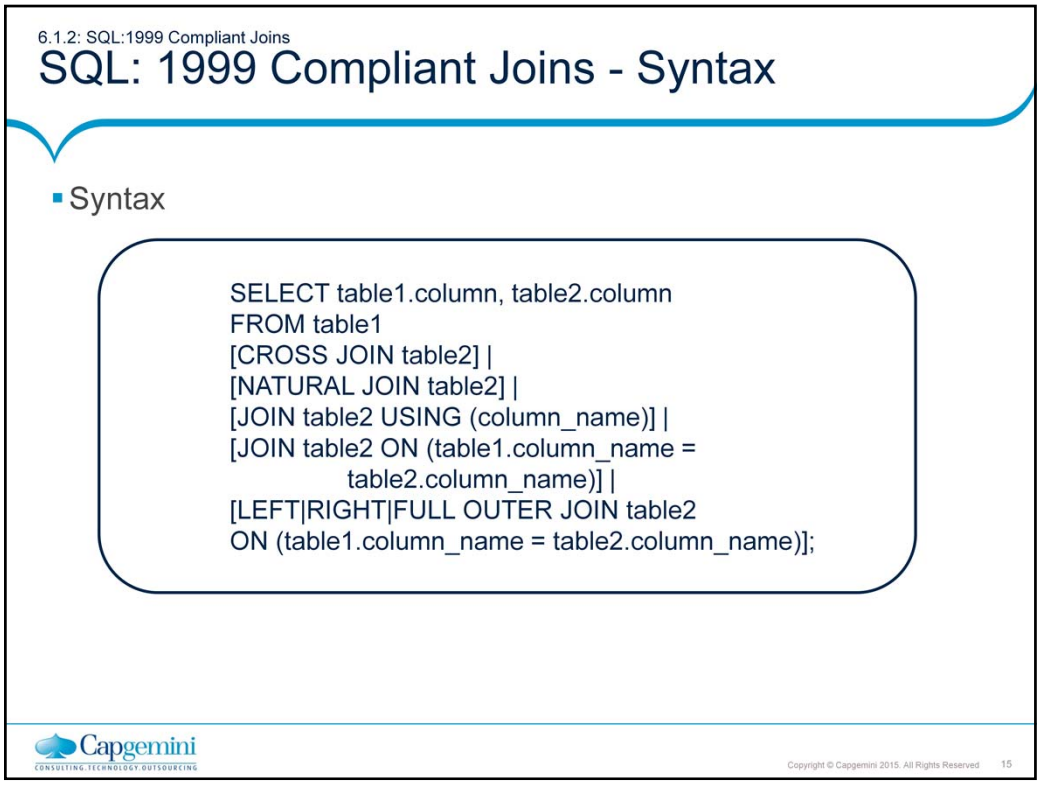
Full outer join diagram





Self Join:

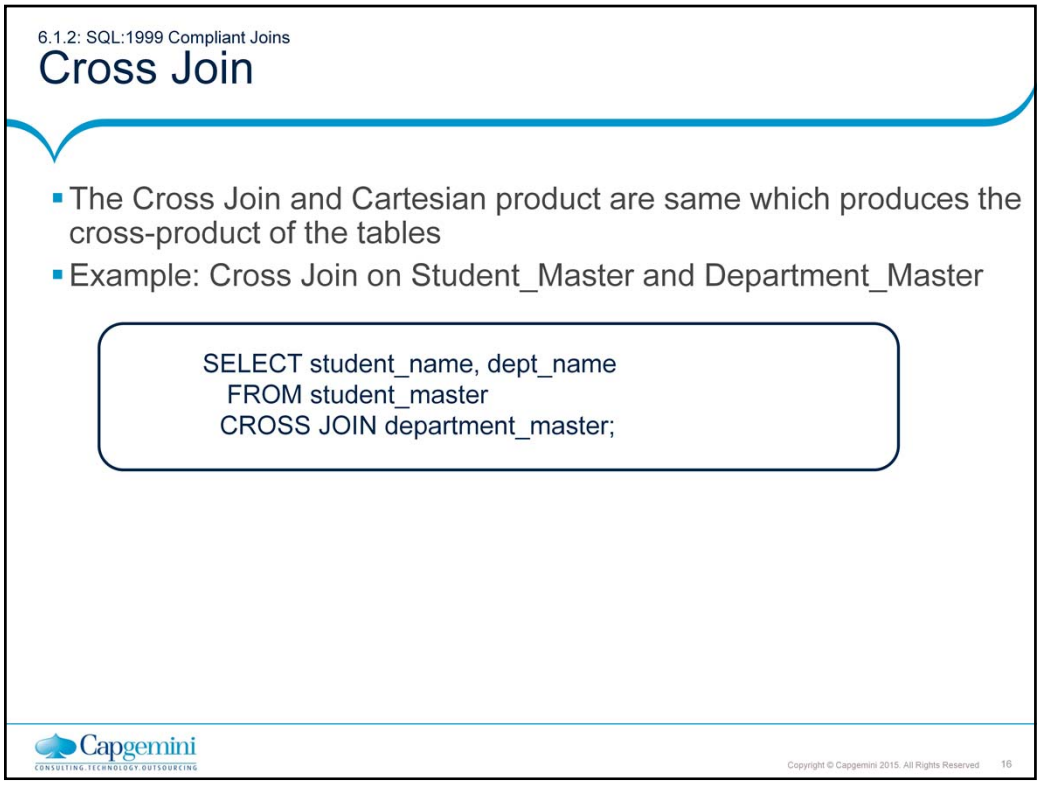
- Sometimes is required to join the table to itself. To join a table to itself, “two copies” of the same table have to be opened in the memory.
- Since the table names are the same, the second table will overwrite the first table. In effect, this will result in only one table being in memory.
  - This is because a table name is translated into a specific memory location.
- Hence in the FROM clause, the table name needs to be mentioned twice with an “alias”
  - These two table aliases will cause two identical tables to be opened in different memory locations.
  - This will result in two identical tables to be physically present in the computer memory.



SQL:1999 Compliant Joins

The SQL Compliant joins can obtain the similar results that we have discussed in the previous slides. They differ only in syntax.

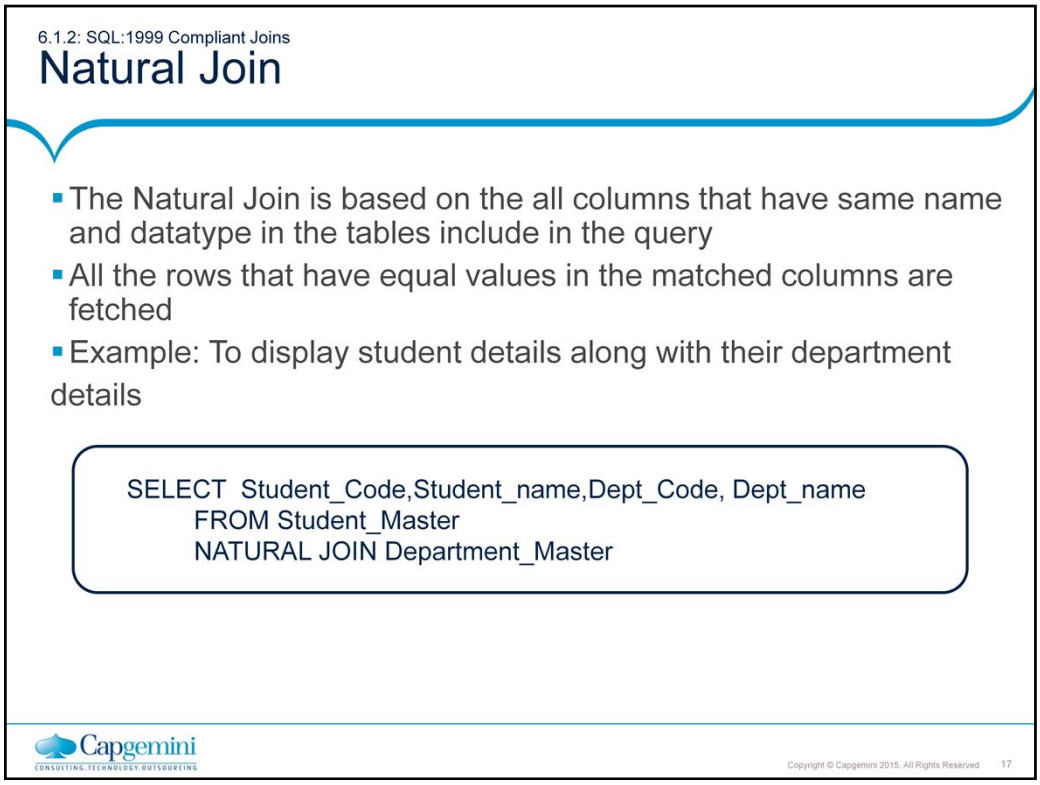
The SQL compliant joins are supported from Oracle 9i version onwards



Cross Join

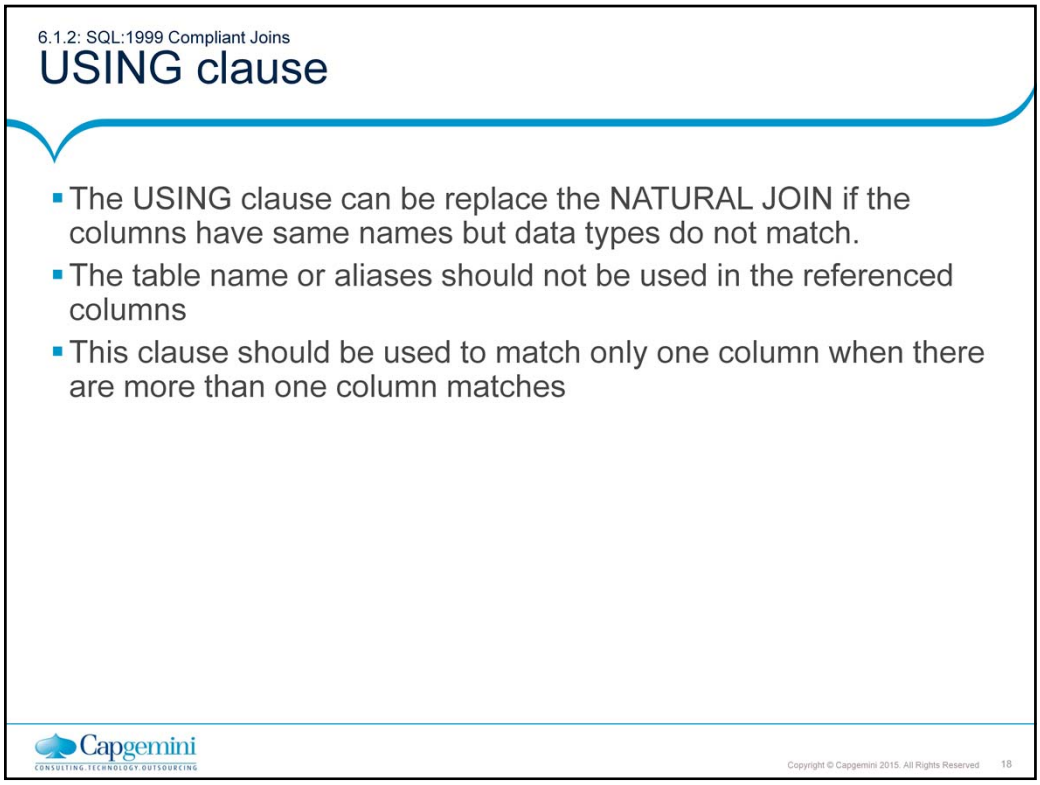
• The example on the slide create a cross product of the two tables. The query result is same as the following query:

```
SELECT student_name,dept_name
FROM Student_Master, Department_Master;
```



Natural Join

Prior to Oracle 9i, without explicitly specifying the columns of the corresponding tables a join was not possible. With the Natural Join clause, the join can happen automatically based on column names that match in name and datatype.
Oracle returns an error if the datatypes of the columns are different though they have the same name.
The Natural Join is same as EquiJoin.



USING clause

• The NATURAL JOIN returns an error if the datatypes of matching columns are different. In that case the USING clause can be used to specify only those columns on which the join has to done.
• The NATURAL JOIN and USING clause are mutually exclusive.
• The column used in USING clause cannot be used in WHERE clause

6.1.2: SQL:1999 Compliant Joins

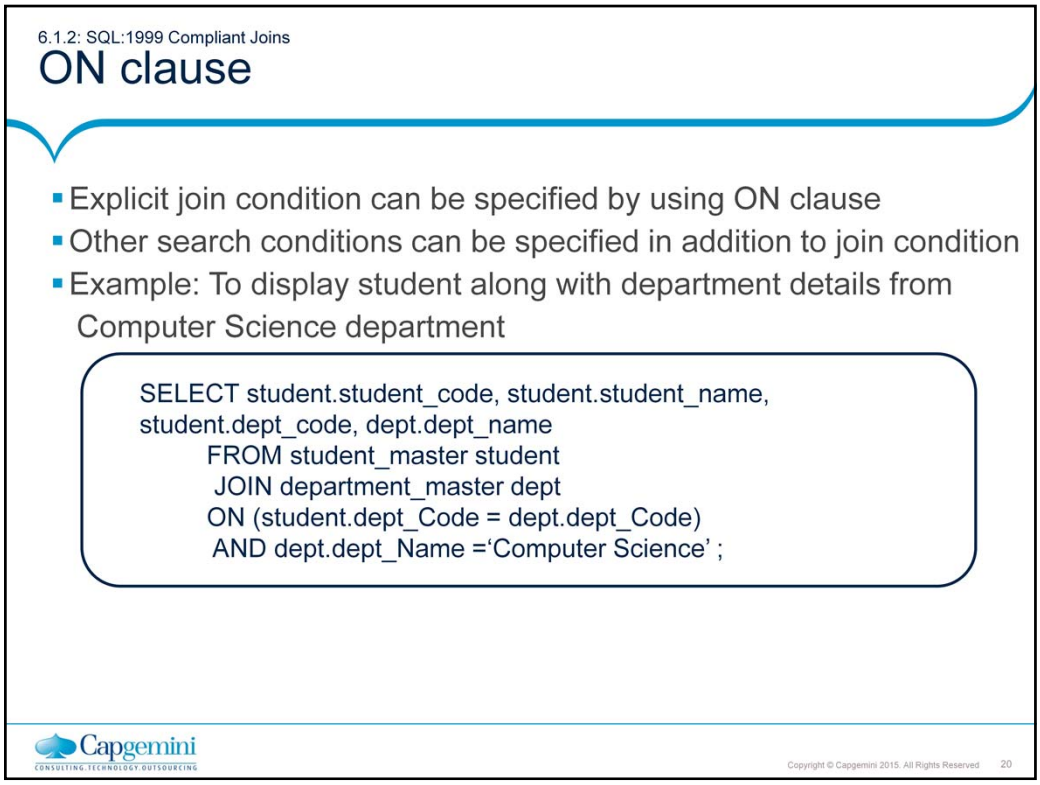
USING clause - Example

- Example 1: To display student details along with their department details. The department code does not match in datatype, hence the join is performed with the USING clause

```
SELECT student_code, student_name, dept_code, dept_name
        FROM student_master
        JOIN department_master
        USING (dept_code, dept_code);
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

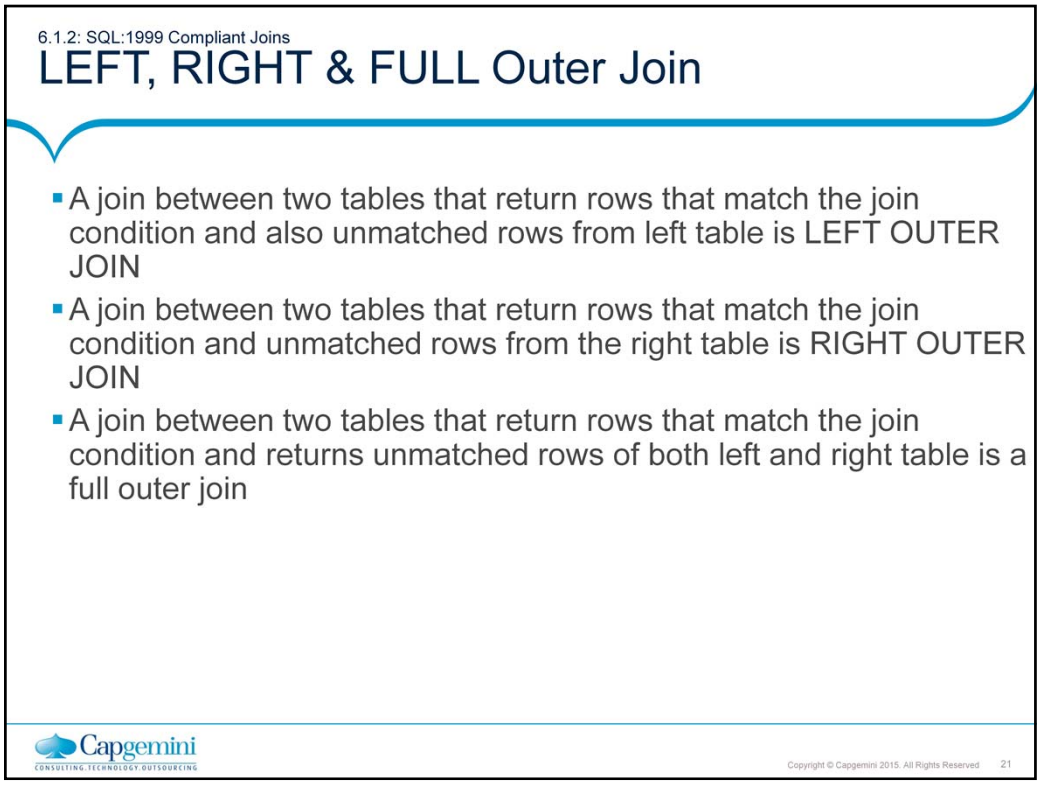
 19



ON clause

• With the ON clause you can specify join conditions separate from any other search conditions.

• The query is readable and easy to understand



LEFT, RIGHT and FULL OUTER JOIN

The SQL compliant outer join is similar to Oracle proprietary outer join except for the fact that it also has support to perform FULL OUTER JOIN

6.1.2: SQL:1999 Compliant Joins

LEFT, RIGHT & FULL Outer Join - Example

- Example 1: Display student & department details and also thos departments who do have students

```
SELECT s.student_code, s.dept_code, d.dept_name
        FROM student_master s
        RIGHT OUTER JOIN department_master d
        ON (s.dept_code = d.dept_code);
```

- Example 2 Display student & department details, also those students who are not assigned to any department

```
SELECT s.student_code, s.dept_code, d.dept_name
        FROM student_master s
        LEFT OUTER JOIN department_master d
        ON (s.dept_code = d.dept_code);
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

6.1.2: SQL:1999 Compliant Joins

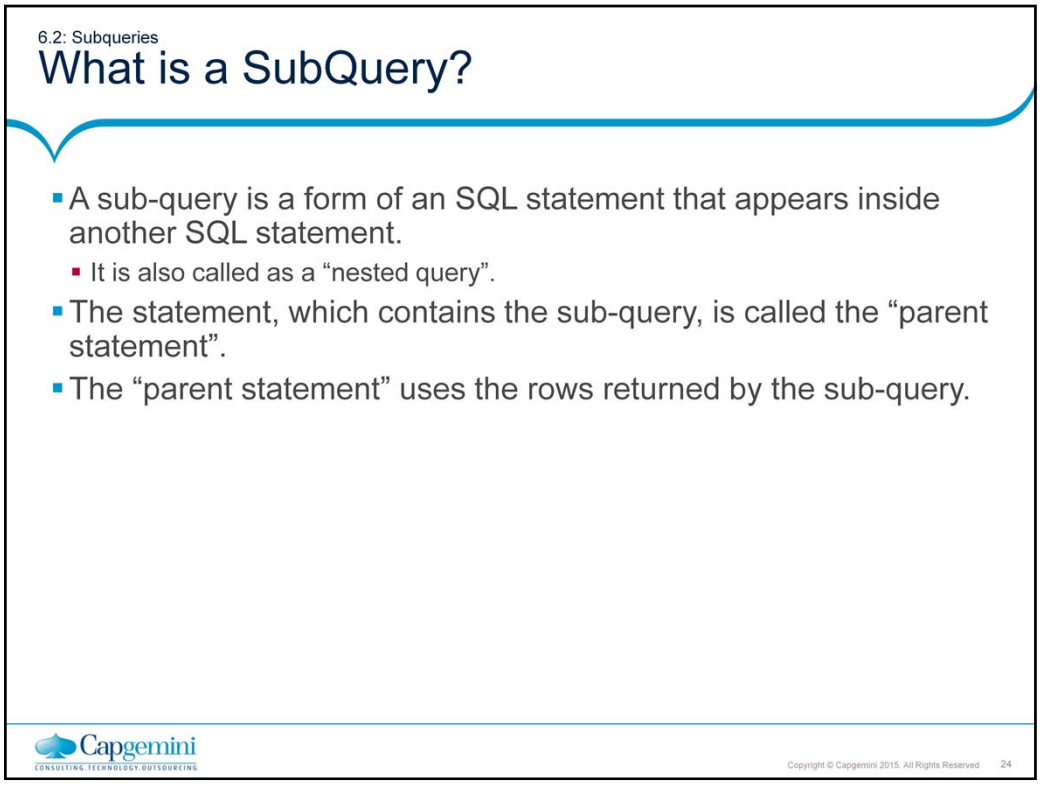
LEFT, RIGHT & FULL Outer Join - Example

- Example 3: Display student & department details. Also those departments who do have students and students who are not assigned to any department

```
SELECT s.student_code,s.dept_code,d.dept_name
        FROM student_master s
        FULL OUTER JOIN department_master d
        ON (s.dept_code = d.dept_code );
```

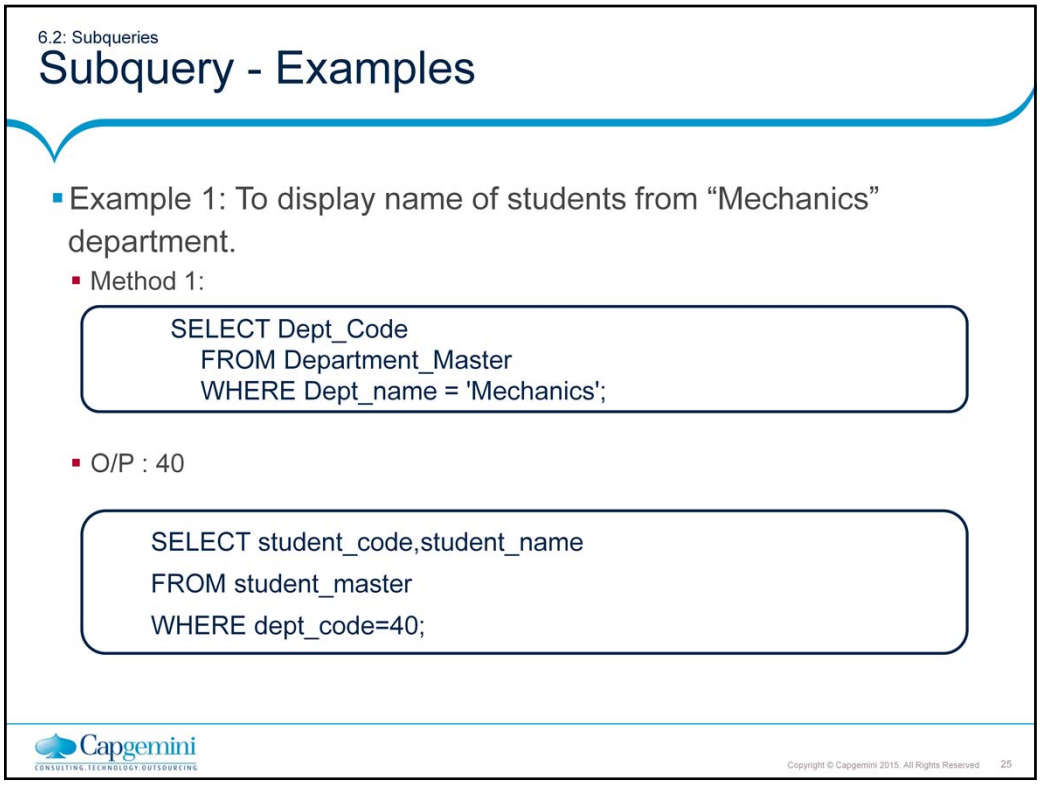
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

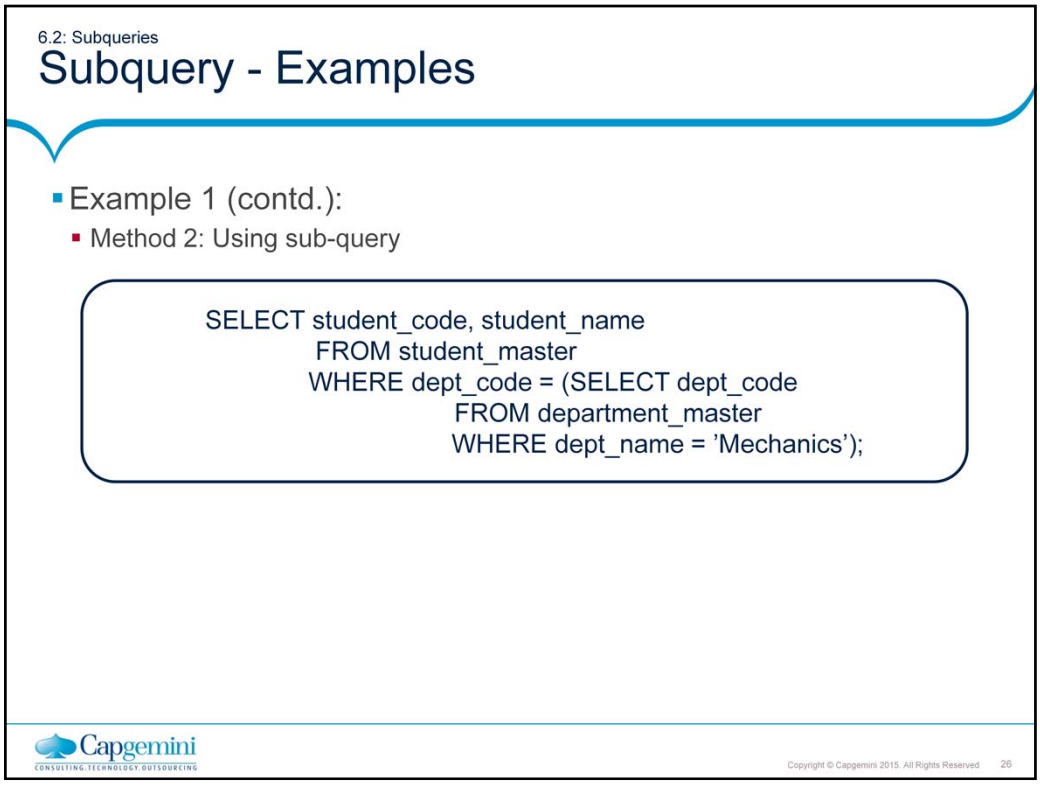


Sub-queries:

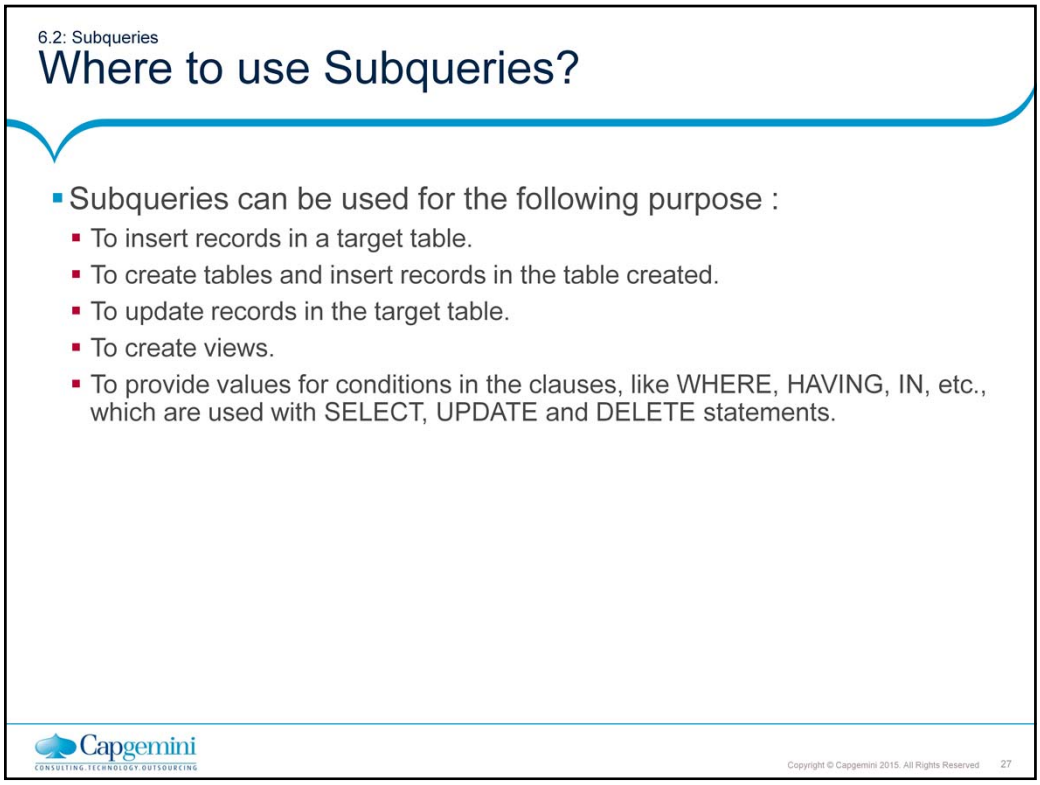
- As mentioned earlier, since a basic SQL query returns a relation, it can be used to construct composite queries.
- Such a SQL query, which is nested within another higher level query, is called a “sub-query”.
- This kind of a nested sub-query is useful in cases, where we need to select rows from tables based on a condition, which depends on the data stored in the table itself.
- These can be useful when you need to select rows from a table with a condition that depends on data within the table itself.



Consider the example given on the slide. We want to find details of students from “Mechanics” department. As you can see two queries are used to resolve these problem. Instead of that you can resolve this problem by combining both the queries into one as shown on the next slide.



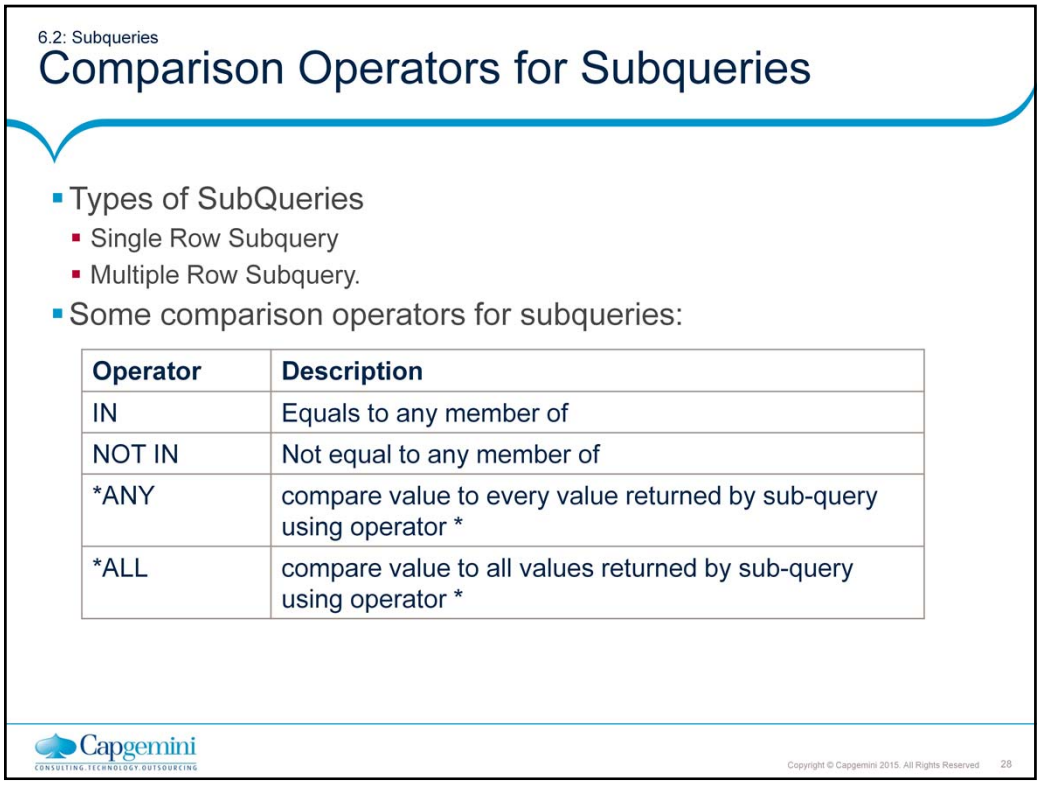
The problem is resolved using a one query inside another. The inner query is called as a subquery or nested query. The subquery result is used by the outer query. The subqueries can have more levels of nesting. The innermost query will execute first. The subquery execute only once before the outer query



• When the WHERE clause needs a set of values which can be only obtained from another query, the Sub-query is used. In the WHERE clause it can become a part of the following predicates

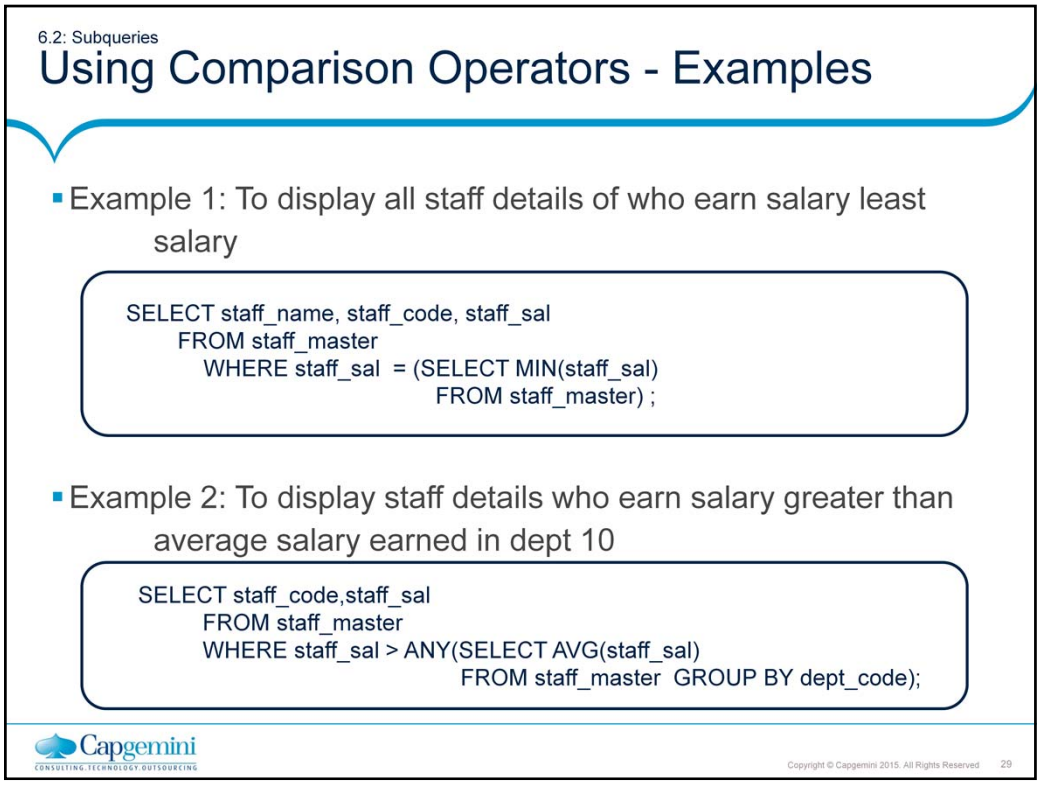
- ➢COMPARISON Predicate
- ➢IN Predicate
- ➢ANY or ALL Predicate
- ➢EXISTS Predicate.

• It can be also used as a part of the condition in the HAVING clause.



Sub-queries by using Comparison operators:

- Sub-queries are divided as “single row” and “multiple row” sub-queries. While single row comparison operators (=, >, >=, <, <=, <>) can only be used in single row sub-queries, multiple row sub-queries can use IN, ANY or ALL.
- **For example**: The assignment operator (=) compares a single value to another single value. In case a value needs to be compared to a list of values, then the IN predicate is used. The IN predicate helps reduce the need to use multiple OR conditions.
- The NOT IN predicate is the opposite of the IN predicate. This will select all rows where values do not match the values in the list.
- The FOR ALL predicate is evaluated as follows:
    1. True if the comparison is true for every value of the list of values.
    2. True if sub-query gives a null set (No values)
    3. False if the comparison is false for one or more of the list of values generated by the sub-query.
- The FOR ANY predicate is evaluated as follows
    1. True if the comparison is true for one or more values generated by the sub-query.
    2. False if sub-query gives a null set (No values).
    3. False if the comparison is false for every value of the list of values generated by the sub-query.



• For Single Row Subquery the sub-query should result in one value of the same data type as the left-hand side.
•Similarly for Multiple Row Subquery the list of values generated by the sub-query should be of same data type as the left-hand side
• The slide above shows examples of Subqueries. The first query is an example of Single Row Subquery and the second is an example of Multiple row Subquery

6.3: Tips and Tricks

Quick Guidelines

- For Using Subqueries
  - Should be enclosed in parenthesis
  - They should be placed on the right side of the comparison condition
  - Use operator carefully. Single Row operators for Single Row Subquery and Multiple Row operator for Multiple Row Subquery

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 30

6.3: Tips and Tricks

Quick Guidelines

- If Single row operators are used for a sub query that returns multiple rows, Oracle would throw an error
- Restrict using the NOT IN clause, which offers poor performance because the optimizer has to use a nested table scan to perform this activity.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31

Summary

- In this lesson, you have learnt:
  - Joins
  - Oracle Proprietary Joins
  - SQL: 1999 Compliant Joins
  - Sub-queries

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 32

Review – Match the Following

1. Equi Join	a. is based on any other operator other than equality
2. Non-equijoin	b. Is based on equality operator
3. Outer Join	c. Joins the table to itself
4. Self Join	d. includes a “+” operator with equality operator

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 33

Review – Questions

- Question 1: The SQL compliant join which is same as EquiJoin.
  - Option 1: Cross Join
  - Option 2: Natural Join
  - Option 3: Full Outer Join
- Question 2: A sub-query is also sometimes termed as _____.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 34

Review – Questions

- Question 3: A sub-query can be used for creating and inserting records.
  - True / False
- Question 4: If a sub-query returns multiple values, then the valid operators is/are ___.
  - Option 1: =
  - Option 2: IN
  - Option 3: >
  - Option 4: Any

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 35

DBMS/SQL

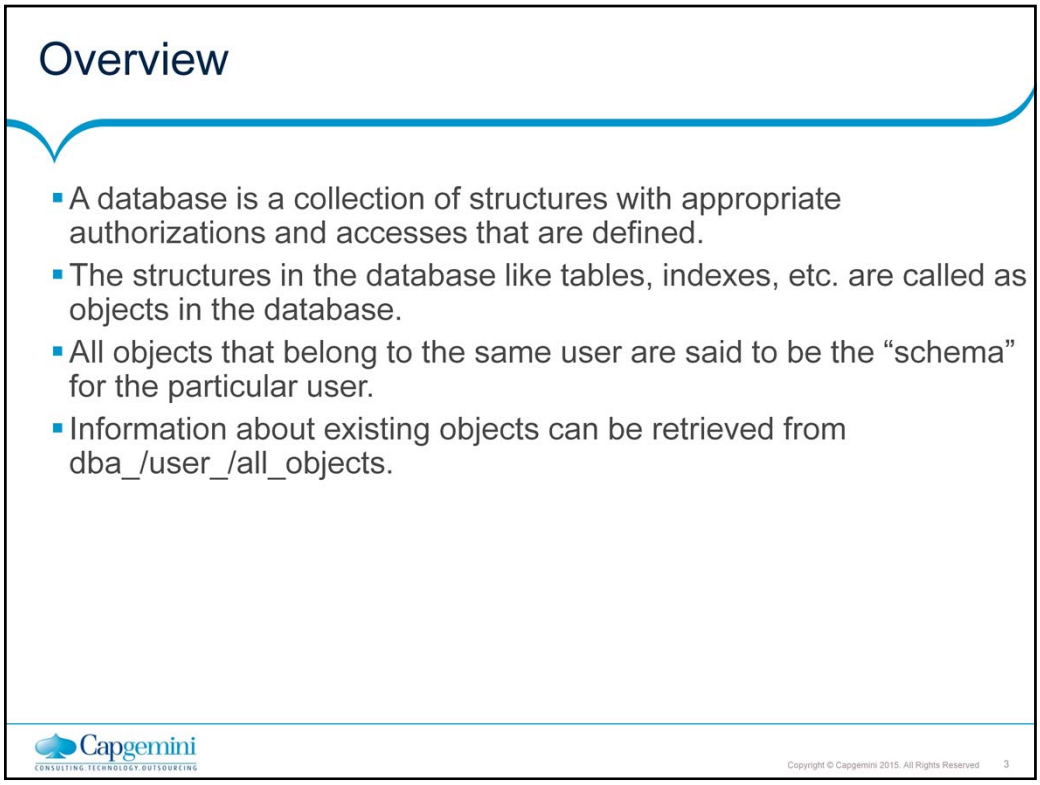
Lesson 07 Database Objects

Lesson Objectives

- To understand the following Database Objects:
  - Basic Data Types
  - Data Integrity
  - Examples of CREATE TABLE
  - Examples of ALTER TABLE
  - Database Objects
  - Index
  - Synonym
  - Sequence
  - View
  - Deleting Database Objects
  - Tips and Tricks

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

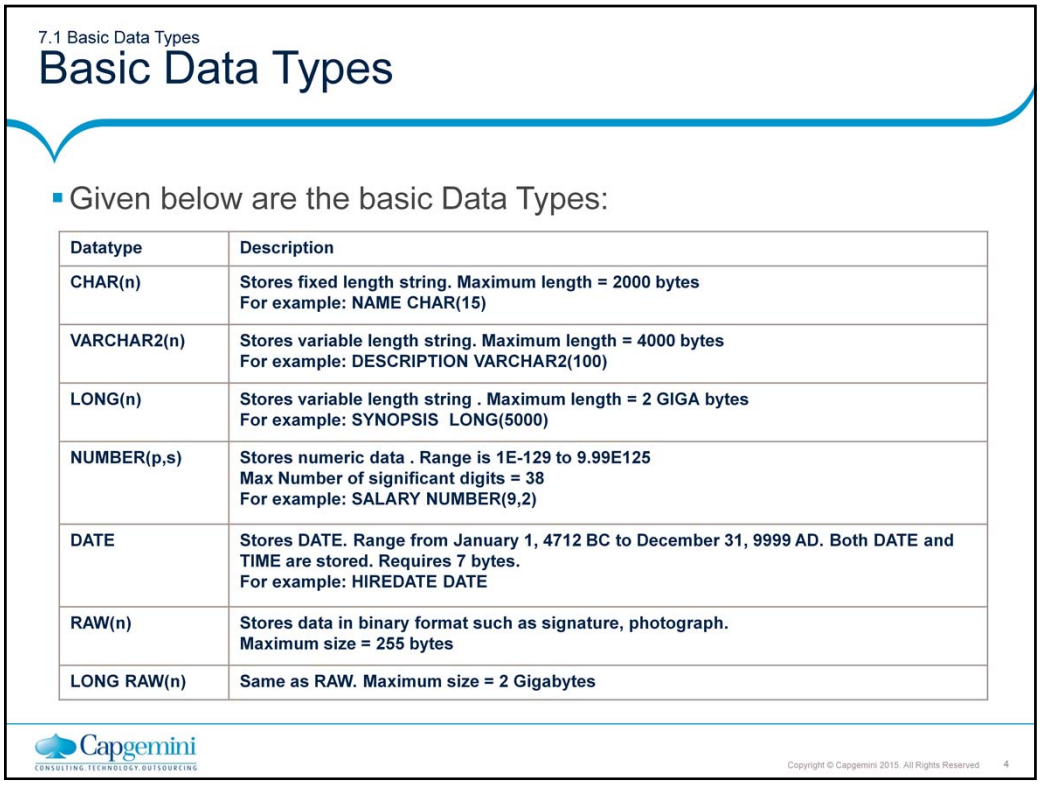
 2



Database Objects:

Following is a list of Database objects:

- Tables
- Views
- Indexes
- Clusters
- Synonyms
- Sequences
- Procedures
- Functions
- Packages
- Triggers .

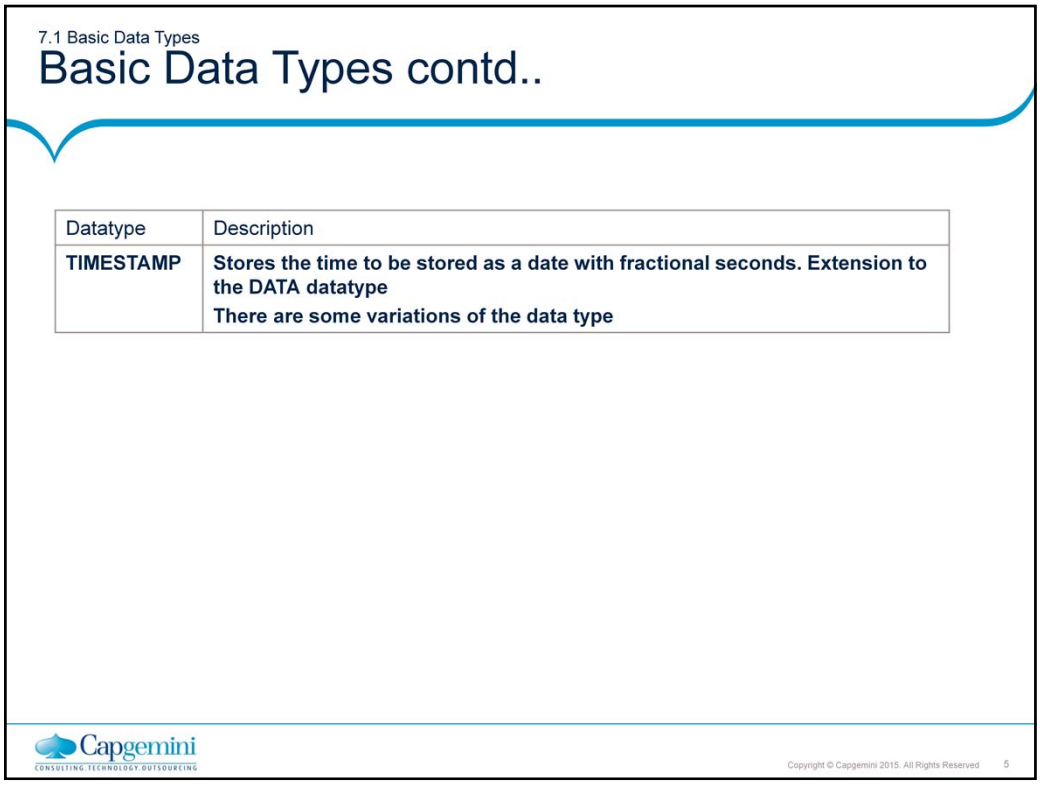


Datatype	Description
CHAR(n)	Stores fixed length string. Maximum length = 2000 bytes For example: NAME CHAR(15)
VARCHAR2(n)	Stores variable length string. Maximum length = 4000 bytes For example: DESCRIPTION VARCHAR2(100)
LONG(n)	Stores variable length string . Maximum length = 2 GIGA bytes For example: SYNOPSIS LONG(5000)
NUMBER(p,s)	Stores numeric data . Range is 1E-129 to 9.99E125 Max Number of significant digits = 38 For example: SALARY NUMBER(9,2)
DATE	Stores DATE. Range from January 1, 4712 BC to December 31, 9999 AD. Both DATE and TIME are stored. Requires 7 bytes. For example: HIREDATE DATE
RAW(n)	Stores data in binary format such as signature, photograph. Maximum size = 255 bytes
LONG RAW(n)	Same as RAW. Maximum size = 2 Gigabytes

Basic Data Types supported in SQL:

Scalar Data Types:

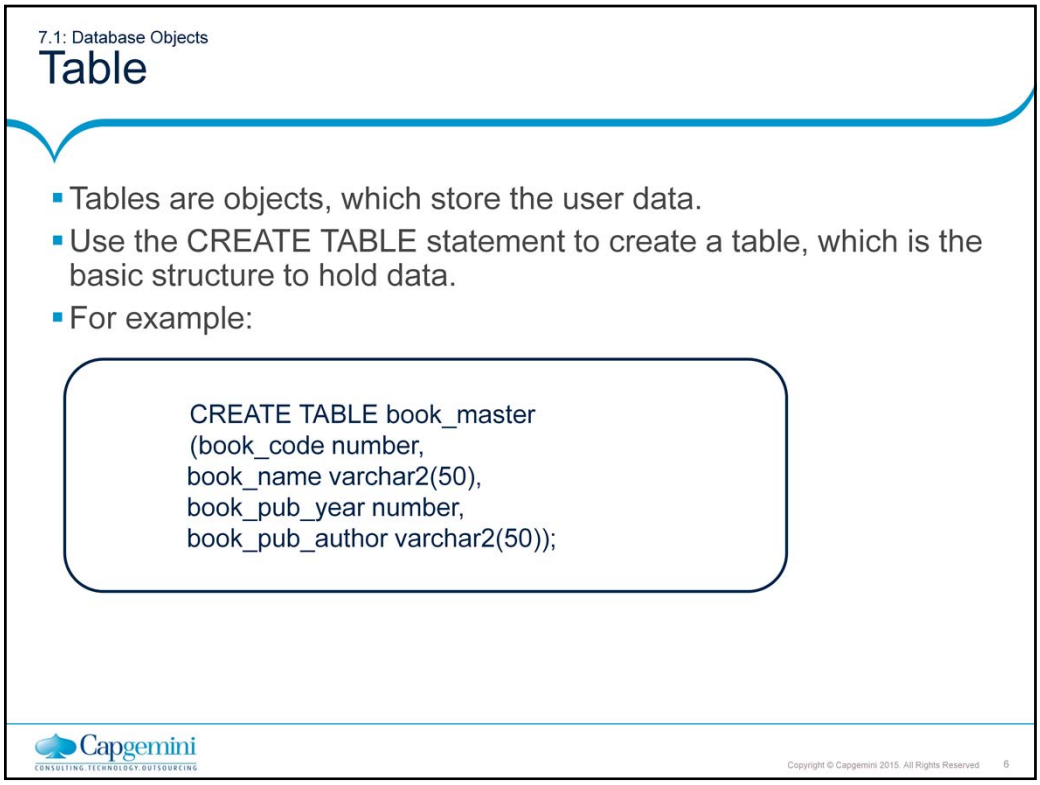
- The traditional Data Types are called “Scalar Data Types”. The slide discusses some of the Scalar Data Types.



Basic Data Types supported in SQL:

The TimeStamp data type is introduced in Oracle 9i. This data type provides support for time zones. It is an extension of the DATE datatype.

TIMESTAMP [(fractional_seconds_precision)] – It stores the year, month & day of the date data type. In addition to that it also stores hour, minute, second and fractional second value.



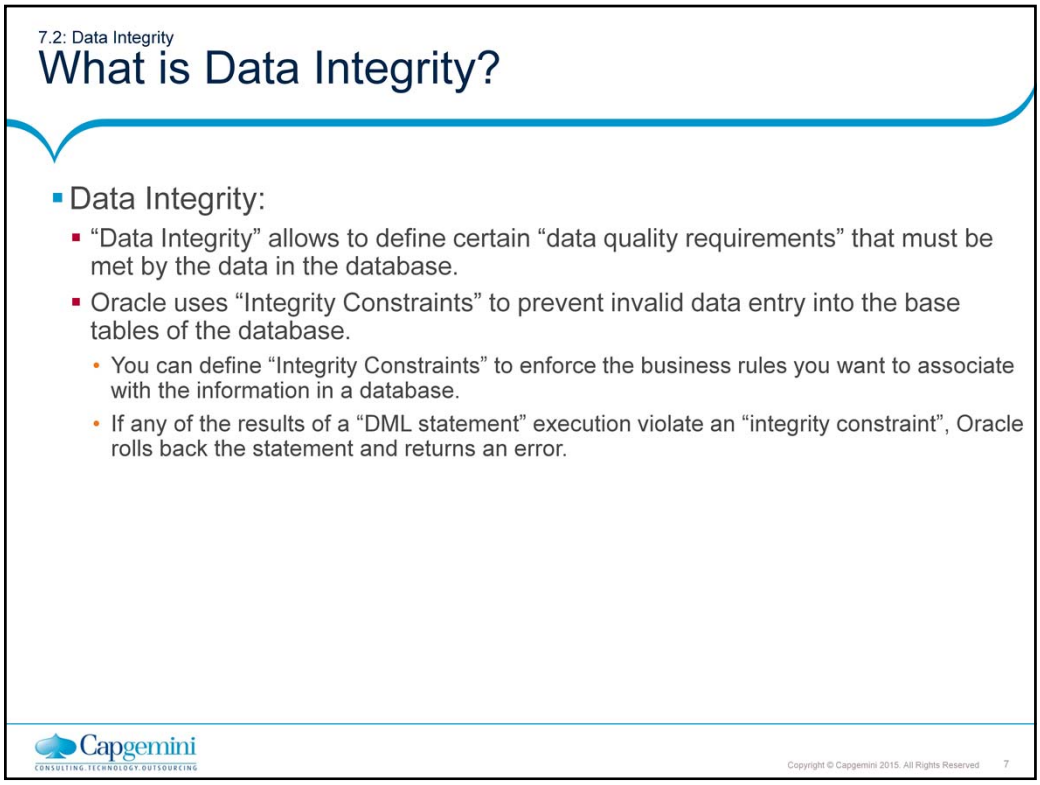
Creating tables is done with the create table command. You can add rows to a table with the INSERT statement, after creating a table.
The above create table command does the following:

- Defines the table name
- Defines the columns in the table and the datatypes of those columns

In above example, we create a table called BOOK_MASTER which has 4 columns. The first column and third column is defined as NUMBER datatype. This means we will be storing numbers in this column. The second and fourth columns are of VARCHAR2 datatype. We will be storing text data in these columns
Syntax:

```
CREATE TABLE table_name
(
{col_name.col_datatype [[CONSTRAINT
const_name][col_constraint]]},...
[table_constraint],...
)
[AS query]
```

- If a table is created as shown in the slide, then there is no restriction on the data that can be stored in the table.
- However, if we wish to put some restriction on the data, which can be stored in the table, then we must supply some “constraints” for the columns. We will see the Constraints as next topic.



Data Integrity: Integrity Constraint

- An Integrity Constraint is a declarative method of defining a rule for a column of a table.

Example of Data Integrity:

- Assume that you define an "Integrity Constraint" for the Staff_Sal column of the Staff_Master table.
- This Integrity Constraint enforces the rule that "no row in this table can contain a numeric value greater than 10,000 in this column".
- If an INSERT or UPDATE statement attempts to violate this "Integrity Constraint", Oracle rolls back the statement and returns an "information error" message.

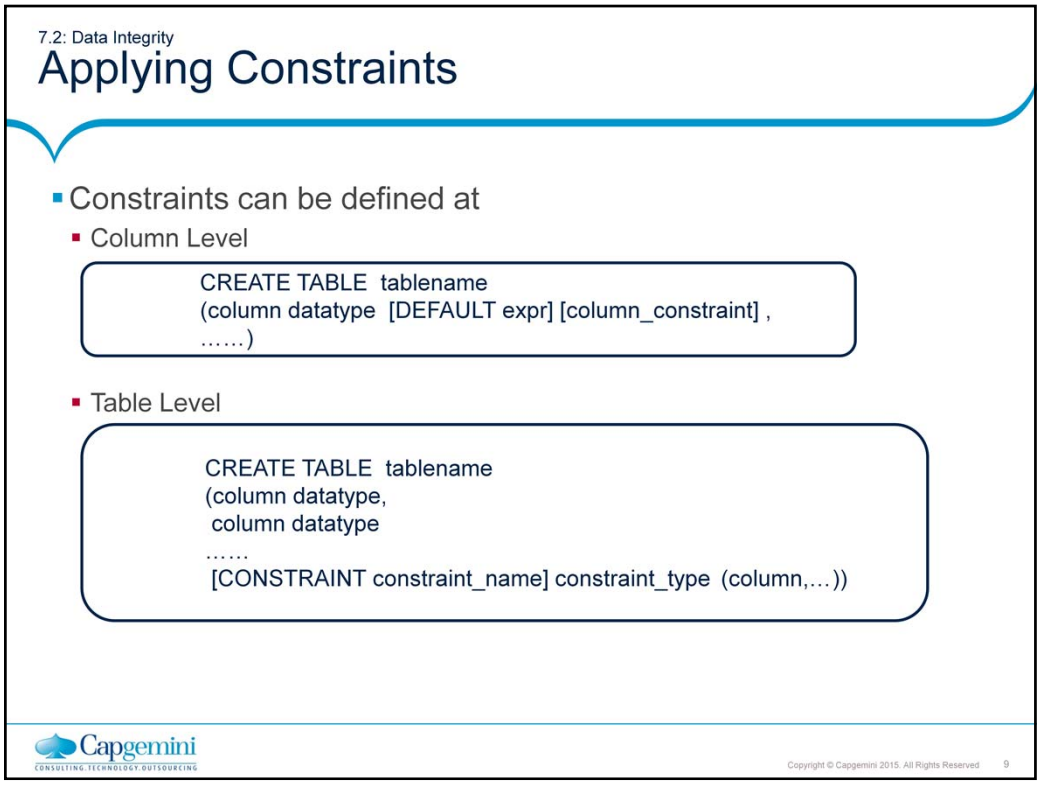
7.2: Data Integrity

Advantages

- Advantages of Integrity Constraints:
  - Integrity Constraints have advantages over other alternatives. They are:
    - Enforcing “business rules” in the code of a database application.
    - Using “stored procedures” to completely control access to data.
    - Enforcing “business rules” with triggered stored database procedures.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

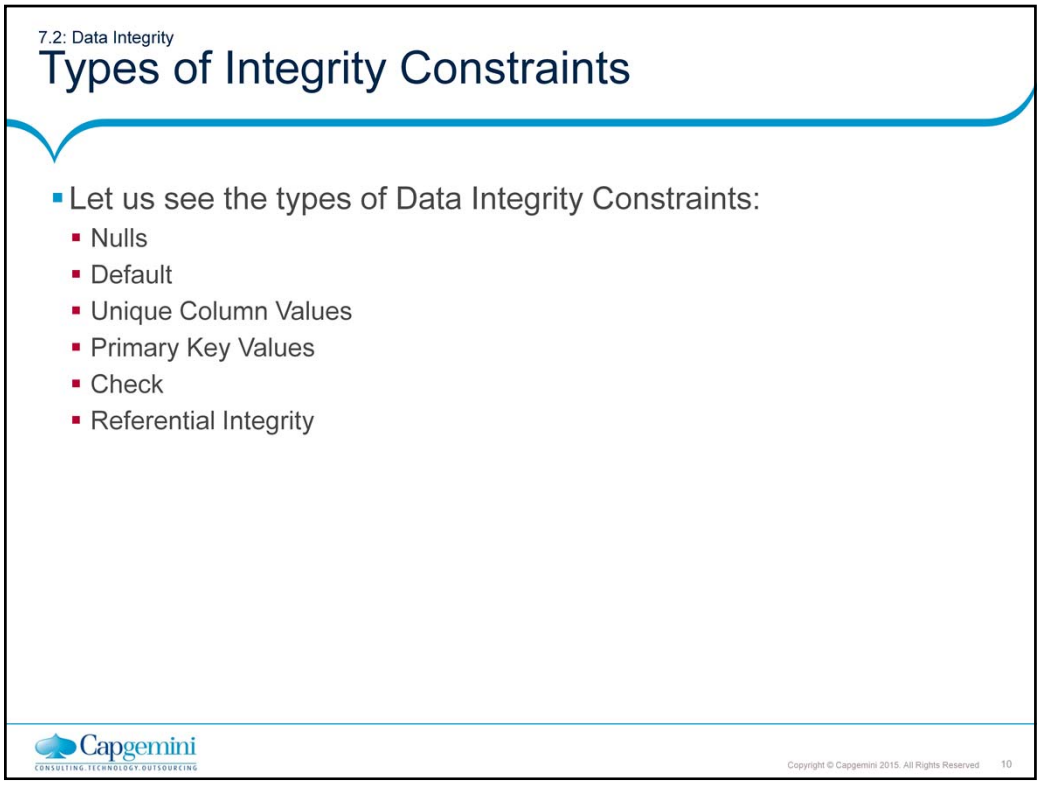


Applying Constraints:

In Oracle you can apply constraints

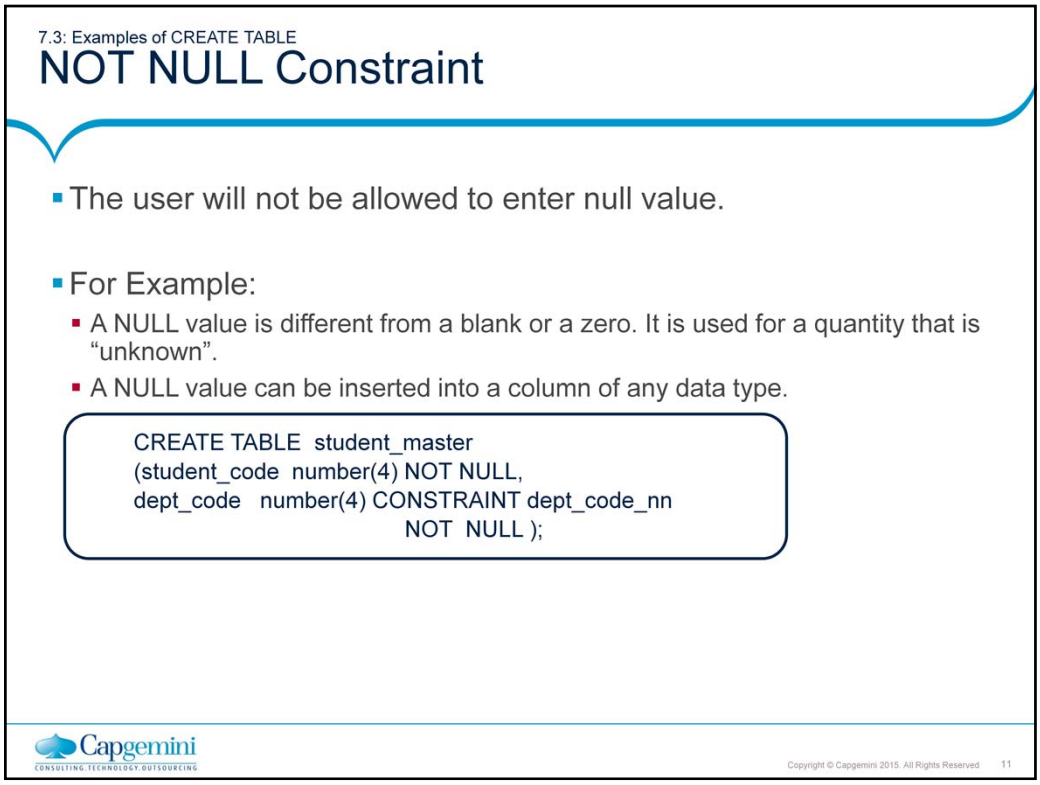
Column Level - References a single column and is defined within a specification for the owning column; can be any type of integrity constraint

Table Level - References one or more columns and is defined separately from the definitions of the columns in the table; can define any constraints except NOT NULL



Types of Data Integrity:

- Oracle supports the following Integrity Constraints:
  - NOT NULL constraints for the rules associated with nulls in a column. "Null" is a rule defined on a single column that allows or disallows, inserts or updates on rows containing a "null" value (the absence of a value) in that column.
  - UNIQUE key constraints for the rule associated with unique column values. A "unique value" rule defined on a column (or set of columns) allows inserting or updating a row only if it contains a "unique value" in that column (or set of columns).
  - PRIMARY KEY constraints for the rule associated with primary identification values. A "primary key" value rule defined on a key (a column or set of columns) specifies that "each row in the table can be uniquely identified by the values in the key".
  - FOREIGN KEY constraints for the rules associated with referential integrity. A "Referential Integrity" rule defined on a key (a column or set of columns) in one table guarantees that "the values in that key, match the values in a key in a related table (the referenced value)". Oracle currently supports the use of FOREIGN KEY integrity constraints to define the referential integrity actions, including:
    - update and delete No Action
    - delete CASCADE
    - delete SET NULL
  - CHECK constraints for complex integrity rules



NOT NULL constraint:

- Often there may be records in a table that do not have values for every field.
  - This could be because the information is not available at the time of the data entry or because the field is not applicable in every case.
- If the column is created as NULLABLE, in the absence of a user-defined value the DBMS will place a NULL value in the column.
- A NULL value is different from a blank or a zero. It is used for a quantity that is "unknown".
- "Null" is a rule defined on a single column that allows or disallows, inserts or updates on rows containing a "null" value (the absence of a value) in that column.
- A NULL value can be inserted into a column of any data type.
- **Principles of NULL values:**
  - Setting a NULL value is appropriate when the "actual value" is unknown, or when a value is not meaningful.
  - A NULL value is not equivalent to the value of "zero" if the data type is number, and it is not equivalent to "spaces" if the data type is character.
  - A NULL value will evaluate to NULL in any expression
    For example: NULL multiplied by 10 is NULL.
  - NULL value can be inserted into columns of any data type.
  - If the column has a NULL value, Oracle ignores any UNIQUE, FOREIGN KEY, and CHECK constraints that may be attached to the column.

7.3: Examples of CREATE TABLE

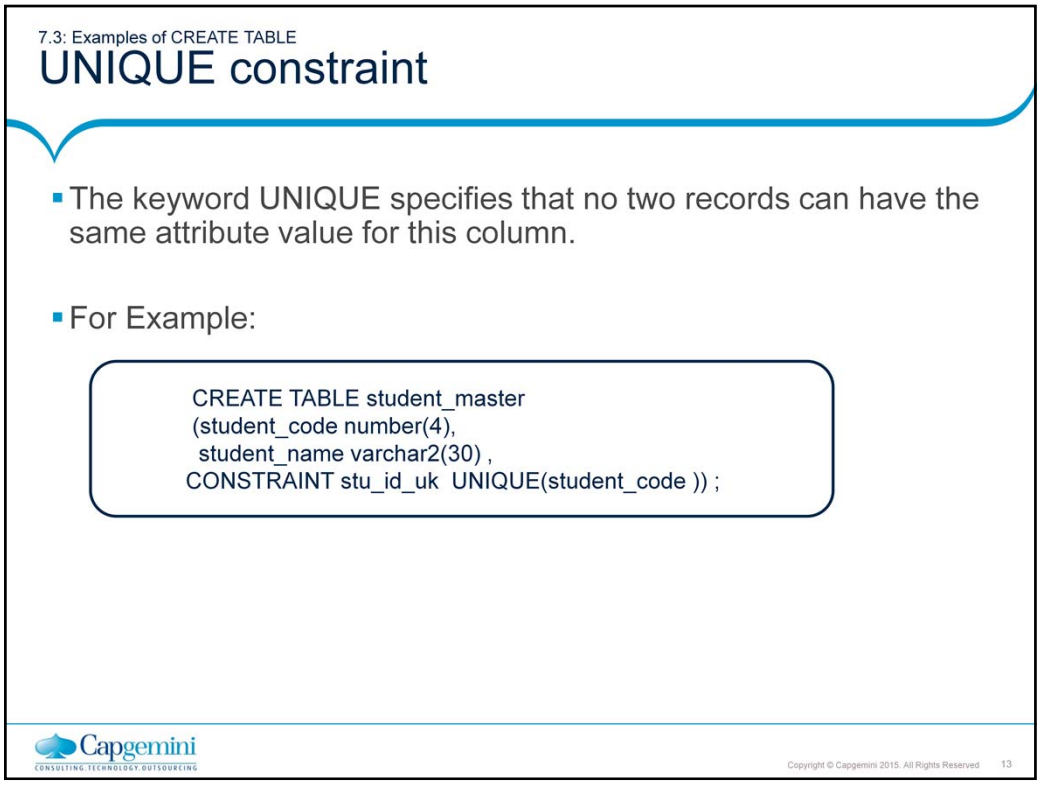
DEFAULT clause

- If no value is given, then instead of using a “Not Null” constraint, it is sometimes useful to specify a default value for an attribute.
- For Example:
  - When a record is inserted the default value can be considered.

```
CREATE TABLE  staff_master(
Staff_Code number(8) PRIMARY KEY,
Staff_Name varchar2(50) NOT NULL,
Staff_dob date,
Hiredate date DEFAULT sysdate,
…..)
```

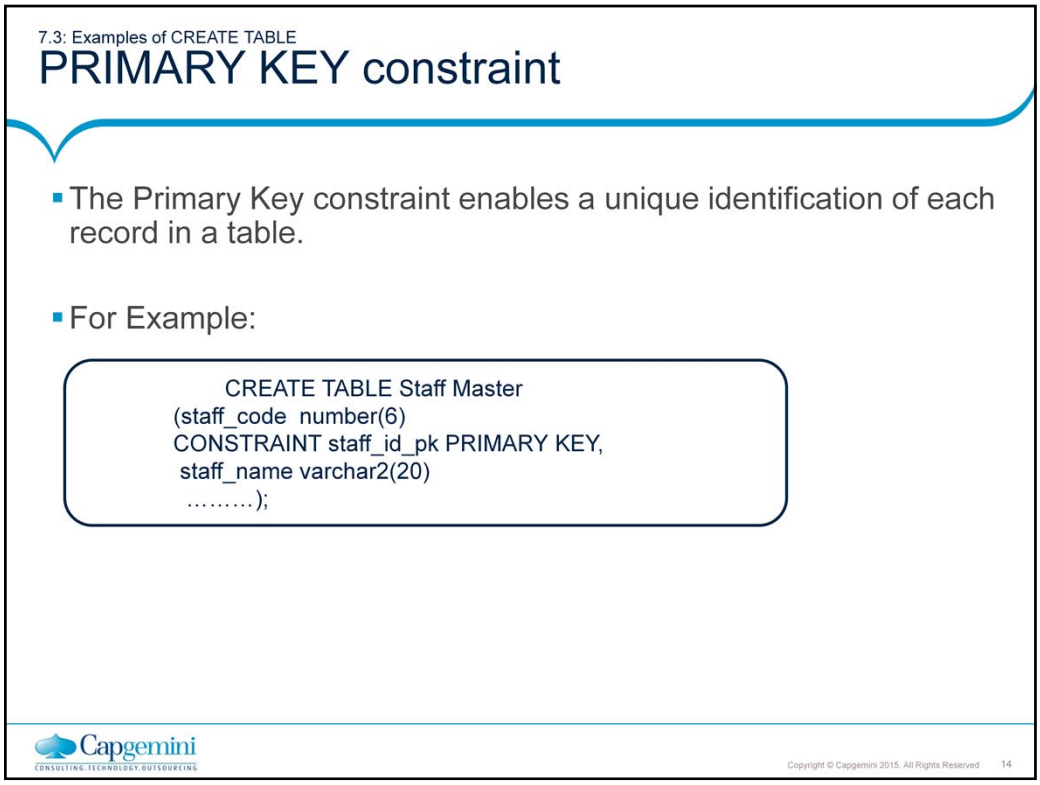
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12



UNIQUE constraint:

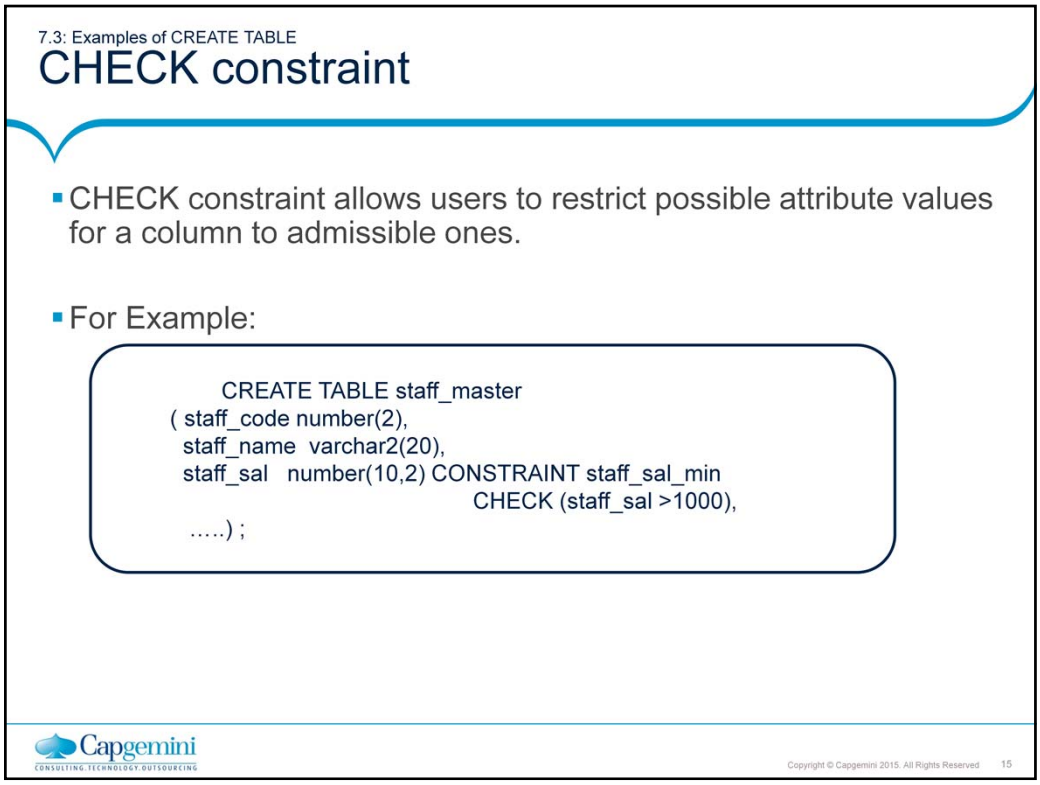
- The UNIQUE constraint does not allow duplicate values in a column.
  - If the UNIQUE constraint encompasses two or more columns, then two equal combinations are not allowed.
  - However, if a column is not explicitly defined as NOT NULL, then NULLS can be inserted multiple number of times.
- A UNIQUE constraint can be extended over multiple columns.
- A “unique value” rule defined on a column (or set of columns) allows inserting or updating a row only if it contains a “unique value” in that column (or set of columns).



PRIMARY KEY constraint:

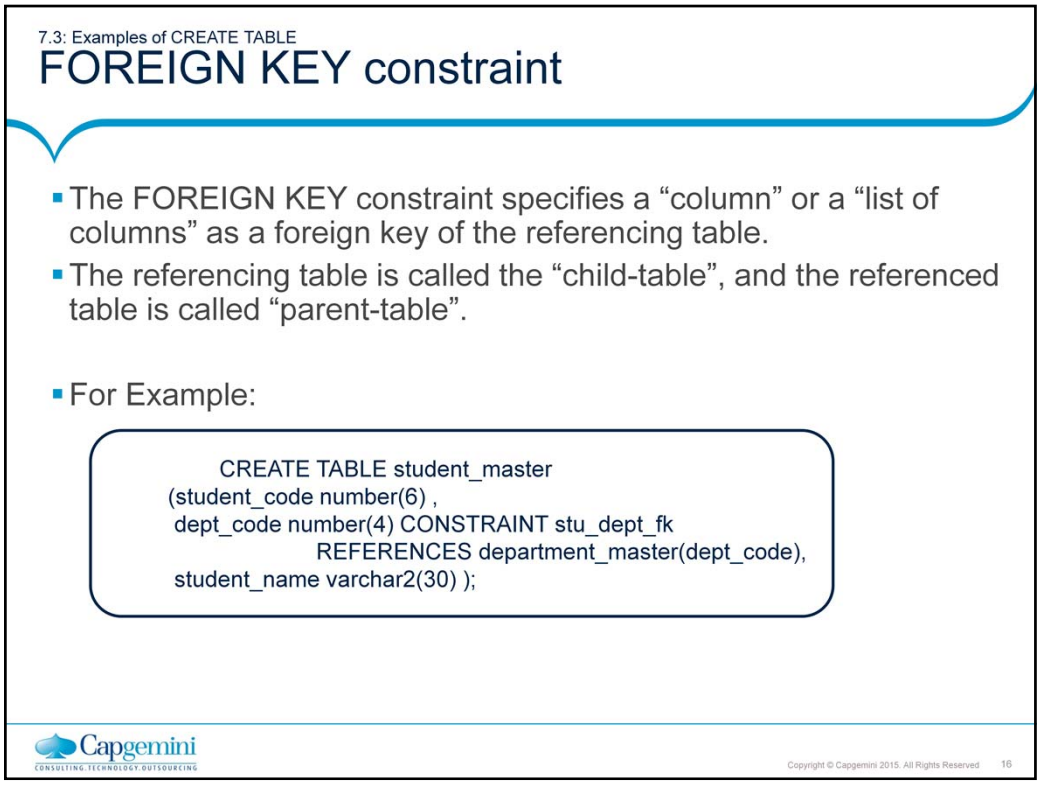
- On a technical level, a PRIMARY KEY combines a UNIQUE constraint and a NOT NULL constraint.
- Additionally, a table can have at the most one PRIMARY KEY.
- After creating a PRIMARY KEY, it can be referenced by a FOREIGN KEY.
- A “primary key” value rule defined on a key (a column or set of columns) specifies that “each row in the table can be uniquely identified by the values in the key”.
- The example on the slide defines the primary key constraint at the column level. The same example is seen below with the constraint defined at table level

```
CREATE TABLE Staff Master
(staff_code  number(6) ,
 staff_name varchar2(20),
  ……
CONSTRAINT staff_id_pk  PRIMARY KEY
(staff_code))
;
```



CHECK constraint:

- A CHECK constraint allows to state a minimum requirement for the value in a column.
- If more complicated requirements are desired, an INSERT trigger must be used.



FOREIGN KEY Constraint Keywords:

- Given below are a few Foreign Key constraint keywords:
  - FOREIGN KEY: Defines the column in the child table at the table constraint level.
  - REFERENCES: Identifies the table and column in the parent table.
  - ON DELETE CASCADE: Deletes the dependent rows in the child table when a row in the parent table is deleted.
  - ON DELETE SET NULL: Converts dependent FOREIGN KEY values to NULL.
- You can query the USER_CONSTRAINTS table to view all constraint definitions and names.
- You can view the columns associated with the constraint names in the USER_CONS_COLUMNS view.

- In EMP table, for deptno column, if we want to allow only those values that already exist in deptno column of the DEPT table, we must enforce what is known as REFERENTIAL INTEGRITY. To enforce REFERENTIAL INTEGRITY, declare deptno field of DEPT table as PRIMARY KEY, and deptno field of EMP table as FOREIGN KEY as follows

FOREIGN KEY Constraint Keywords:

- In the given example, FOREIGN KEY has been declared as a Table constraint.

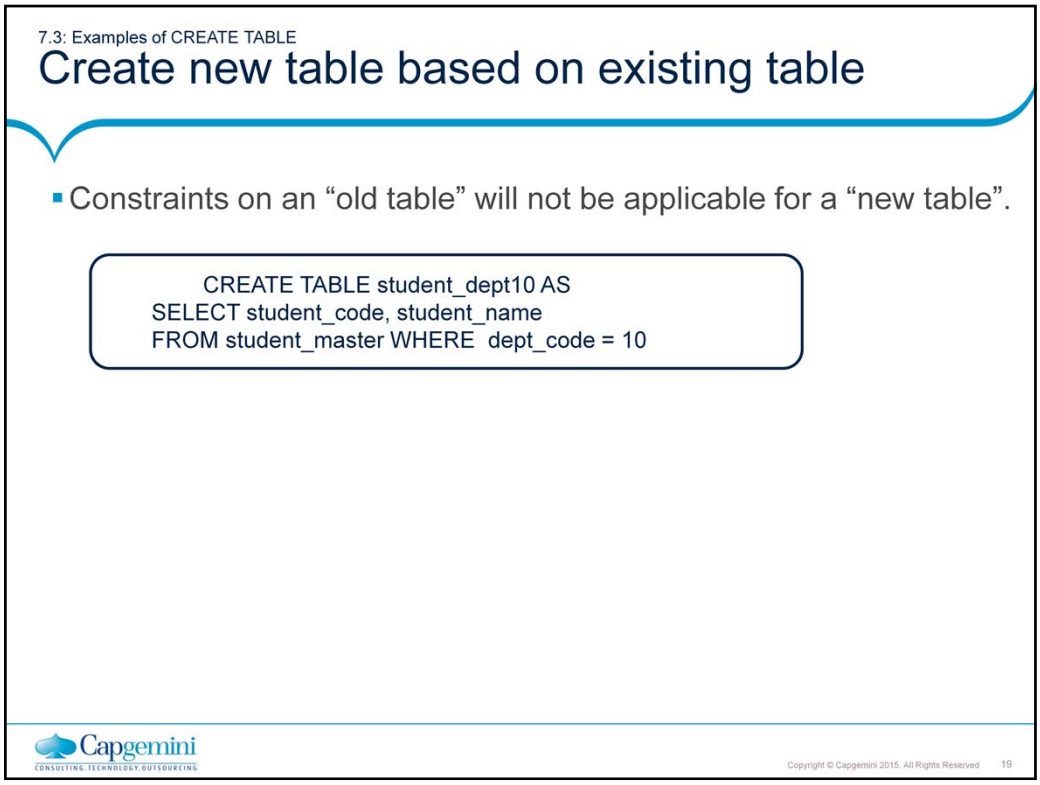
```
CREATE TABLE Dept
(
Deptno      NUMBER(2) CONSTRAINT
DEPTNO_P_KEY PRIMARY KEY,
Dname       VARCHAR2(14)    NOT NULL,
Loc         VARCHAR2(13)    NOT NULL
    );
CREATE TABLE Emp
    (
    Empno NUMBER(4) CONSTRAINT P_KEY
PRIMARY KEY,
    Ename VARCHAR2(10)  CONSTRAINT
ENAME_NOT_NULL  NOT NULL,
    Deptno NUMBER(2),
Job CHAR(9) CONSTRAINT JOB_ALL_UPPER
CHECK (Job =UPPER(Job)),
    Hiredate DATE DEFAULT SYSDATE,
    CONSTRAINT DEPTNO_F_KEY FOREIGN KEY
(Deptno)
  REFERENCES Dept(Deptno)
    );
```

Creation of database objects: Tables (contd.):

- A table can have a maximum of 1000 columns. Only one column of type LONG is allowed per table.

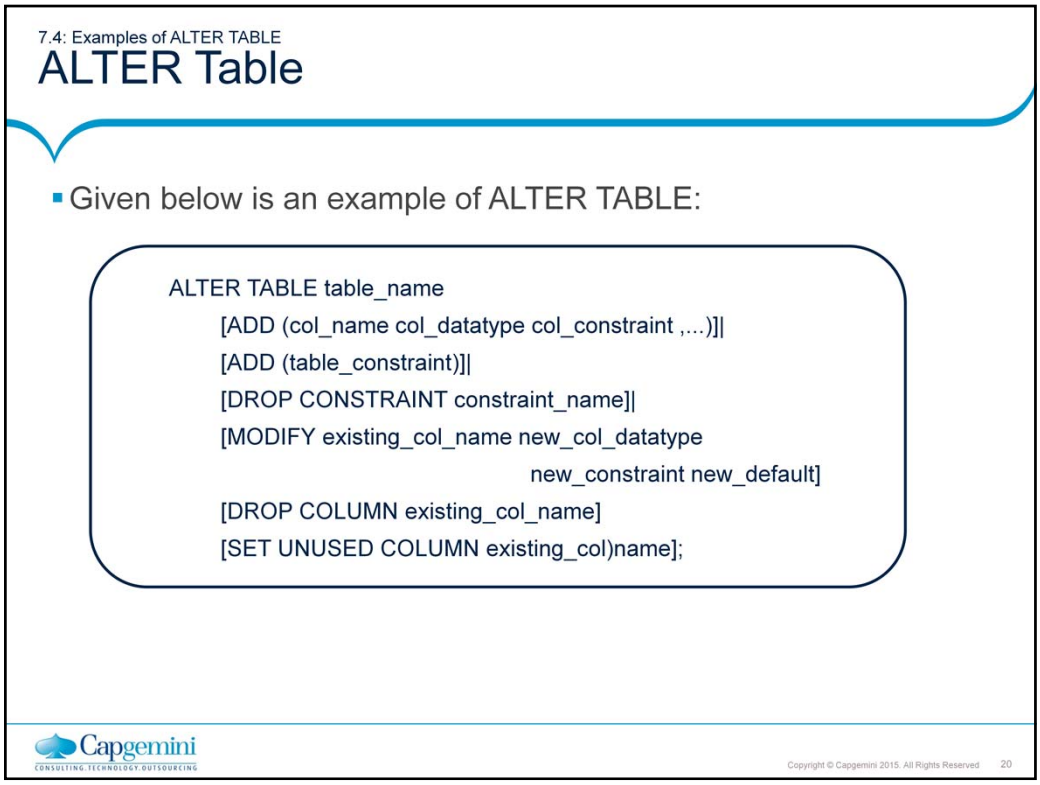
Table_name, col_name, const_name	A string upto 30 characters length. Can be made up to A-Z,0-9,$,_,# Must begin with a non-numeric ORACLE data characters.
Col_datatype	One of the previously mentioned types.
Col_constraint	A restriction on the column can be of following types: PRIMARY KEY, NOT NULL, UNIQUE, FOREIGN KEY, CHECK, Can be named. Only one PRIMARY KEY is allowed per table.
Table_constraint	A restriction on single or multiple columns. The types are same as in col_constraint. (NULL constraint is not allowed here).
AS query	Query is an SQL statement using SELECT command. SELECT command returns the rows from tables. It is useful if the table being created is based on an existing table. New table need not have all the columns of the old table. Names of columns can be different. All or part of the data can be copied.

- Any object created by a user is accessible to the user and the DBA only.
- To make the object accessible to other users, the creator or the DBA must explicitly give permission to others.



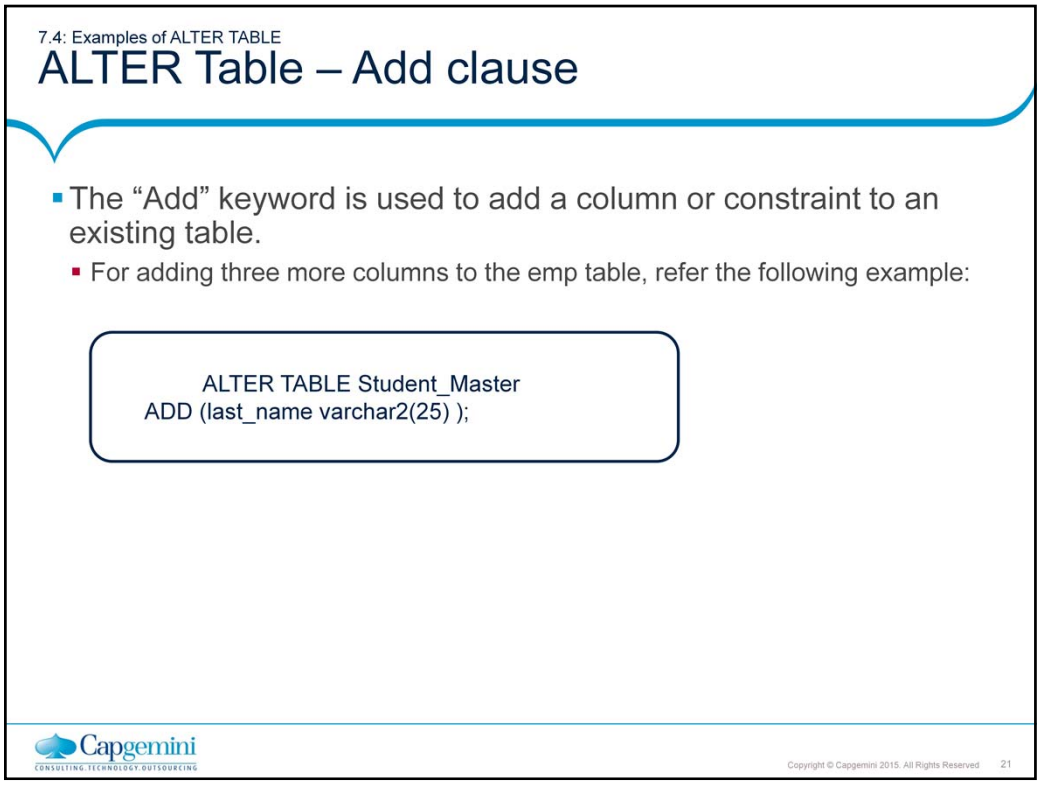
Creating new table based on an existing table:

- The example in the slide shows how to create a new table based on an existing table.
  - When the new table will be created, it will contain student information from department 10.
  - Constraints on an old table will not be applicable for the new table except NOT NULL constraint.



Examples of ALTER TABLE:

- Table_name must be an existing table.
- A column can be removed from an existing table by using ALTER TABLE.
- The uses of modifying columns are:
  - Can increase the width of a character column, any time.
  - Can increase the number of digits in a number, any time.
  - Can increase or decrease the number of decimal places in a number column, any time. Any reduction on precision and scale can be on empty columns only.
  - Can add only NOT NULL constraint by using Column constraints. All other constraints have to be specified as Table constraints.



ALTER TABLE – Add clause:

- The ADD clause allows to add a column or constraint. You can also add multiple columns in one statement separated by comma.
- A column with constraints can also be added as shown in the following example:

```
ALTER TABLE Department_Master
ADD (dept_name varchar2(10) NOT NULL);
```

7.4: Examples of ALTER TABLE

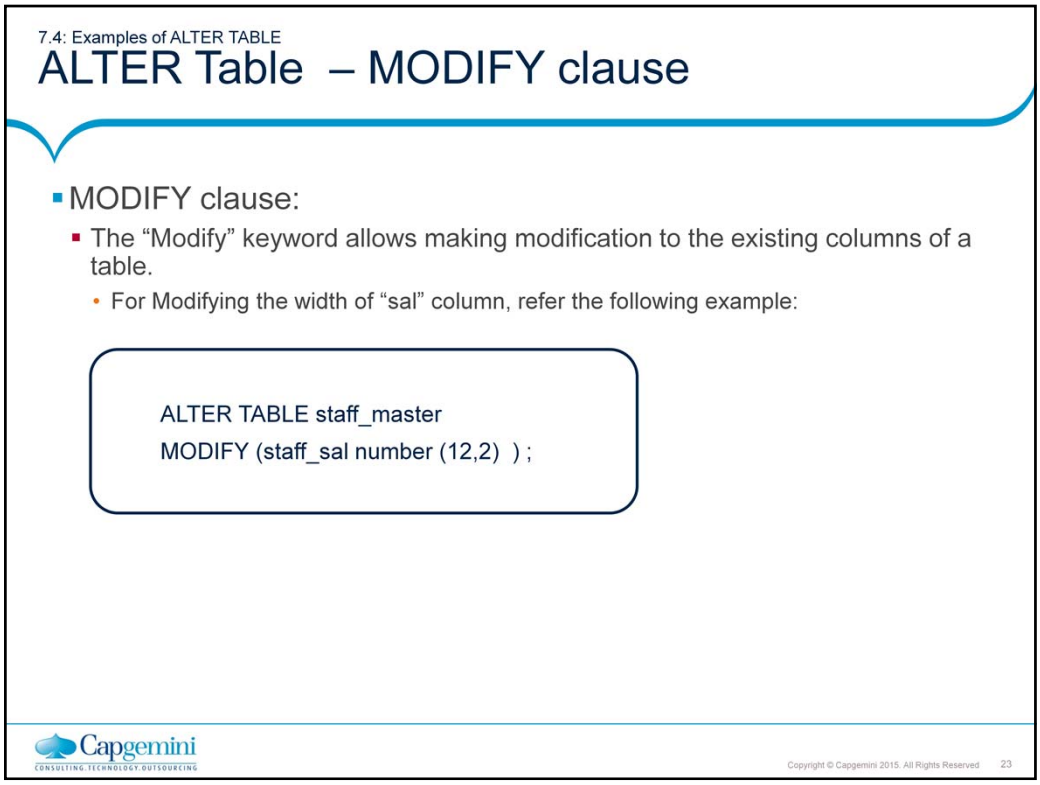
ALTER Table – Add clause

- For adding Referential Integrity on “mgr_code" column, refer the following example:

```
ALTER TABLE staff_master
  ADD CONSTRAINT FK FOREIGN KEY  (mgr_code) REFERENCES
staff_master(staff_code);
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22



ALTER TABLE- Modify Clause

The use of modifying the columns with the Enable | Disable clause are:

- Can increase "column width" of a character any time.
- Can increase the "number of digits" in a number at any time.
- Can increase or decrease the "number of decimal places" in a number column at any time. Any reduction on "precision" and "scale" can only be on empty columns.
- Can only add the NOT NULL constraint by using "column constraints". Rest all other constraints have to be specified as "table constraints".

7.4: Examples of ALTER TABLE

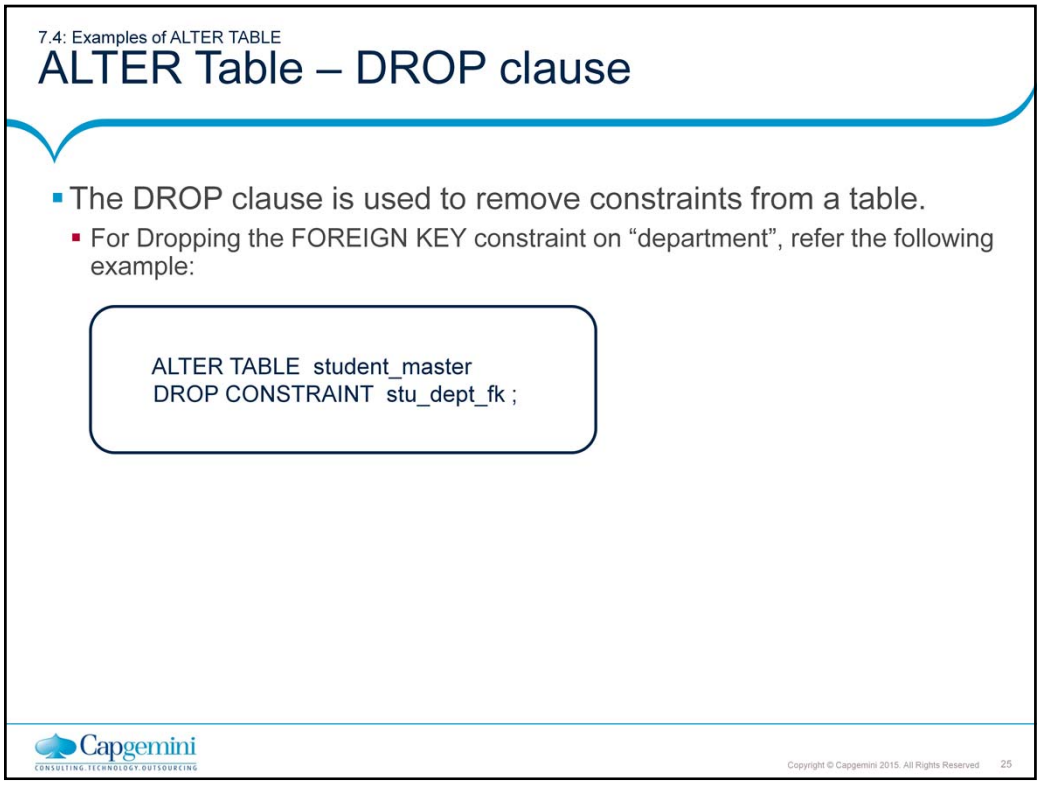
ALTER Table – Enable | Disable clause

- ENABLE | DISABLE Clause:
  - The ENABLE | DISABLE clause allows constraints to be enabled or disabled according to the user choice without removing them from a table.
  - Refer the following example:

```
ALTER TABLE staff_master DISABLE CONSTRAINT
SYS_C000934;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

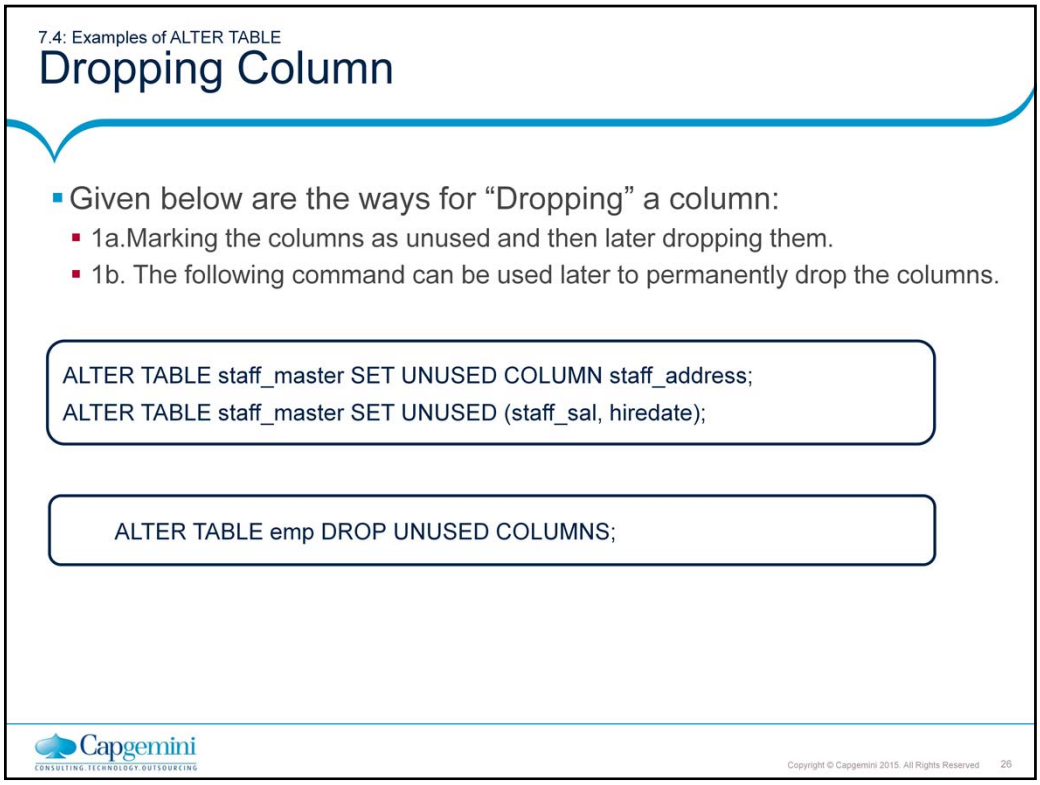
 24



ALTER TABLE – Drop Clause

- To remove the PRIMARY KEY constraint on the DEPARTMENT table and drop the associated FOREIGN KEY constraint on the DEPT_CODE column.

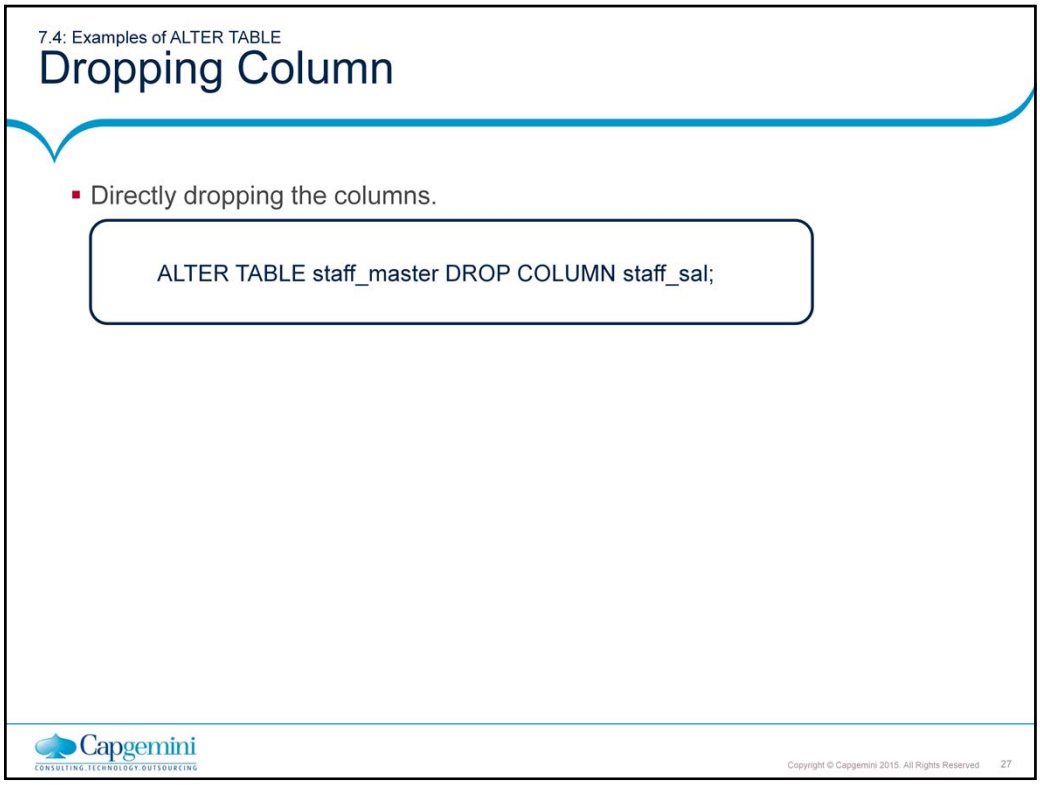
```
ALTER TABLE department_master
DROP PRIMARY KEY CASCADE;
```



Ways for "Dropping" a column:

1. Marking the columns as "Unused" and then later dropping them:

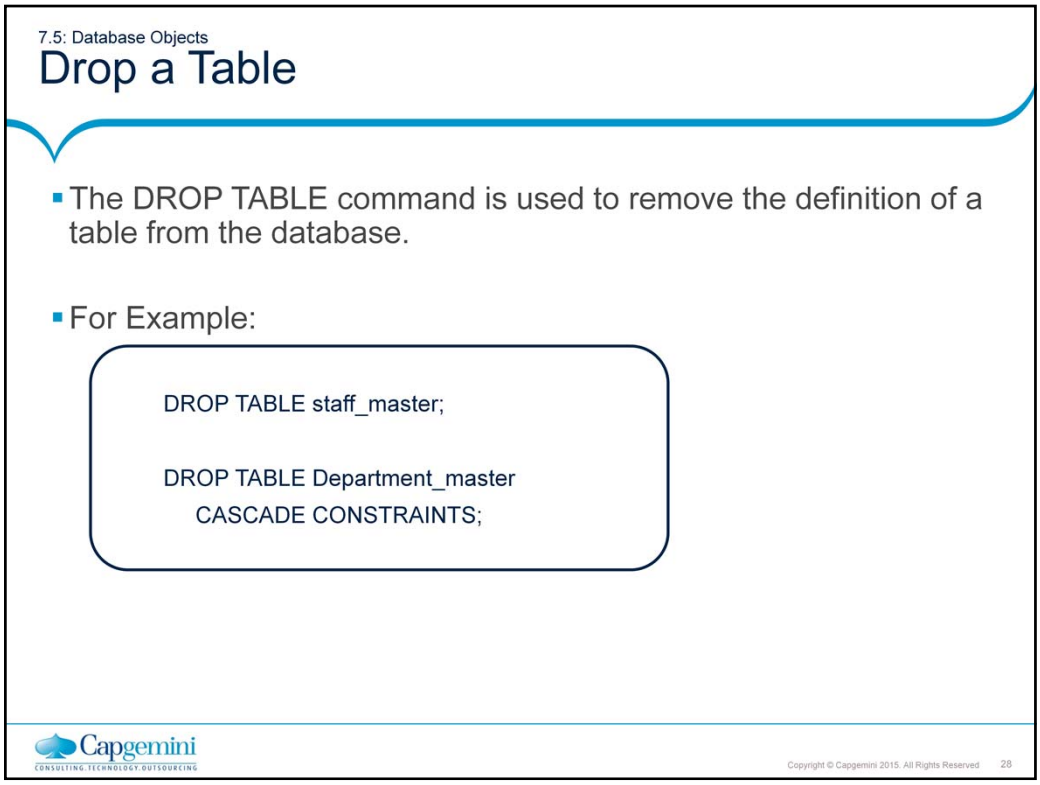
- Oracle onwards a new feature to "drop" and "set" the "unused columns" in a table is added
  - First command as shown in the slide, allows you to mark a column as unused and
  - Second command as shown in the following slide, lets you drop the unused column from the table to create more free space.
    - This feature drops the column from a table and releases any space back to the segment.
- **Note that:**
  - Columns once marked as unused cannot be recovered.
  - Marking the columns as unused does not release the space occupied by them back to the database.
    - Until you actually drop these columns, they continue to count towards the absolute limit of 1000 columns per table.
    - If you mark a column of data type LONG as UNUSED, you cannot add another LONG column to the table until you actually drop the unused LONG column.
- The advantage of the marking column as "unused" and then dropping them is that marking the columns is much faster process than dropping the columns.
- You can refer to the data dictionary table USER_UNUSED_COL_TABS to get information regarding the tables with columns marked as unused.



Ways for "Dropping" a column (contd.):

2. DROP COLUMN:

- This feature allows directly dropping the column.
- For DROP COLUMN command shown in the slide, to work successfully, the table should be exclusively locked by the user by giving the command.
  - All "indexes" defined on any of the target columns are also dropped.
  - All "constraints" that reference a target column are removed.
- Note that this command should be used with caution.
- The CASCADE CONSTRAINTS clause is used along with the DROP COLUMN clause.
- The CASCADE CONSTRAINTS clause drops all referential integrity constraints that refer to the primary and unique keys defined on the dropped columns.
- The CASCADE CONSTRAINTS clause also drops all multicolumn constraints defined on the dropped columns



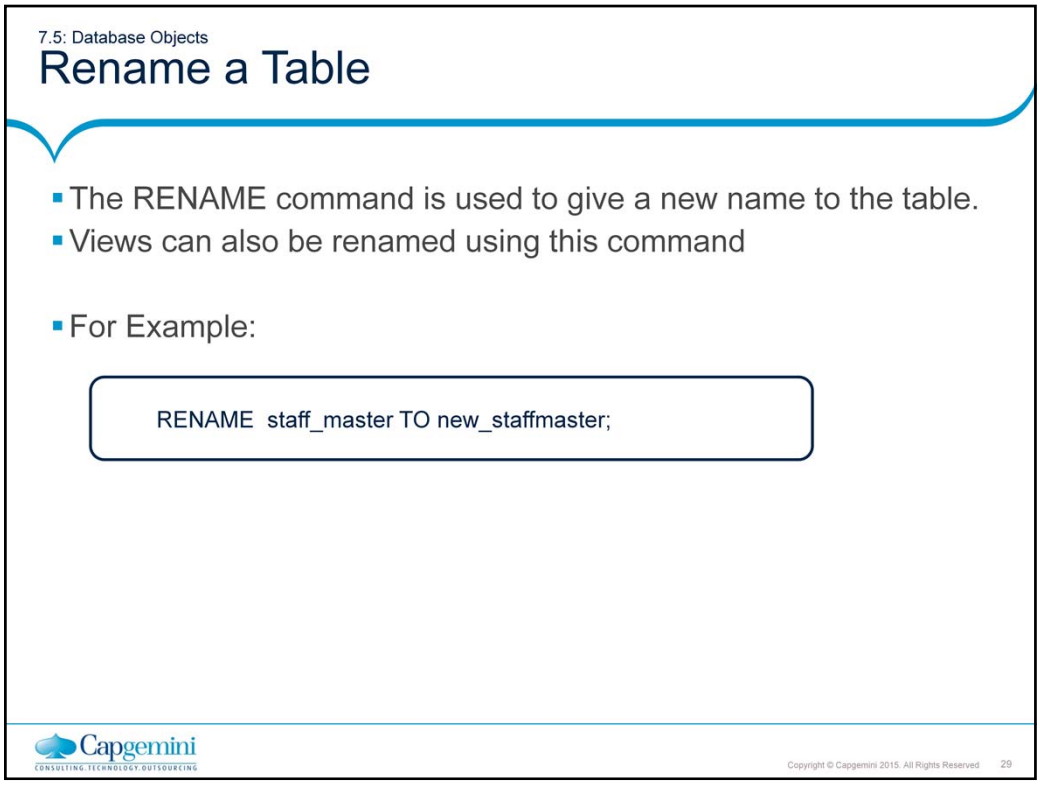
Deleting Database Objects: Tables

- Deleting objects that exist in the database is an easy task. Just say:

```
DROP Obj_Type obj_name;
```

- A table that is dropped cannot be recovered. When a table is dropped, dependent objects such as indexes are automatically dropped. Synonyms and views created on the table remain, but give an error if they are referenced.
- You cannot delete a table that is being referenced by another table. To do so use the following:

```
DROP  TABLE table-name CASCADE CONSTRAINTS;
```

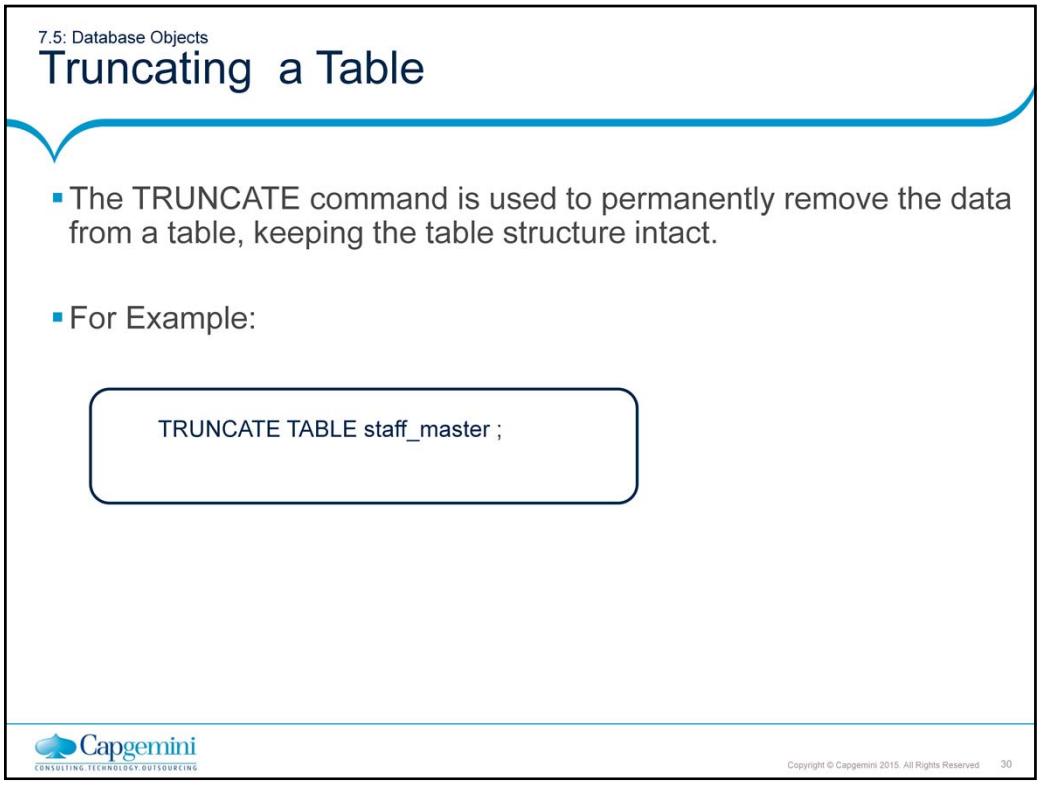


Renaming Database Objects: Tables

- Renaming objects that exist in the database is an easy task. Just say:

```
RENAME Existing_TableName TO new_tablename;
```

- You can RENAME a table that is being referenced by another table.



Truncating Database Objects: Tables

- Truncating objects that exist in the database is an easy task. Just say:

```
TRUNCATE TABLE  Existing_TableName
```

7.5: Database Objects

User_Tables & User_Objects

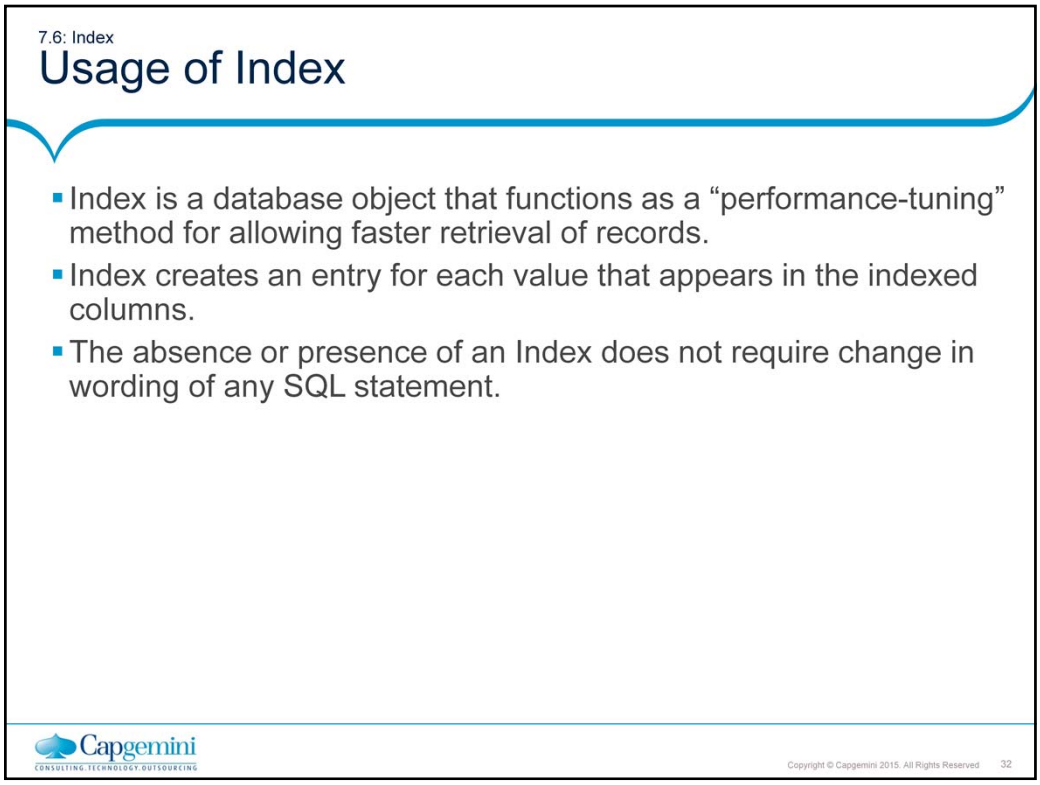
- To view the names of tables owned by the user, use the following query:
- To view distinct object types owned by the user, use the following query:

```
SELECT table_name
FROM user_tables
```

```
SELECT DISTINCT object_type
FROM user_objects ;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31



Index

• An index

- Is a schema object

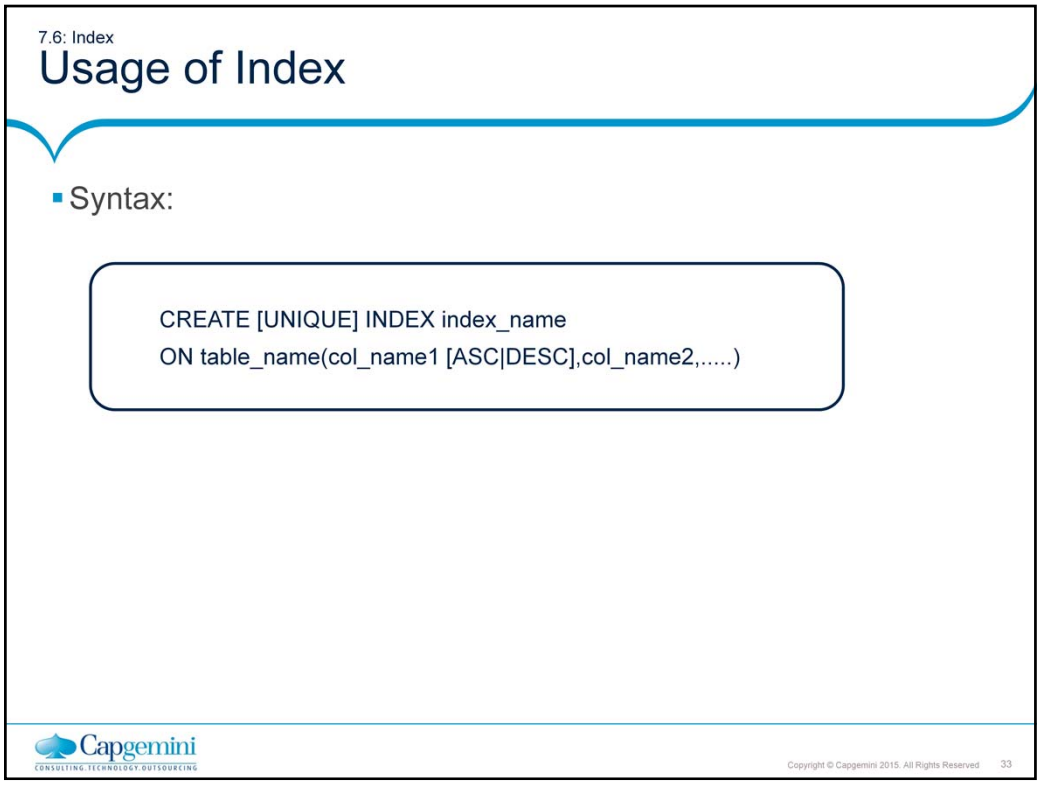
- Is used by the Oracle server to speed up the retrieval of rows by using a pointer

- Can reduce disk I/O by using a rapid path access method to locate data quickly

- Is independent of the table it indexes

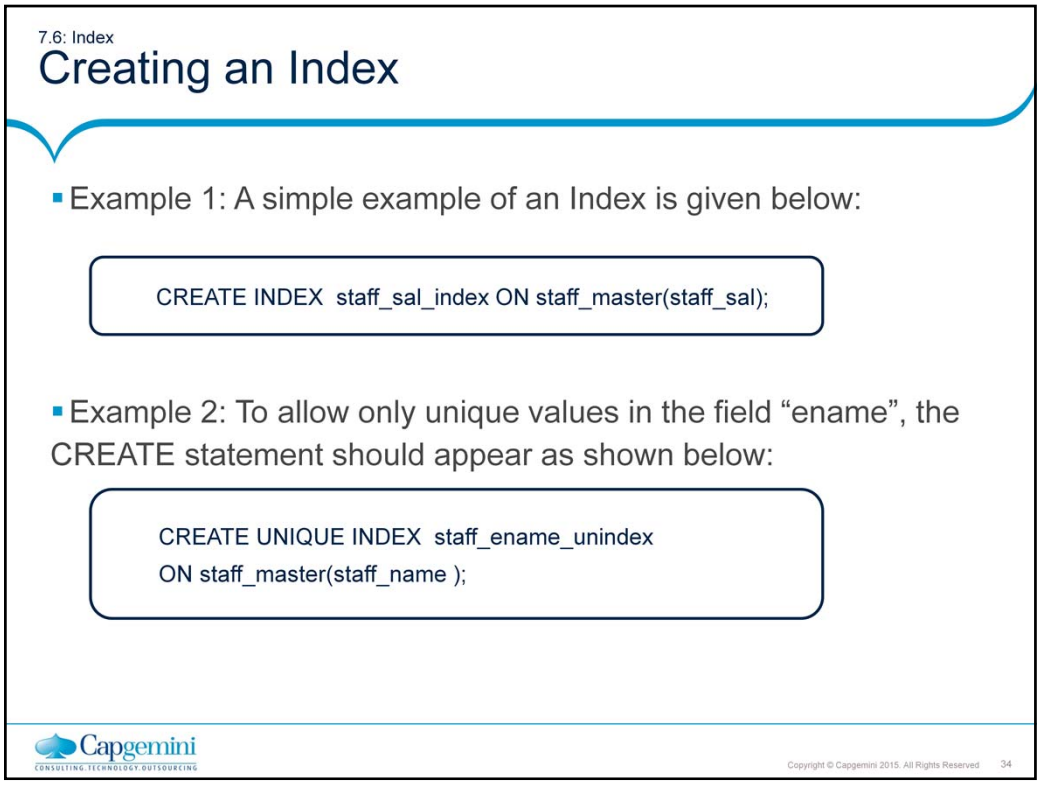
- Is used and maintained automatically by the Oracle Server.

- Oracle uses indexes to avoid the need for large-table, full-table scans and disk sorts, which are required when the SQL optimizer cannot find an efficient way to service the SQL query



Index:

- An index creates an entry for each value that appears in the indexed columns (by default, Oracle creates B-tree indexes).
- The absence or presence of an Index does not require change in wording of any SQL statement.
  - An Index is merely a fast access path to the data.
  - An Index only affects the speed of execution.
- There are different types of Indexes, which can be created by the user.
- A “Table” can have any number of “Indexes”.
- An Index can be on a single column or on multiple columns.
  - In Oracle, up to 32 columns can be included in one Index.
- Updation of all Indexes is handled by RDBMS.
- UNIQUE ensures that all rows in a Table are unique.
- An Index is in an ascending order, by default.
- The RDBMS decides when to use the Index while accessing the data.
- Removal of an Index does not affect the Table on which it is based.
- When you create a PRIMARY or a UNIQUE constraint on the “Table”, a UNIQUE index is automatically created.
  - The name of the constraint is used for the Index name.

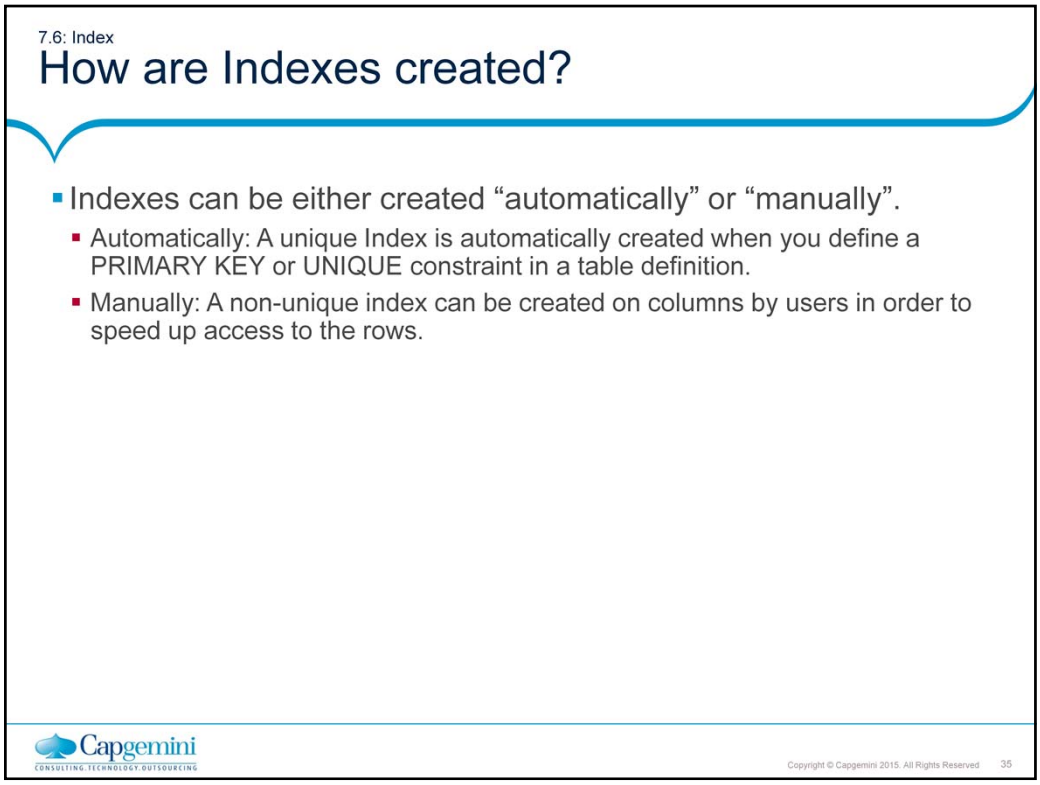


Note:

- More Indexes on a Table does not mean faster queries.
  - Each DML operation that is committed on a Table with Indexes imply that the Indexes must be updated.
  - The more number of Indexes are associated with a Table, the more effort the Oracle server must make to update all the Indexes after a DML operation.

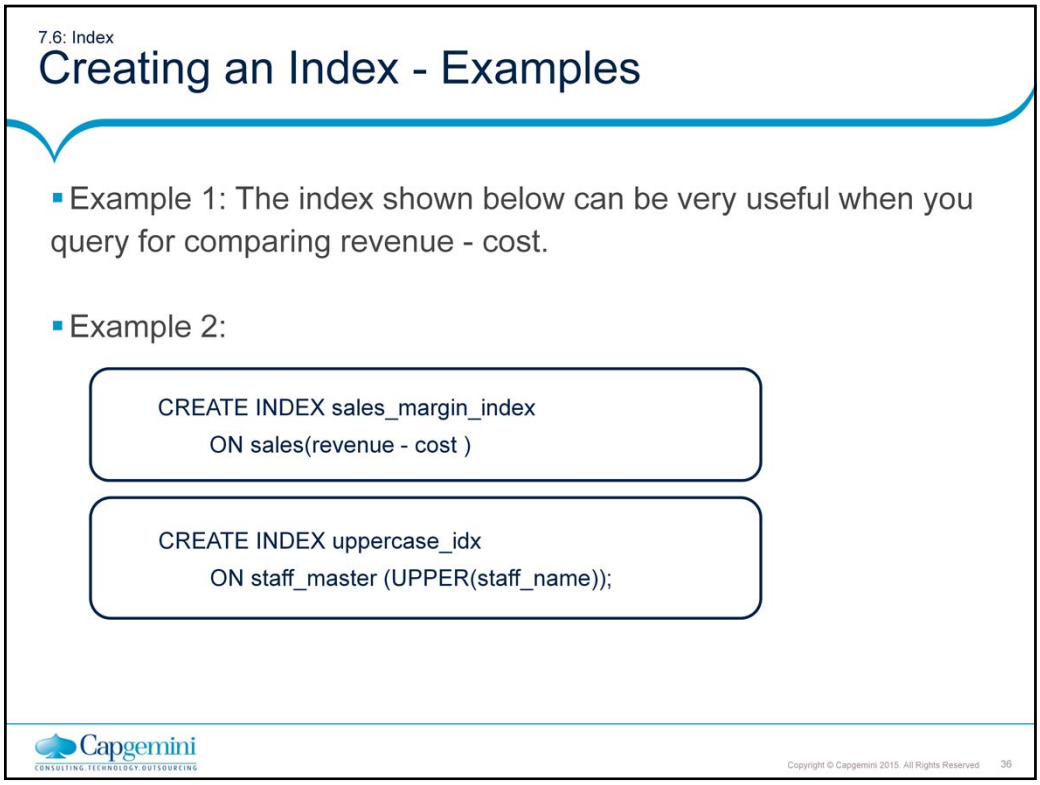
When do you create an Index?

- You should create Indexes, only if:
  - the column contains a wide range of values
  - the column contains a large number of NULL values
  - one or more columns are frequently used together in a WHERE clause or join condition
  - the table is large and most queries are expected to retrieve less than 2–4% of the rows
- If you want to enforce uniqueness, you should define a UNIQUE constraint in the Table definition. Then a “unique index” will be automatically created.
- It is usually not worth creating an Index, if:
  - The Table is small.
  - The columns are not often used as a condition in the query.
  - Most queries are expected to retrieve more than 2 to 4 percent of the rows in the table.
  - The Table is frequently updated.
  - The indexed columns are referenced as part of an expression.



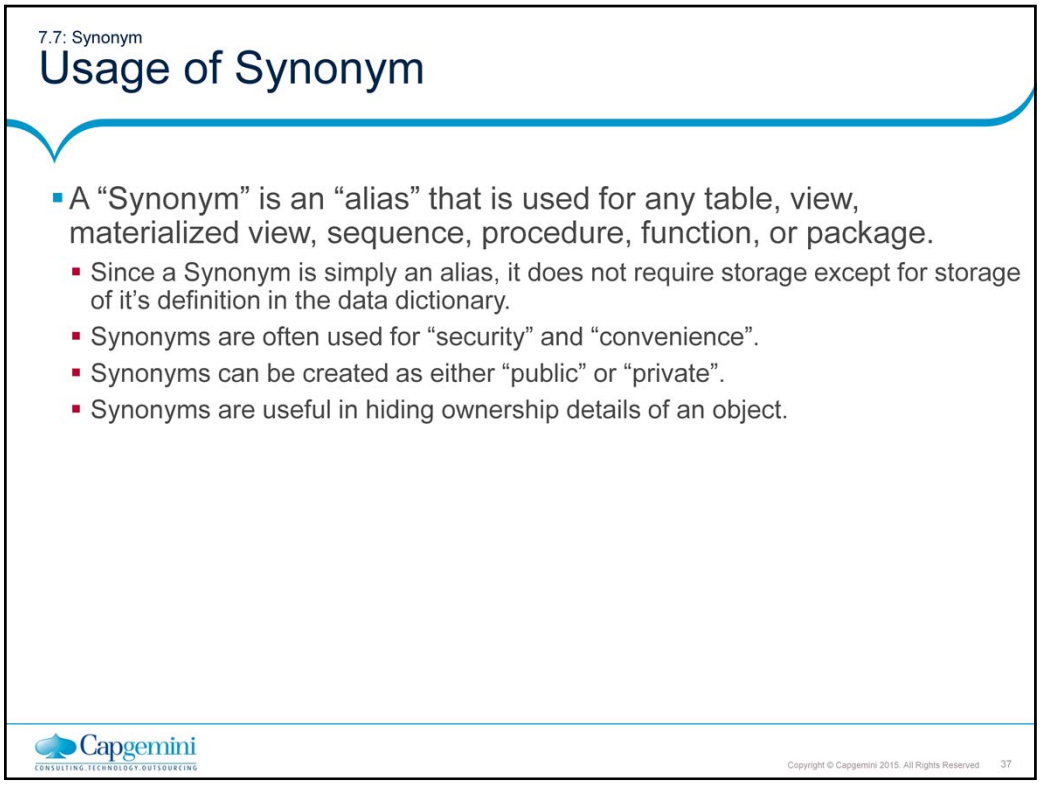
Types of Indexes:

- Two types of Indexes can be created.
  - One type of index is an “unique index”. The Oracle server automatically creates this index when you define a column in a table that has a PRIMARY KEY or a UNIQUE key constraint. The name of the Index is same as the name given to the constraint.
  - The other type of index is a “non-unique index”, which can be created by a user.
    For example: You can create a FOREIGN KEY column index for a “join” in a query in order to improve retrieval speed.
    Note: You can manually create a “unique index”. However, it is recommended that you create a “UNIQUE constraint”, which implicitly creates a “unique index”.



Note:

- The USER_INDEXES data dictionary view contains the name of the index and its uniqueness.
- The USER_IND_COLUMNS view contains the index name, the table name, and the column name.



Synonym:

- A Synonym simplifies access to objects. A Synonym is simply another name for an object.
- With Synonyms, you can:
  - Ease referring to a table owned by another user.
  - Shorten lengthy object names.

7.7: Synonym

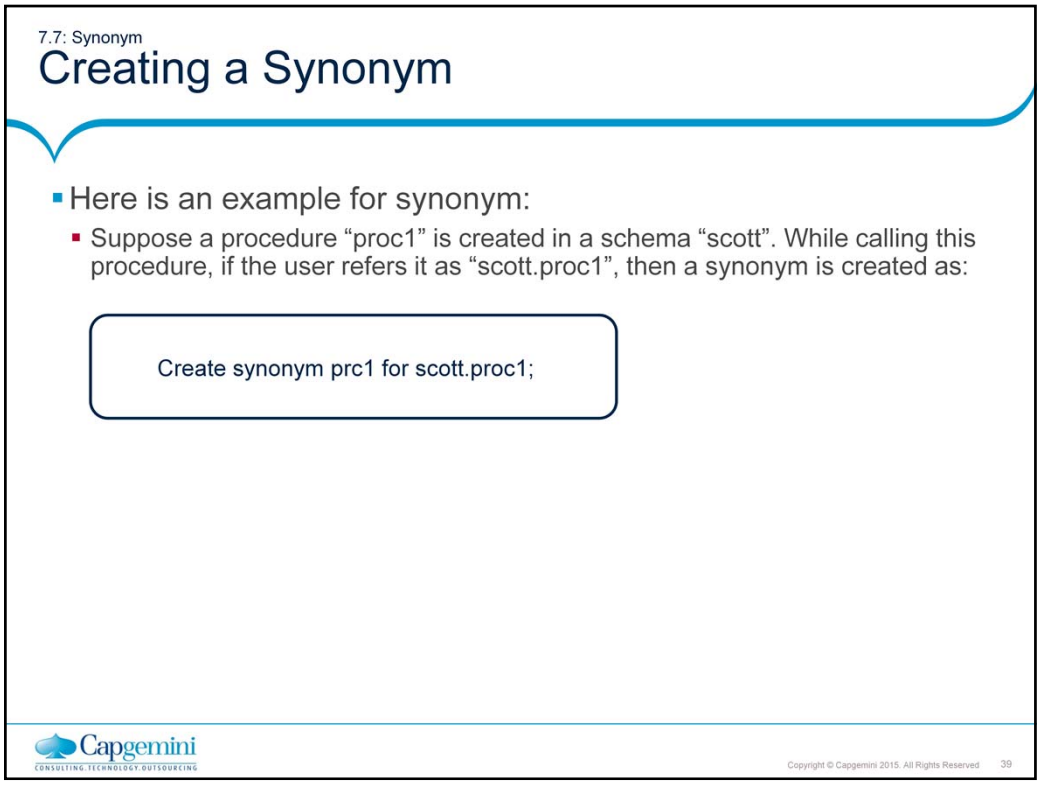
Usage of Synonym

- Syntax
  - where:
    - Existing_name is the name of a table, view, or sequence.
    - PUBLIC is used to grant permission to all users for accessing the object by using the new name. (This is done only by a DBA.)

```
CREATE [PUBLIC] SYNONYM another_name  FOR existing_name
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 38



Creating and Removing Synonyms:

- You can create a shortened name for the SALGRADE table as shown in the following example:

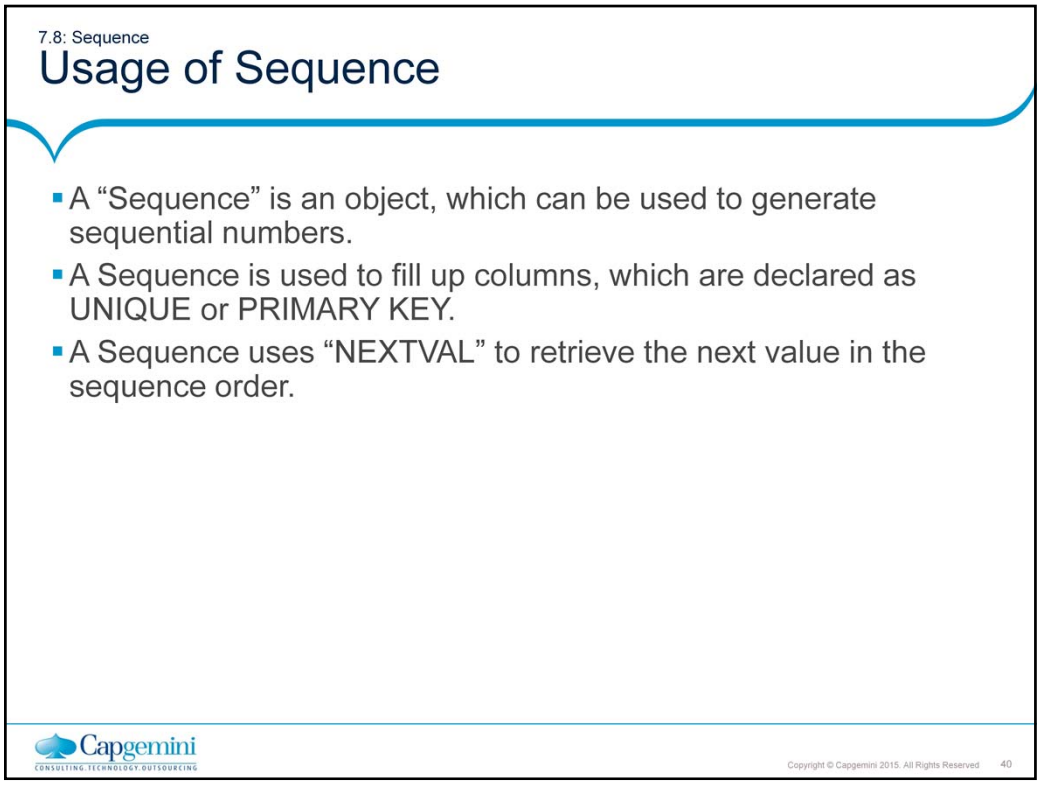
```
CREATE SYNONYM  s
FOR salgrade
Synonym Created.
```

- You can create a shortened name for the DEPT_SUM_VU view as shown in the following example:

```
CREATE SYNONYM  d_sum
FOR  dept_sum_vu;
Synonym Created.
```

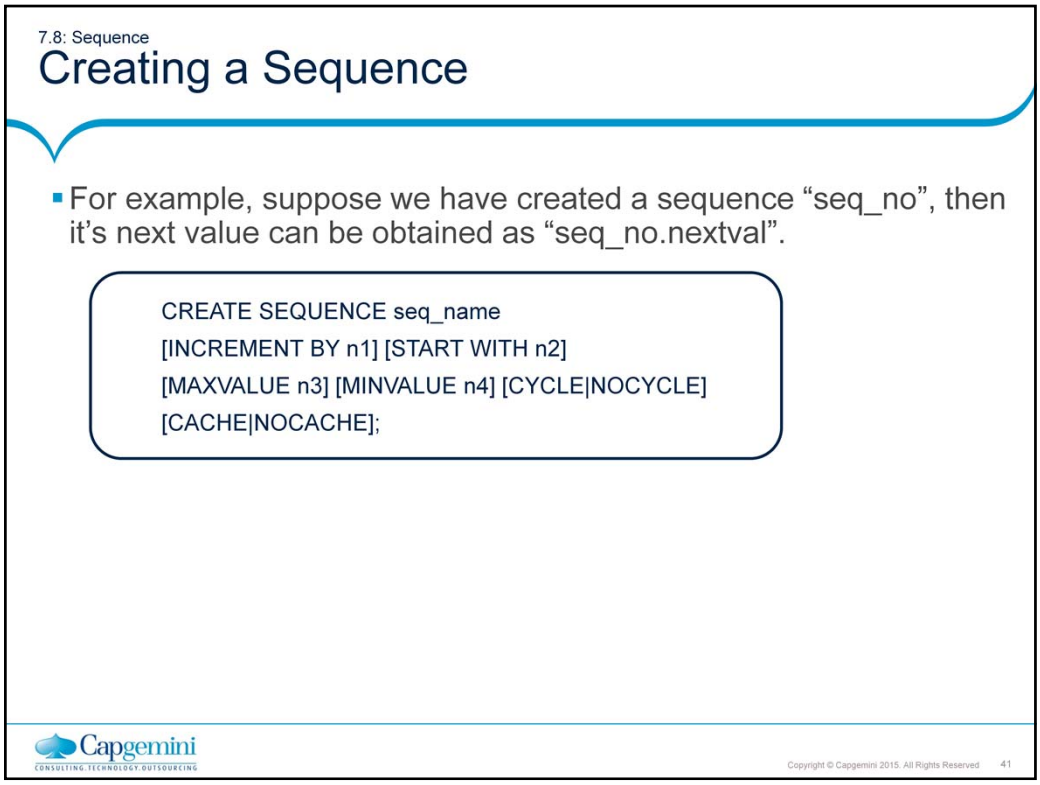
- You can drop a Synonym as shown in the following example:

```
DROP SYNONYM d_sum;
Synonym dropped.
```



Sequence:

- A Sequence:
  - automatically generates unique numbers.
  - is a “sharable object”.
  - is typically used to create a PRIMARY KEY value.
  - replaces application code.
  - speeds up the efficiency of accessing sequence values when cached in memory.



Sequence:

- In the example shown in the slide:
  - START WITH indicates the first NUMBER in the series.
  - INCREMENT BY is the difference between consecutive numbers. If n1 is negative, then the numbers that are generated are in a descending order.
  - MAXVALUE and MINVALUE indicate the extreme values.
  - CYCLE indicates that once the extreme is reached, it starts the cycle again with n2.
  - NOCYCLE means that once the extreme is reached, it stops generating numbers.
  - CACHE caches the specified number of sequence values in the memory. This speeds access, but all cached numbers are lost when the database is shut down. The default value is 20

Confirming Sequences

- As shown below, you need to verify your sequence values in the USER_SEQUENCES data dictionary table.

```
SELECT sequence_name, min_value, max_value,
       increment_by, last_number
FROM user_sequences;
```

- The LAST_NUMBER column displays the next available sequence number if NOCACHE is specified.

7.8: Sequence

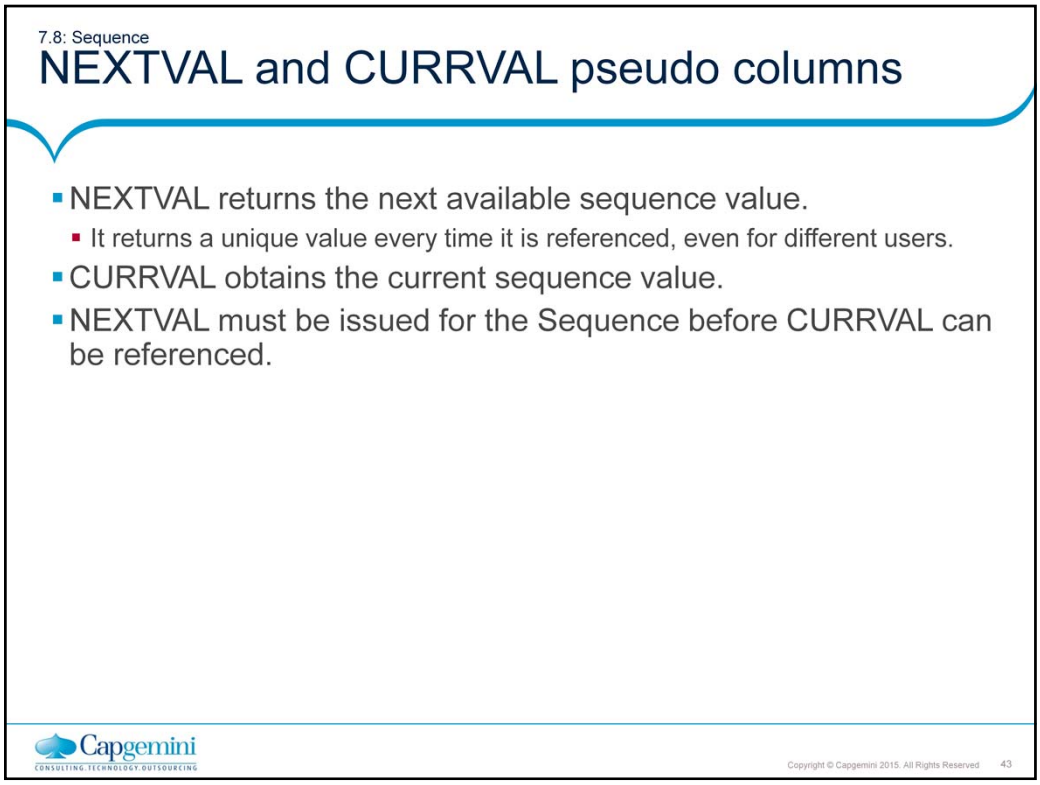
Creating a Sequence

- Here is one more example of sequence:
  - s1 will generate numbers 1,2,3....,10000, and then stop.

```
CREATE SEQUENCE  s1
                INCREMENT BY 1
                START WITH 1
                MAXVALUE 10000
                NOCYCLE ;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 42



Referencing a Sequence:

- After you create a Sequence, it generates sequential numbers that can be used in your tables. You can reference the Sequence values by using the NEXTVAL and CURRVAL pseudocolumns.
- **NEXTVAL and CURRVAL Pseudocolumns:**
  - The **NEXTVAL pseudocolumn** is used to extract successive sequence numbers from a specified Sequence. You must qualify NEXTVAL with the sequence name. When you reference sequence.NEXTVAL, a new sequence number is generated and the current sequence number is placed in CURRVAL.
  - The **CURRVAL pseudocolumn** is used to refer a Sequence number that the current user has just generated.
- NEXTVAL must be used to generate a sequence number in the session of the current user, before CURRVAL can be referenced.
  - You must qualify CURRVAL with the sequence name.
  - When “sequence.CURRVAL” is referenced, the last value returned to that user’s process is displayed.

- By using a sequence you can Insert a new employee with empno 1 as shown below:

```
INSERT into staff_master(staff_code,staff_name)
VALUES(S1.NEXTVAL,'XYZ')
```

- You can view the current value for the S1 sequence as shown below:

```
SELECT S1.CURRVAL FROM dual;
```

Usage of a Sequence:

1. Caching Sequence Values

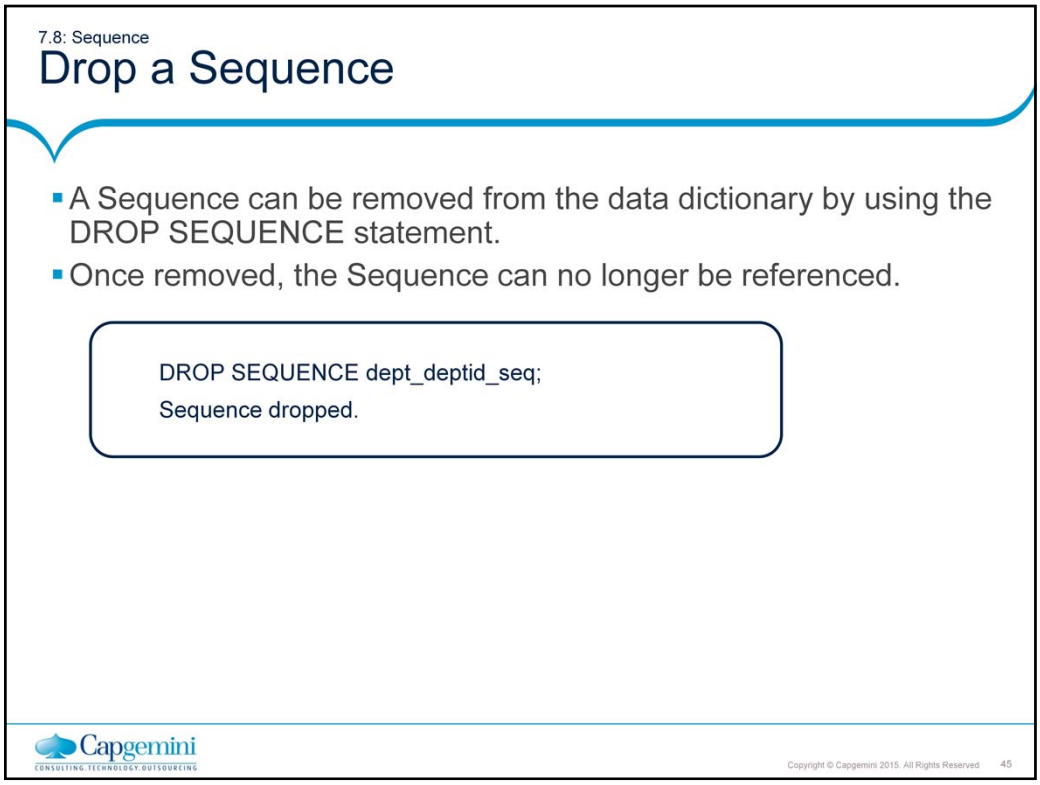
- Cache sequences in memory to provide faster access to those "sequence values".
  - The first time you refer the Sequence, the cache is populated.
  - Each request for the next Sequence value is retrieved from the cached sequence.
  - After the last sequence value is used, the next request for the Sequence pulls another cache of sequences into the memory.

Gaps in the Sequence

- Although "sequence generators" issue "sequential numbers" without gaps, this action occurs independent of a **commit** or **rollback**. Therefore, if you roll back a statement containing a Sequence, the number is lost.
- Another event that can cause gaps in the Sequence is a **system crash**. If the Sequence caches the values in memory, then those values are lost if there is a system crash.
- Since Sequences are not directly tied to tables, the same Sequence can be used for multiple tables. If you do so, each table can contain gaps in the Sequential numbers.

2. Viewing the Next Available Sequence Value without Incrementing It

- If the Sequence is created with NOCACHE, it is possible to view the next available Sequence value without incrementing it by querying the USER_SEQUENCES table.



Removing a Sequence

- To remove a sequence from the data dictionary, use the DROP SEQUENCE statement.
- To remove the Sequence, you must be the owner of the Sequence or possess the DROP ANY SEQUENCE privilege.

Syntax:

```
DROP  SEQUENCE  sequence;
```

where: sequence is the name of the sequence generator.

- For more information, refer Oracle9i SQL Reference, “DROP SEQUENCE”.

contd.

Altering a Sequence

- To alter a Sequence use the following syntax:

```
ALTER SEQUENCE s1
INCREMENT BY n1
MAXVALUE n3
CYCLE;
```

  - s1 is an existing Sequence.

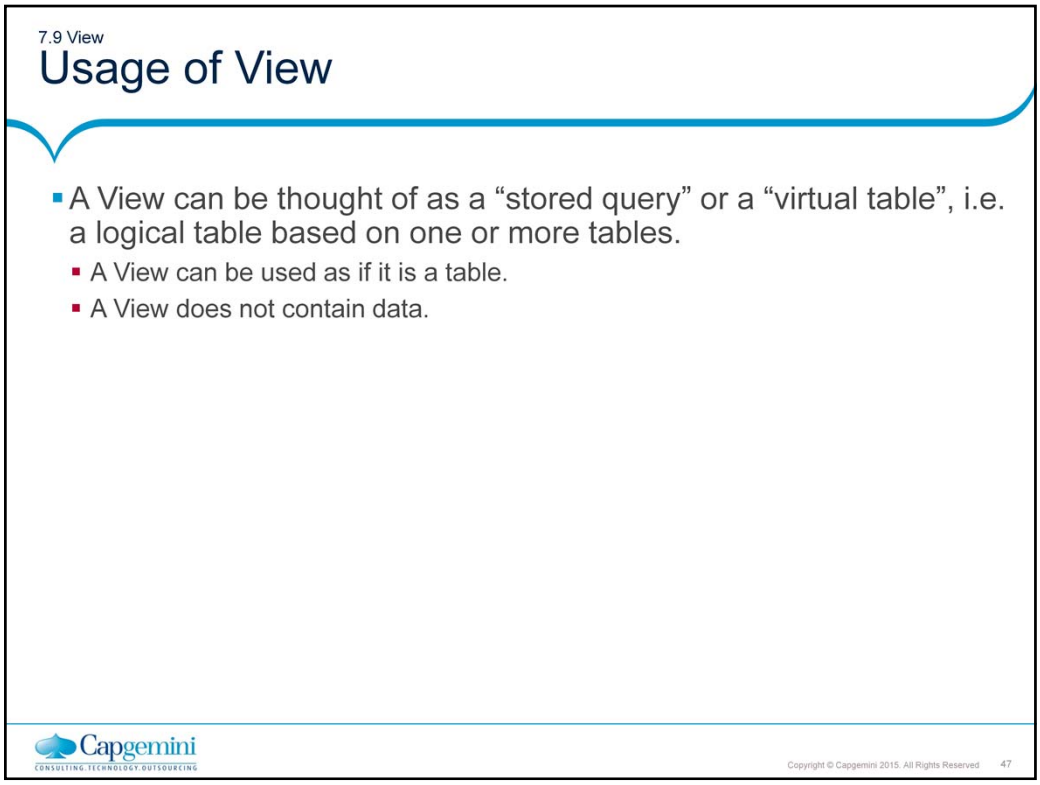
- To make the Sequence start from a new number, drop the Sequence and create it again.
- ALTER SEQUENCE has no effect on numbers that are already generated.

```
ALTER SEQUENCE s1
INCREMENT BY 5
MAXVALUE 25000
CYCLE;
```

Guidelines for modifying Sequences:

Given below are a few guidelines for modifying Sequences:

- You must be the owner or have the ALTER privilege for the Sequence.
- Only future sequence numbers are affected.
- The Sequence must be dropped, and re-created to restart the Sequence at a different number.
- Some validation should be performed.



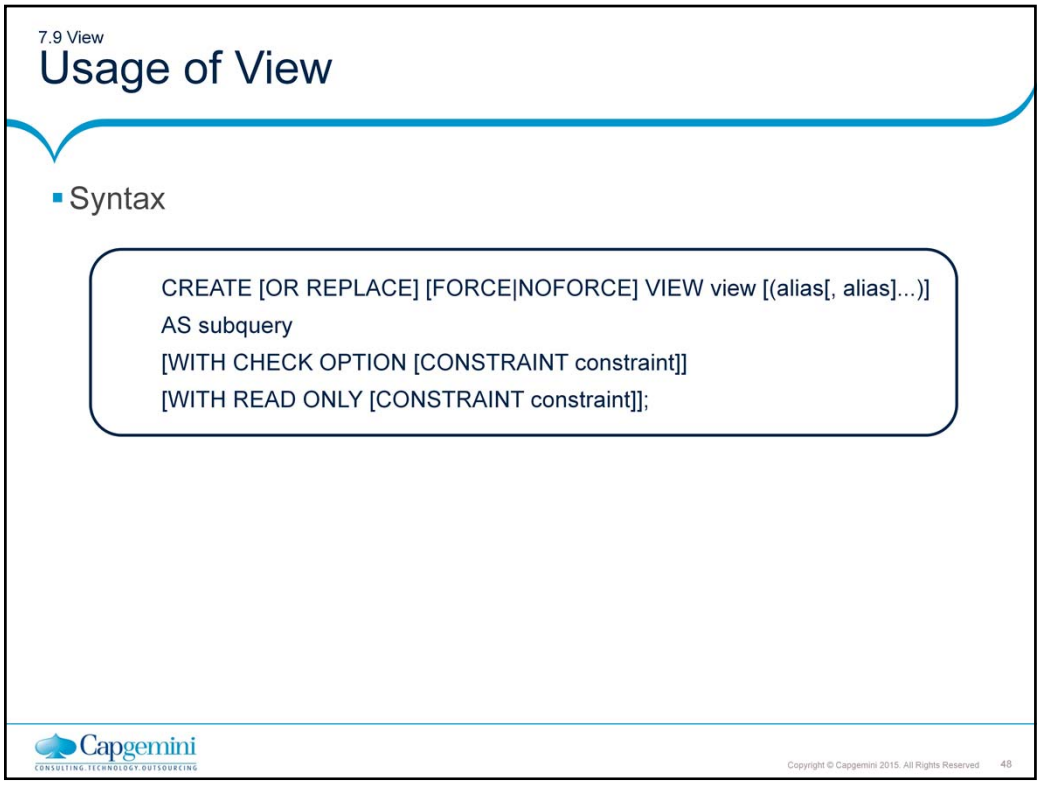
Why do we use Views?

Views are used:

- To restrict data access.
- To make complex queries easy.
- To provide data independence.
- To present different views of the same data.

Features of Simple and Complex Views:

Features	Simple View	Complex View
Number of tables	One	One or more
Contains functions	No	Yes
Contains groups of data	No	Yes
DML operations through a View.	Yes	Not always



View:

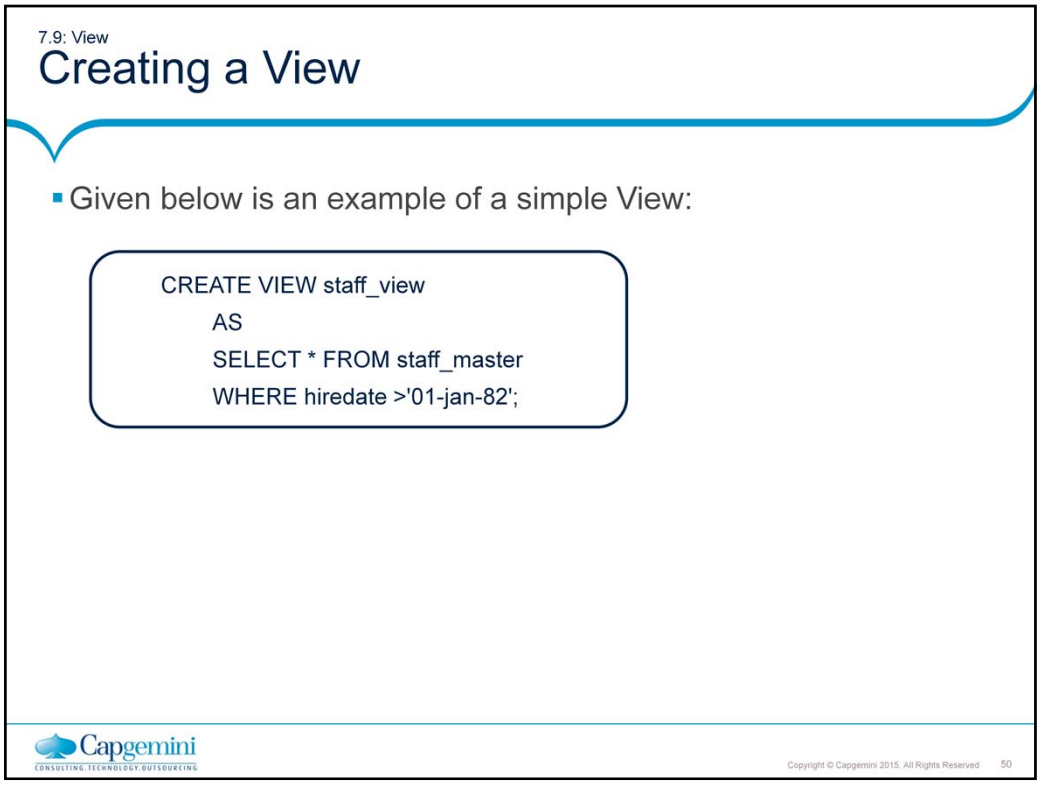
Creating a View:

- You can embed a sub-query within the CREATE VIEW statement.
- The sub-query can contain complex SELECT syntax.

Characteristics of a View:

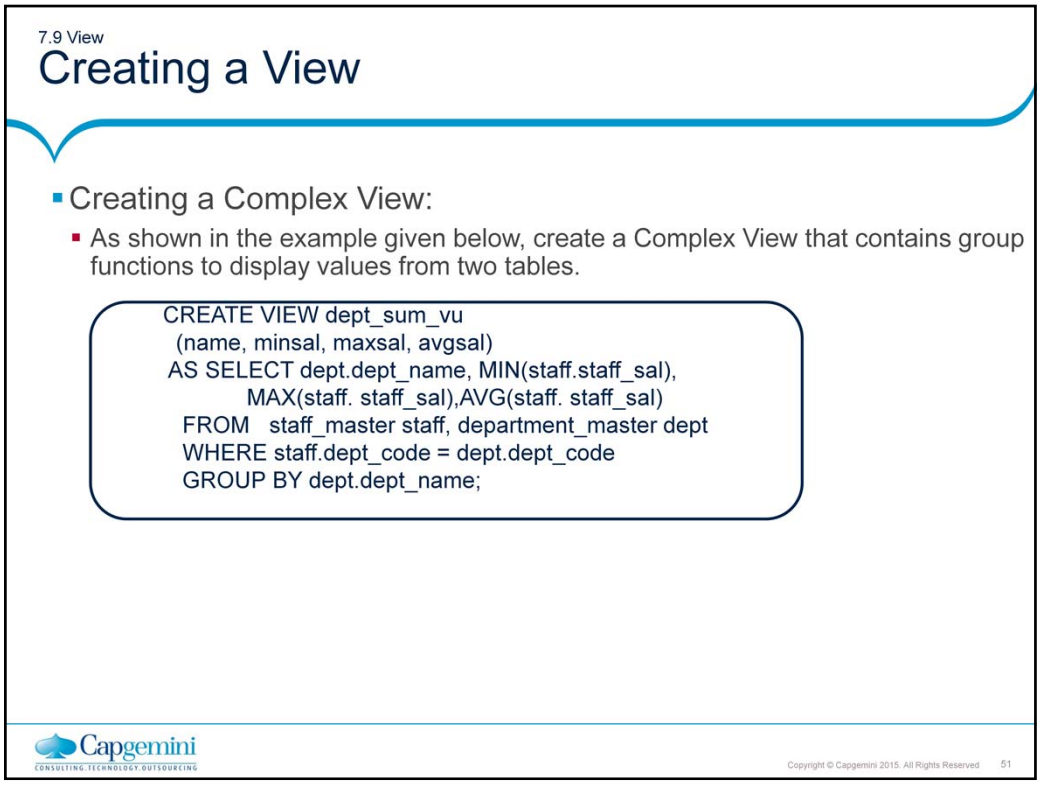
- View is a logical table that is based on one or more Tables.
- View can be used as if it is a Table.
- View does not contain data.
- Whenever a View is accessed, the query is evaluated. Thus a View is dynamic.
- Any changes made in the View affects the Tables on which the View is based.
- View helps to hide the following from the user:
  - Ownership details of a Table, and
  - Complexity of the query used to retrieve the data
- "With check option" implies you cannot insert a row in the underlying Table, which cannot be selected by using the View.
- If you use a ORDER BY clause in the View, then it automatically becomes a “Read-only View”.

Understanding the syntax
•OR REPLACE - re-creates the view if it already exists
•FORCE - creates the view regardless of whether or not the base tables exist
•NOFORCE - creates the view only if the base tables exist. This is default
•View - is the name of the view
•Alias - specifies names for the expressions selected by the view's query
•Subquery - is a complete and valid SELECT statement
•WITH CHECK OPTION - specifies that only rows accessible to the view can be inserted or updated
•Constraint - is the name assigned to the CHECK OPTION
•WITH READ ONLY - ensures that no DML operations can be performed on this view



Restrictions while using Views:

- Updating / Inserting rows in the base table **is possible** by using Views, however, with some restrictions. This is possible only if the View is based on a single table.
- The restrictions are :
  - Updation / Insertion not possible if View is based on two tables. However, this can be done in ORACLE 8 onwards.
  - Insertion is not allowed if the underlying table has any NOT NULL columns, which are not included in the View.
  - Insertion / Updation is not allowed if any column of the View referenced in UPDATE / INSERT contains “functions” or “calculations”.
  - Insertion / Updation / Deletion is not allowed if View contains GROUP BY or DISTINCT clauses in the query.

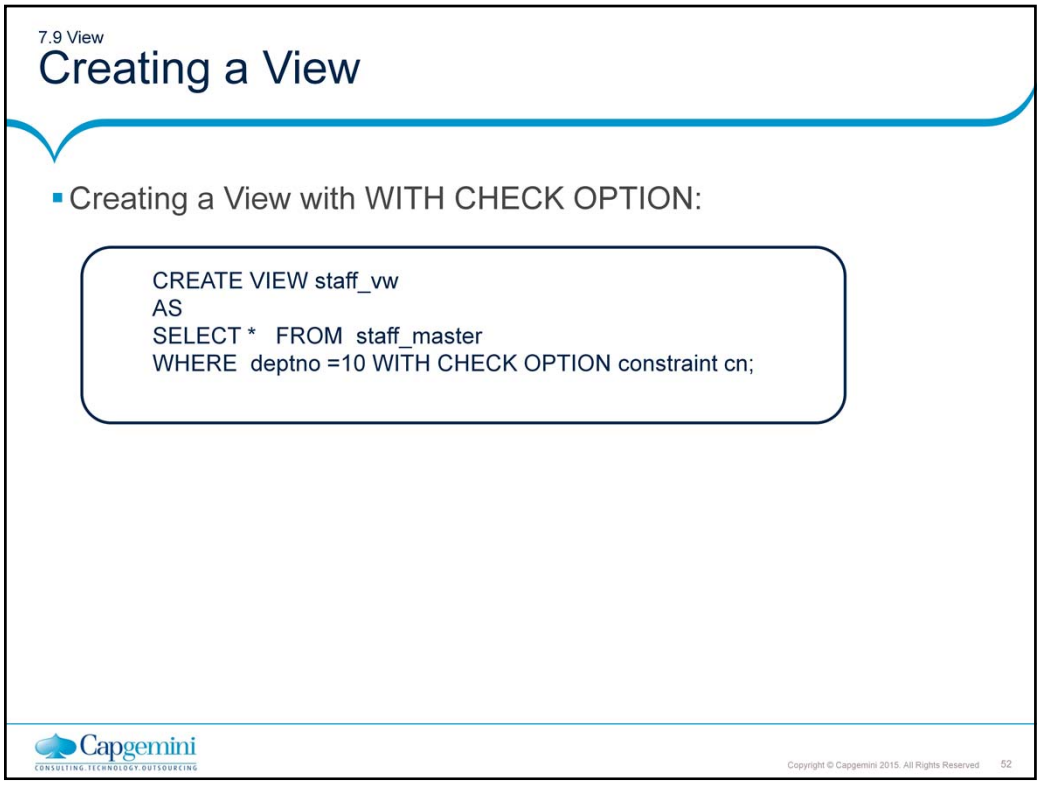


Modifying a View:

- As shown in the example given below, modify the View "staff_vu" by using the CREATE OR REPLACE VIEW clause. Add an alias for each column name.

```
CREATE OR REPLACE VIEW staff_vu
  (id_number, name, sal, department_id)
AS SELECT staff_code,staff_name,staff_sal,dept_code
   FROM  staff_master
   WHERE   dept_code = 80;
```

- Column aliases in the CREATE VIEW clause are listed in the same order as the columns in the sub-query.

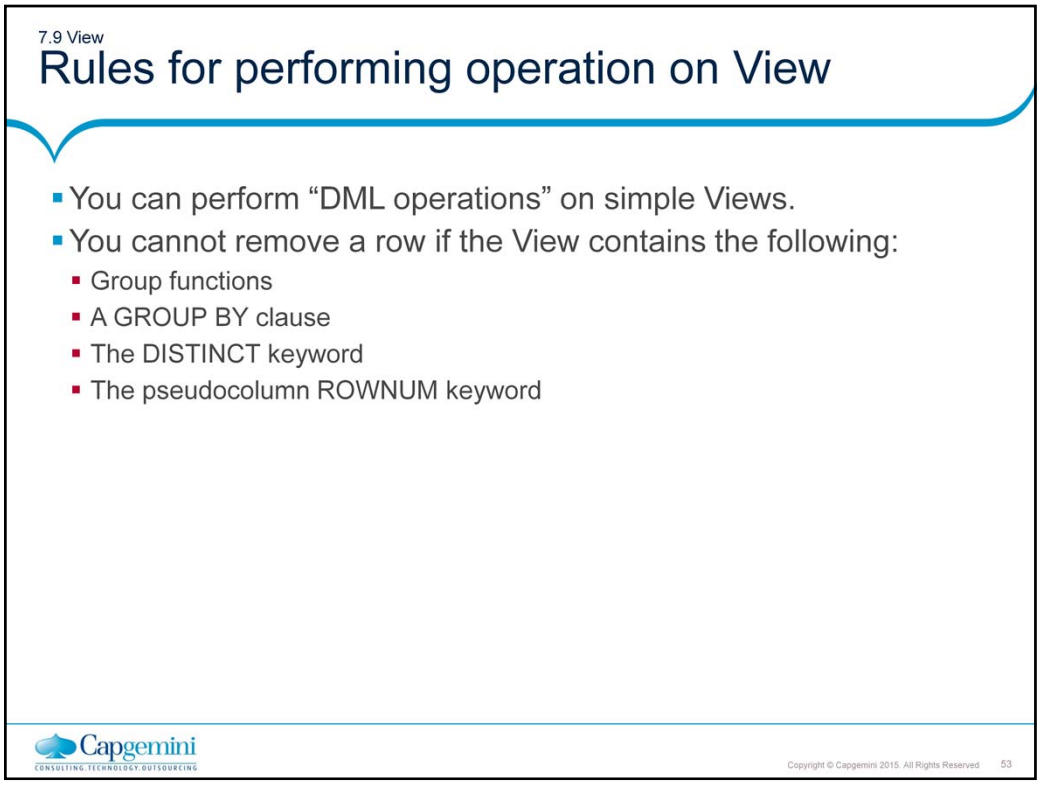


WITH CHECK OPTION Specifies that only the rows accessible to the view can be inserted or updated. Constraint "cn" is the name assigned to the CHECK OPTION constraint in the above example. Any attempt to change the department number for any row in the view fails because it violates the WITH CHECK OPTION constraint. A violation produces:
ORA-01402: view WITH CHECK OPTION where-clause violation
With the given error, the SQL tried to INSERT or UPDATE a record in a view that contained a WITH CHECK OPTION. The resulting INSERT or UPDATE violates the WHERE clause of the view.

Creating a Read-only View:

•If a View is based on a single table, the user may manipulate records in the View, and it's underlying base table.
•The WITH READ ONLY clause of the CREATE VIEW command can be used to prevent users from manipulating records in a View.

```
CREATE VIEW student_view
    AS
    SELECT * FROM student_master
    WHERE hiredate >'01-jan-82'  WITH  READ ONLY.
```



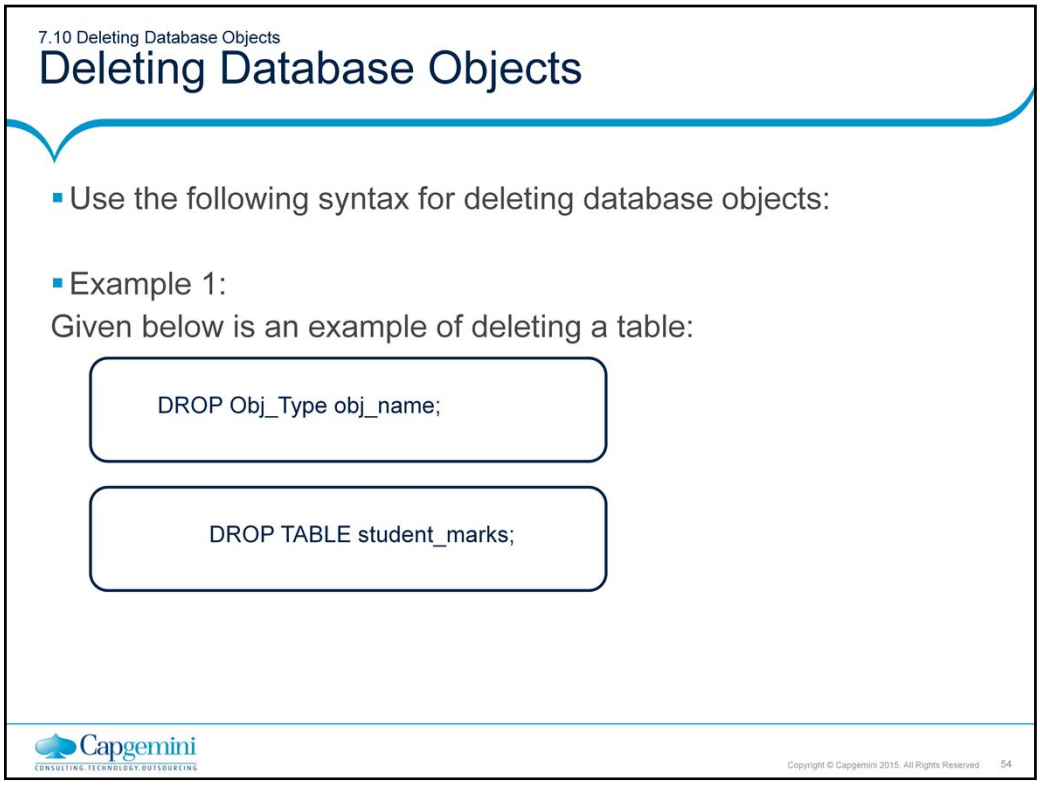
Rules for performing DML Operations on a View:

- You can perform “DML operations” on data “through a View” only if those operations follow certain rules.
  - You can remove a row from a view, unless it contains any of the following:
    - Group functions
    - A GROUP BY clause
    - The DISTINCT keyword
    - The pseudocolumn ROWNUM keyword

Removing a View:

- You can remove a View by using the following syntax:

```
DROP VIEW viewname;
```



Examples:

- A table, which is dropped, cannot be recovered.
- Dependent objects such as Indexes are automatically dropped.
- Synonyms and Views remain intact. However, they give an error when referenced.

7.10 Deleting Database Objects

Deleting a Database Objects

- Example 2:
- If new_emp is a Synonym for a table, then the Table is not affected in any way. Only the duplicate name is removed.

```
DROP SYNONYM new_emp;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 55

7.11: Tips and Tricks

Guidelines

- When creating tables based on subquery the number of specified columns if defined for the table should match to the number of columns in the subquery.
- Create an index if
  - A column contains a wide range of values
  - A column contains a large number of null values
  - One or more columns are frequently used together in a WHERE clause or a join condition
  - The table is large and most queries are expected to retrieve less than 2 to 4 percent of the rows

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 56

7.11: Tips and Tricks

Guidelines

- An Index is not very useful if :
  - The table is small
  - The columns are not often used as a condition in the query
  - Most queries are expected to retrieve more than 2 to 4 percent of rows in the table
  - The table is updated frequently
  - The indexed columns are referenced as part of an expression

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 57

Summary

- In this lesson, you have learnt:
  - What are Database Objects?
  - Basic Data Types
  - Data Integrity
  - Different types of Database Objects:
  - Modification of Database Objects
  - Deleting Database Objects

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 58

Review – Questions

- Question 1: Indexes can be created _____ or _____
- Question 2: _____ obtains the current sequence value
- Question 3: Synonyms can be created as either _____ or _____

Review – Questions

- Question 4: Gaps in sequence values can occur when there is a rollback
  - True / False
- Question 5: Synonyms are useful in hiding ownership details of an object.
  - True / False

DBMS/SQL

Lesson 08 Data Manipulation Language

Lesson Objectives

- To understand the following topics:
  - Adding Data
  - Removing Data
  - Modifying Data

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

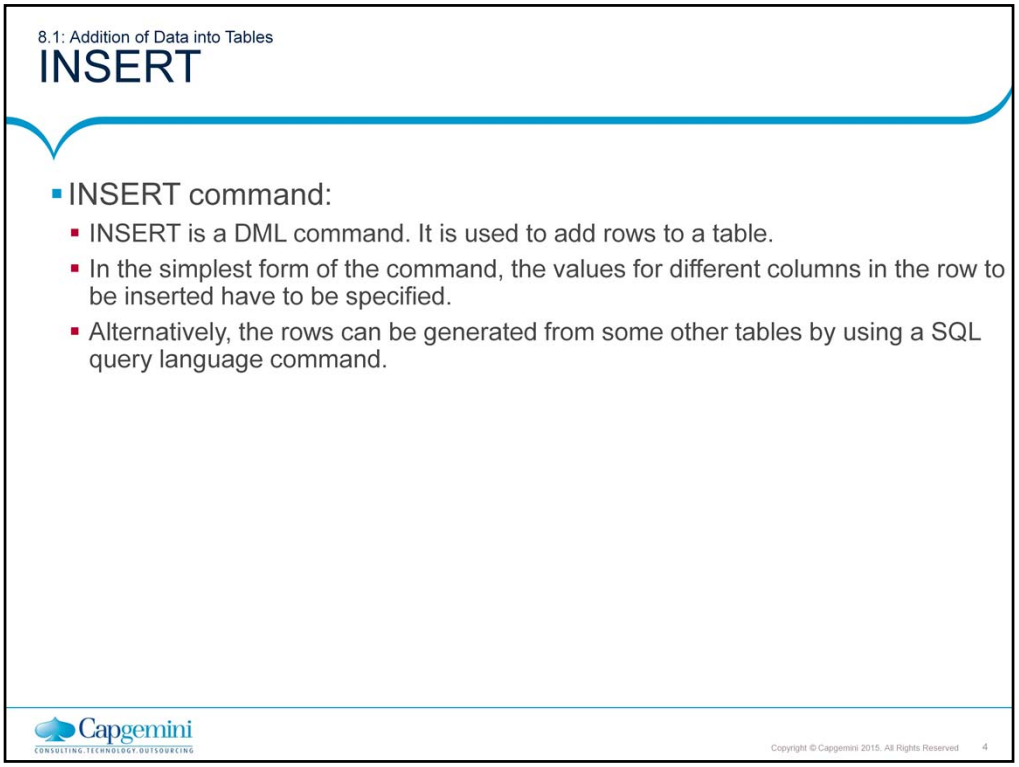
8.1: Concept of Data Manipulation Language

Data Manipulation Language

- Data Manipulation Language (DML) is used to perform the following routines on database information:
  - Retrieve
  - Insert
  - Modify
- DML changes data in an object. If you insert a row into a table, that is DML.
- All DML statements change data, and must be committed before the change becomes permanent.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

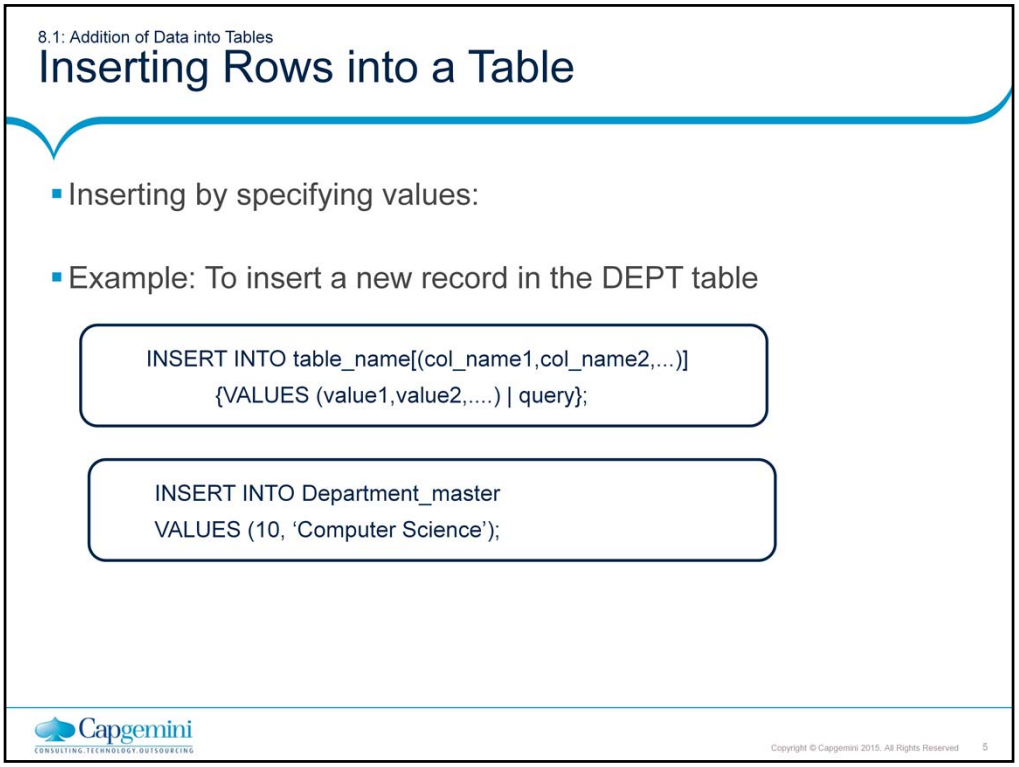
 3



Addition of Data into Tables:

Requisites for using INSERT command:

- If values are specified for all columns in the order specified at creation, then col_names could be omitted.
- Values should match “data type” of the respective columns.
- Number of values should match the number of column names mentioned.
- All columns declared as NOT NULL should be supplied with a value.
- Character strings should be enclosed in quotes.
- Date values should be enclosed in quotes.
- Values will insert one row at a time.
- Query will insert all the rows returned by the query.
- The table_name can be a “table” or a “view”. If table_name is a “view”, then the following restrictions apply:
    1. The “view” cannot have a GROUP BY, CONNECT BY, START WITH, DISTINCT, UNION, INTERSECT, or MINUS clause or a join.
    2. If the “view” has WITH CHECK OPTION clause, then a row, which will not be returned by the view, cannot be inserted.



Inserting Rows into a Table:

Example:

- Inserting a row in EMP table giving all values.

```
INSERT INTO student_master
VALUES(1001,'Amit',10,'11-Jan-80','Chennai');
```

  - 10 is a dept number which exists in DEPARTMENT_MASTER table

- Inserting a row in STAFF_MASTER table giving some values.

```
INSERT INTO staff_master
(staff_code,staff_name,design_code,dept_code)
VALUES(100001,'Arvind',102,30);
```

  - This row will be created if all the constraints like NOT NULL are satisfied.

8.1: Addition of Data into Tables

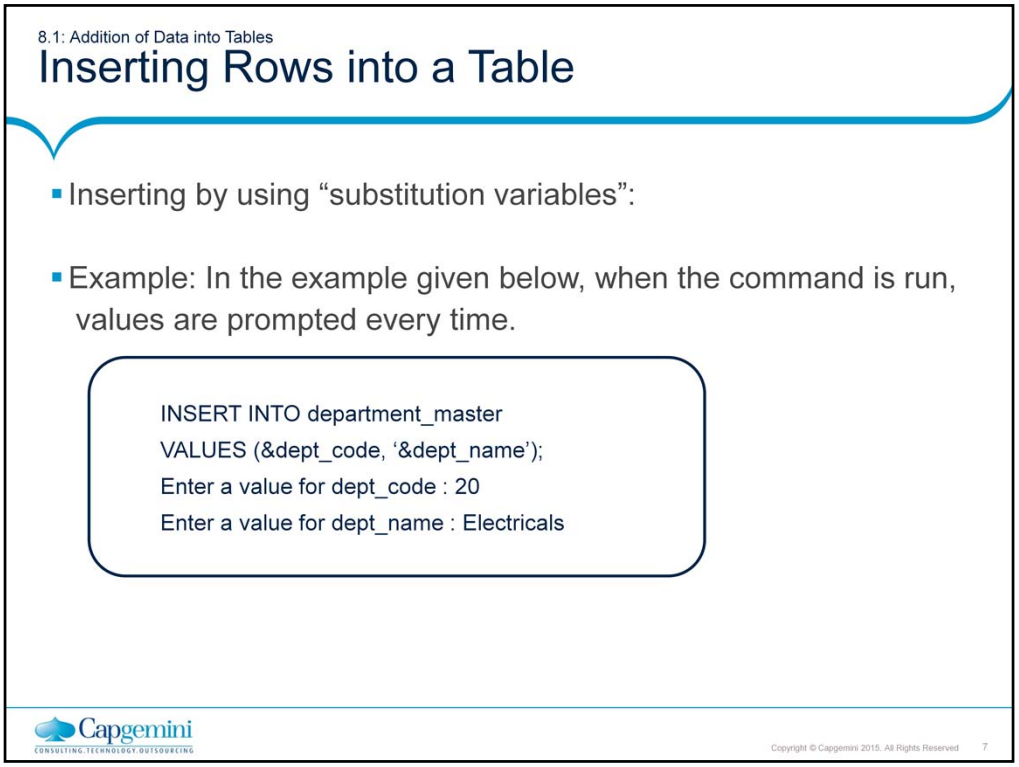
Inserting Rows into a Table

- Inserting rows in a table from another table using Subquery:
- Example: The example given below assumes that a new_emp_table exists. You can use a subquery to insert rows from another table.

```
INSERT INTO new_staff_table
SELECT * FROM staff_master
WHERE staff_master.hiredate > '01-jan-82';
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

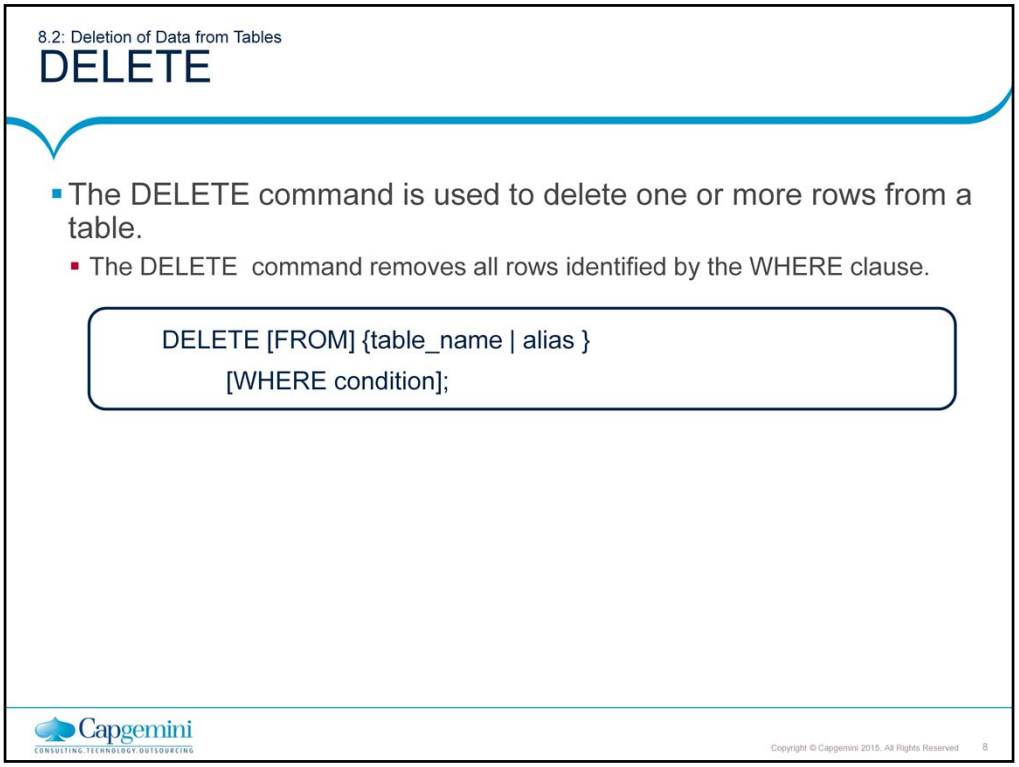
 6



Inserting Rows into a Table:

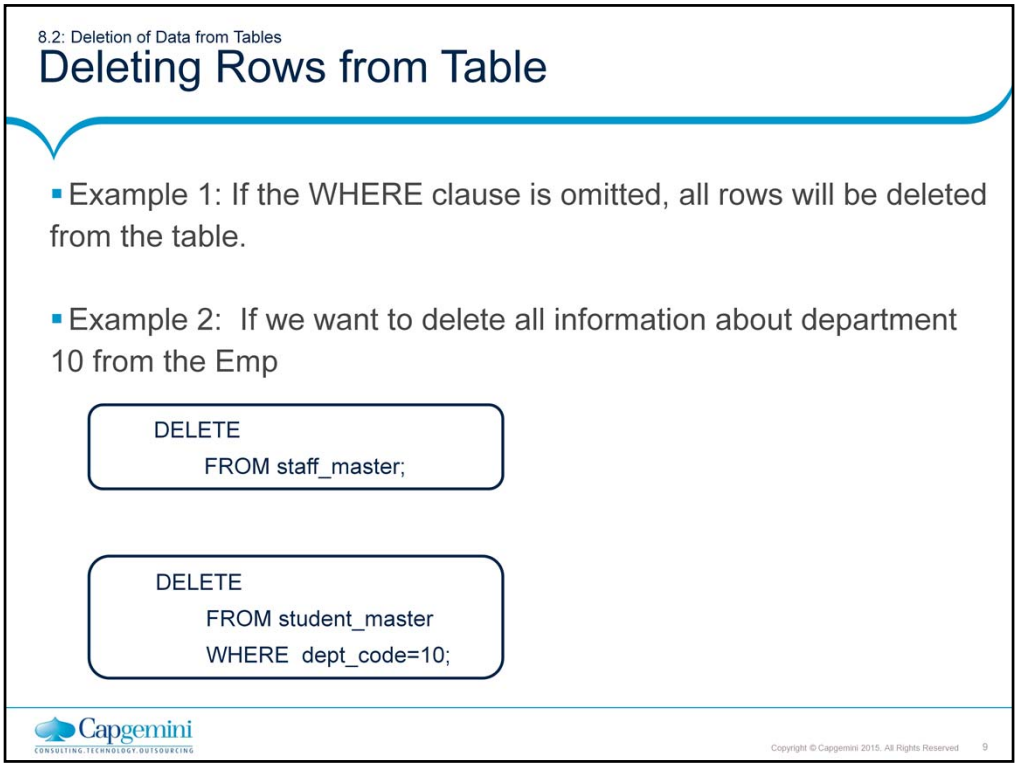
Inserting by using “substitution variables”:

- The problem with the INSERT statement is that it adds only “one row” to the table.
- However, by using “substitution variables” the speed of data input can be increased.
- Whenever a “substitution variable” is placed in a “value” field, the user will be prompted to enter the “actual value” when the command is executed.



Deletion of Data from Tables

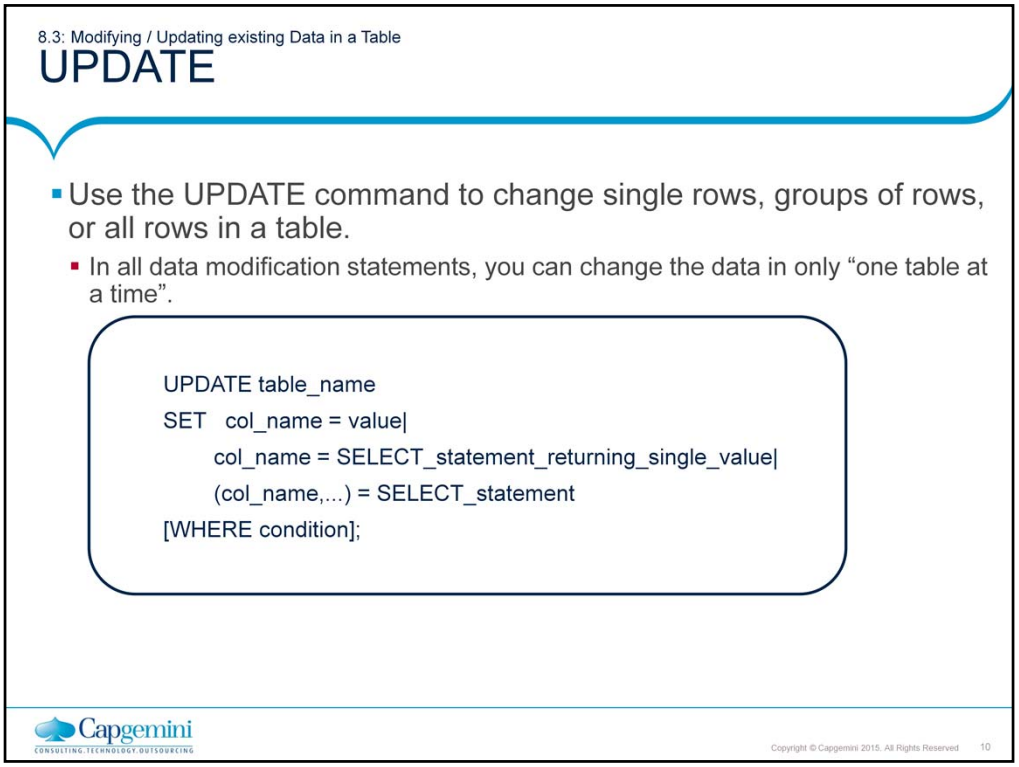
- The table_name can be a “table” or a “view”.
- The DELETE command is used to delete one or more rows from a table.
- The DELETE statement removes all rows identified by the WHERE clause.
  - This is another DML, which means we can rollback the deleted data, and that to make our changes permanent.
- If WHERE clause is omitted, all rows from the table are removed. Else all rows which satisfy the condition are removed.
- FROM clause can be omitted without affecting the statement.



Deletion of Data from Tables

Example 3:

```
DELETE staff_master WHERE staff_name = 'Anil';
```



Modifying / Updating existing Data in a Table:

- The table_name can be a “table” or a “view”.
- The “value” can be a value, an expression, or a query, which returns a single value.
- The UPDATE command provides automatic navigation to the data.
- **Note:** If the WHERE clause is omitted, all rows in the table will be updated by a value that is currently specified for the field. Else only those rows which satisfy the condition will be updated.

8.3: Modifying / Updating existing Data in a Table

Updating Rows from Table

- Example 1: To UPDATE the column “dname” of a row, where deptno is 10, give the following command:

```
UPDATE department_master
SET dept_name= ‘Information Technology’
WHERE dept_code=10;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

8.3: Modifying / Updating existing Data in a Table

Updating Rows from Table

- Example 2: To UPDATE the subject marks details of a particular student, give the following command:

```
UPDATE student_marks
SET subject1= 80 , subject2= 70
        WHERE student_code=1005;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

8.3: Modifying / Updating existing Data in a Table

Using a Subquery to do an Update

- For making salary of “Anil” equal to that of staff member 100006, use the following command:

```
UPDATE staff_master
SET staff_sal = (SELECT staff_sal FROM staff_master
                        WHERE staff_code = 100006 )
WHERE staff_name = 'Anil';
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Summary

- In this lesson, you have learnt:
  - The concept of Data Manipulation Language
  - Inserting rows into a table
  - Deleting rows from a table
  - Updating rows in a table

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Review - Questions

- Question 1: Both TRUNCATE statement and DELETE without condition removes the entire date from a table
  - True/False
- Question 2: All DML statements are auto committed
  - True/False
- Question 3: Inserting rows in a table emp1 from another table can be done using ___.
  - Option 1: insert into emp1(t1) as select empno from emp
  - Option 2: insert into emp1(t1) select empno from emp
  - Option 3: insert into emp1(t1) as select * from emp

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

DBMS/SQL

Lesson 09 Transaction Control Language

Lesson Objectives

- To understand the following topics:
  - Introduction to Transactions
  - Statement execution and Transaction control
  - Commit Transactions
  - Rollback transactions
  - Savepoints

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

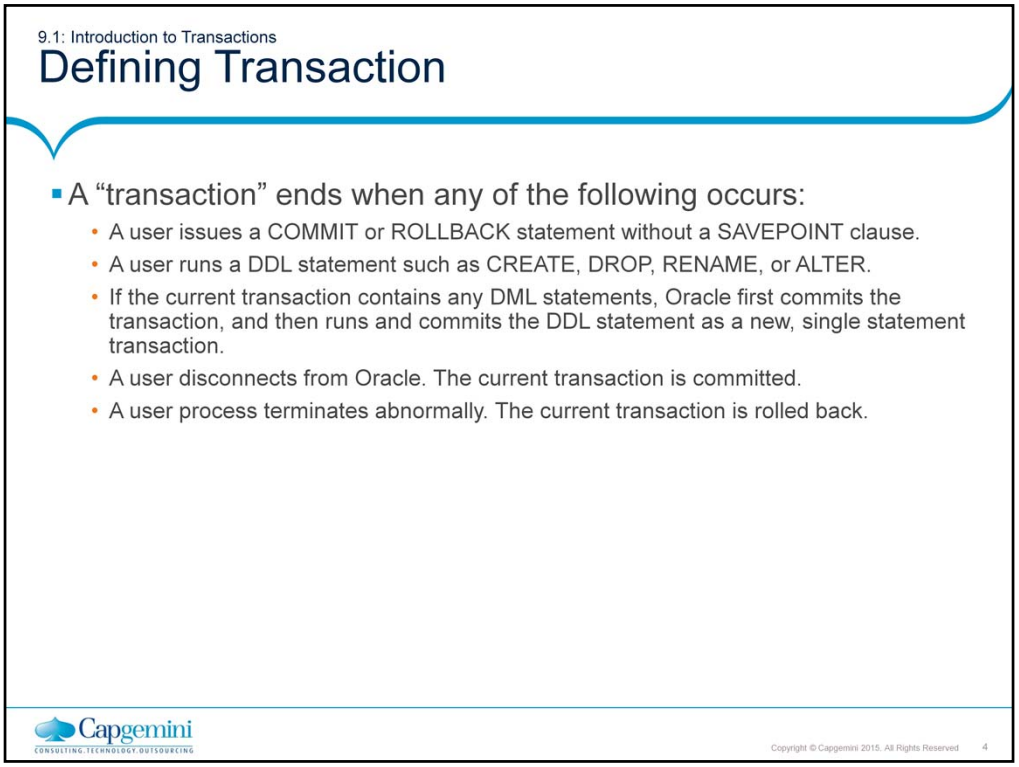
9.1: Introduction to Transactions

Defining Transaction

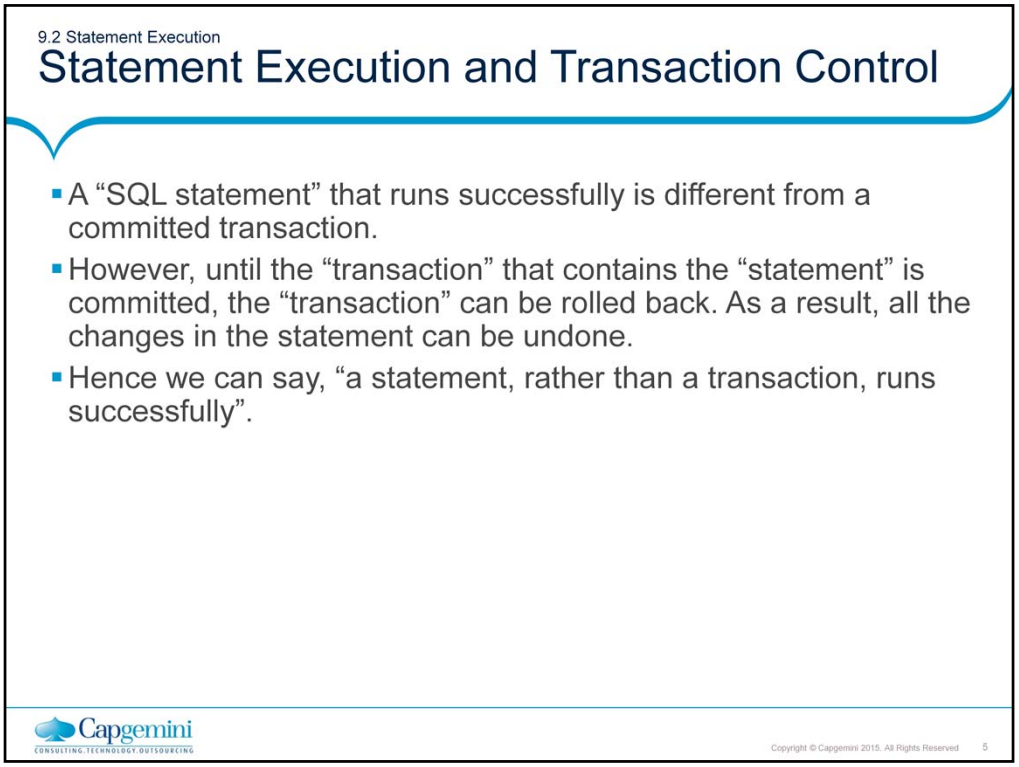
- A “transaction” is a logical unit of work that contains one or more SQL statements.
  - “Transaction” is an atomic unit.
  - The effects of all the SQL statements in a transaction can be either:
    - all committed (applied to the database), or
    - all rolled back (undone from the database)
  - A “transaction” begins with the first executable SQL statement.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

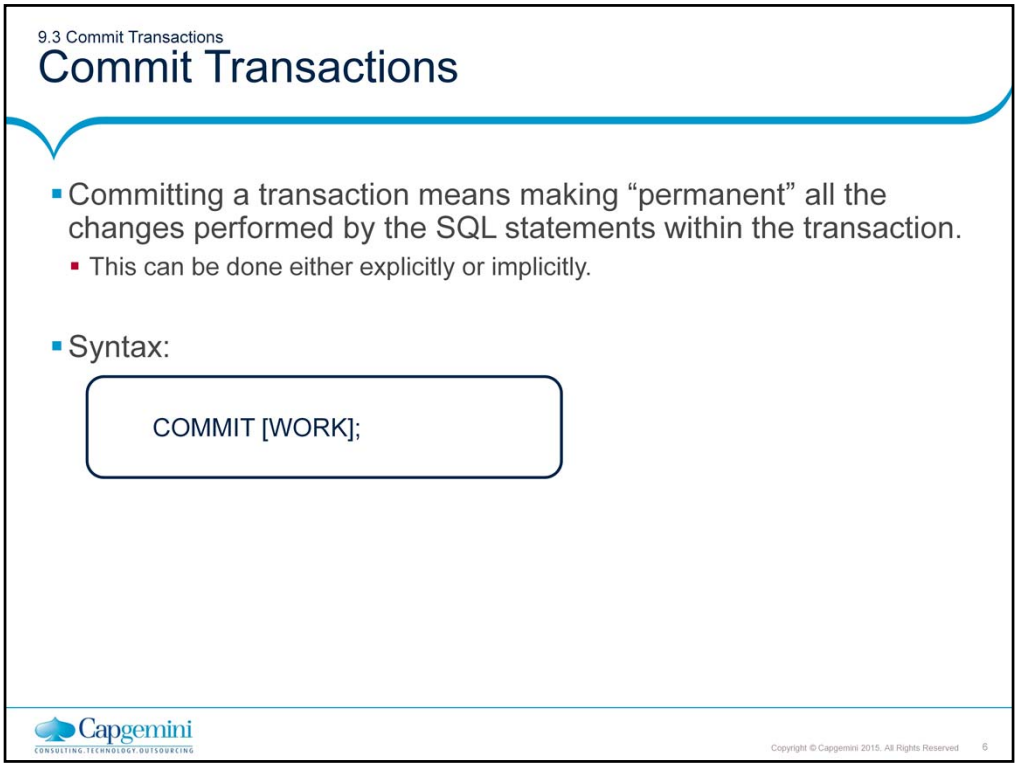


Note:
After one transaction ends, the next executable SQL statement automatically starts the subsequent transaction.



Statement Execution and Transaction Control:

- Executing successfully means that a single statement was:
  - Parsed
  - Found to be a valid SQL construction
  - Run without error as an atomic unit.
- For example: All rows of a multi-row update are changed.



Note:

- An “explicit request” occurs when the user issues a COMMIT statement.
- An “implicit request” occurs after normal termination of an application or completion of a data definition language (DDL) operation.
- The changes made by the SQL statement(s) of a transaction become permanent and visible to other users only after that transaction is committed. Queries that are issued after the transaction is committed will see the committed changes.

Statement-Level Rollback

- If at any time during execution, a SQL statement causes an error, then all effects of the statement are rolled back.
  - The effect of the rollback is as if that statement had never been run. This operation is a statement-level rollback.
- Errors discovered during SQL statement execution cause statement-level rollbacks.
  For example: Attempting to insert a duplicate value in a primary key.
- Single SQL statements involved in a deadlock (competition for the same data) can also cause a statement-level rollback.
- Errors discovered during SQL statement parsing, such as a syntax error, have not yet been run, so they do not cause a statement-level rollback.
- A SQL statement that fails causes a loss only of any work it would have performed by itself.
  - It does not cause the loss of any work that preceded it in the current transaction.
  - If the statement is a DDL statement, then the implicit commit that immediately preceded it is not undone.

9.3 Commit Transactions

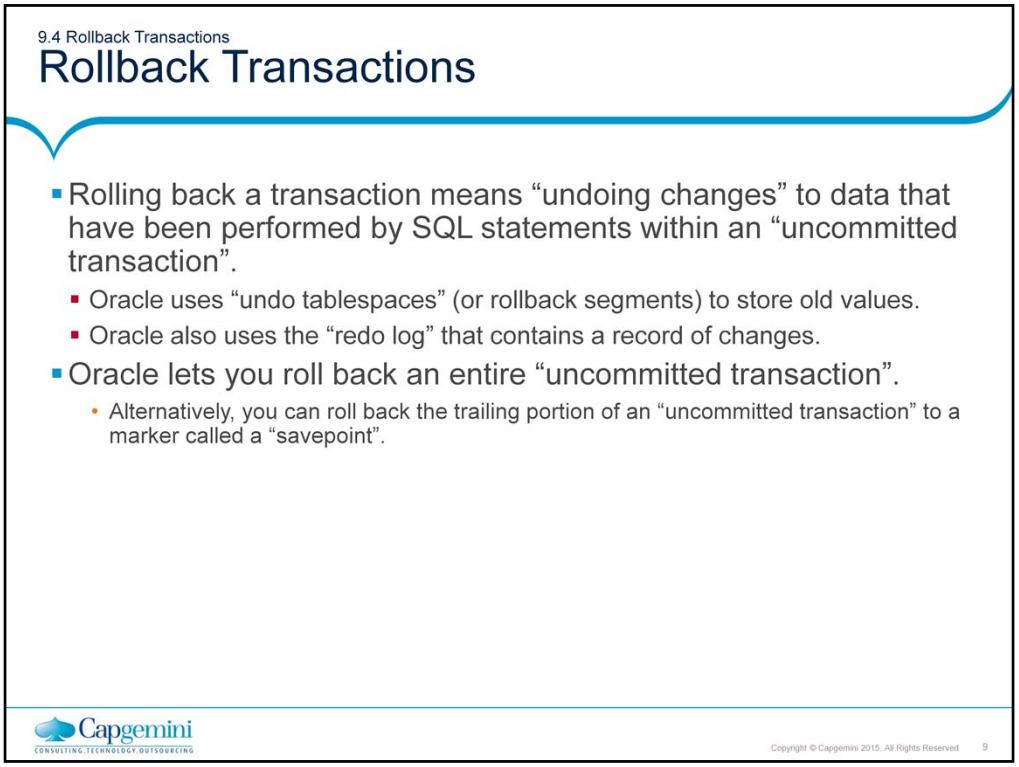
Commit Transactions

- COMMIT types:
  - Implicit: Database issues an implicit COMMIT before and after any data definition language (DDL) statement
  - Explicit
- Example of COMMIT command:

```
DELETE FROM student_master
WHERE student_name = 'Amit';
COMMIT ;
```

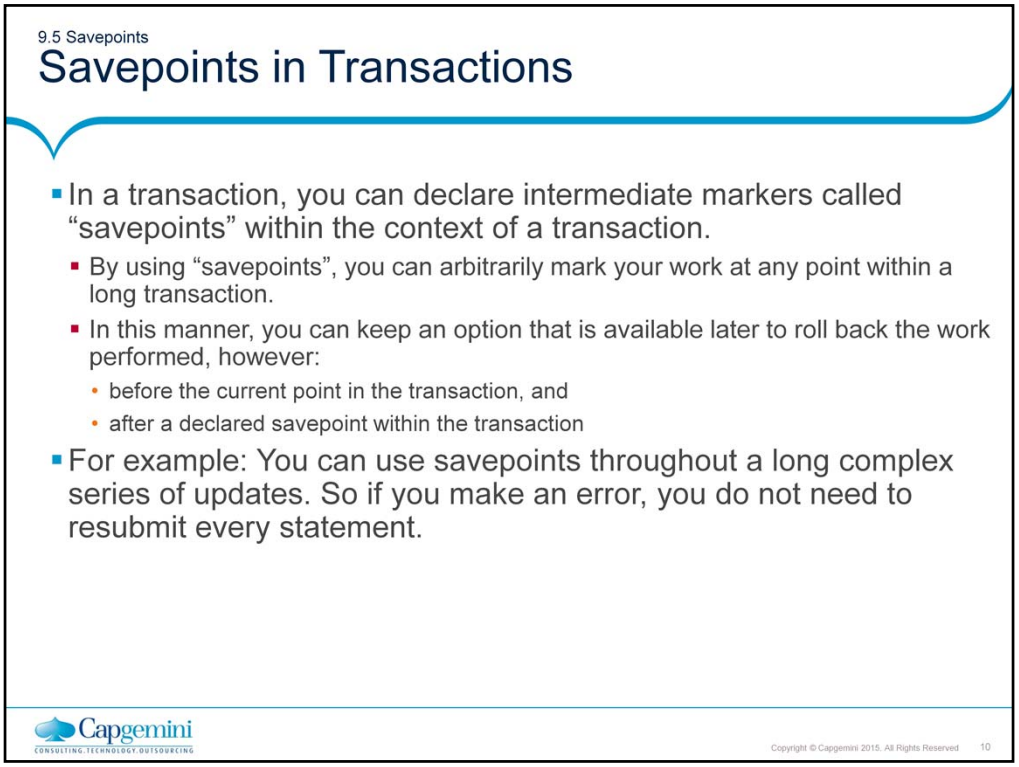
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8



Types of Roll back:

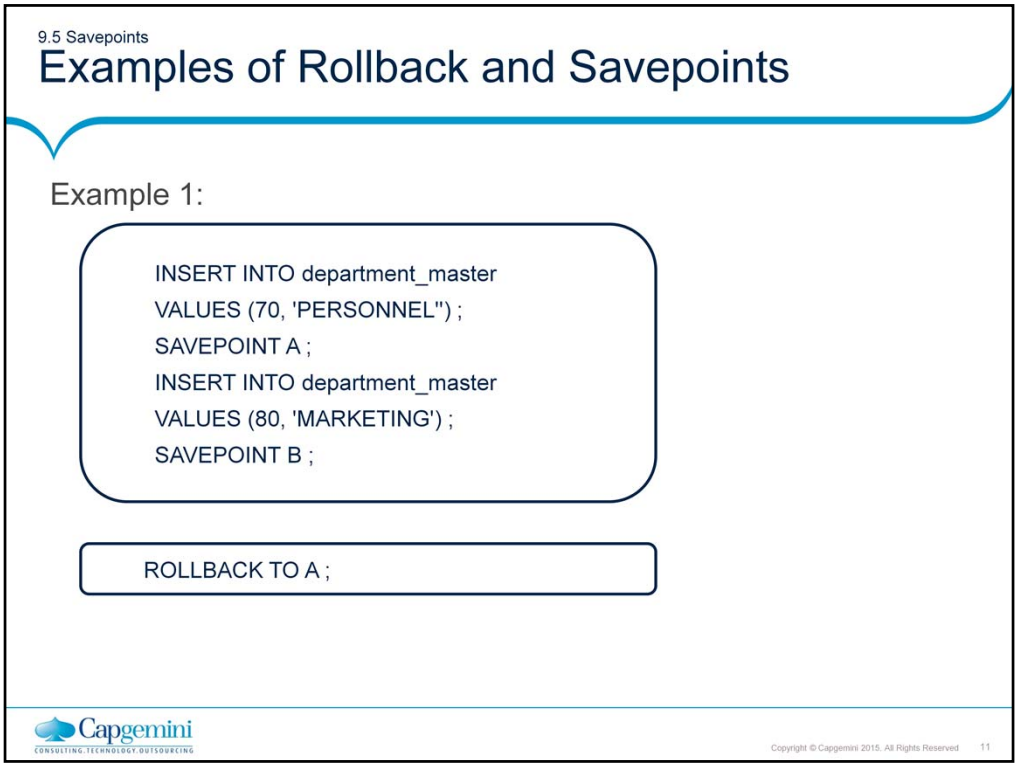
- All the following types of rollbacks use the same roll back procedure:
  - Statement-level rollback (due to statement or deadlock execution error)
  - Rollback to a savepoint
  - Rollback of a transaction due to user request
  - Rollback of a transaction due to abnormal process termination
  - Rollback of all outstanding transactions when an instance terminates abnormally
  - Rollback of incomplete transactions during recovery
- In rolling back an entire transaction, without referencing any savepoints, there is an occurrence of the following sequence:
  1. Oracle undoes all changes made by all the SQL statements in the transaction by using the corresponding undo tablespace.
  2. Oracle releases all the locks of data for the transaction.
  3. The transaction ends.



Savepoints in Transactions:

- Savepoints are useful in application programs, as well. If a procedure contains several functions, then you can create a savepoint at the beginning of each function.
  - Then, if a function fails, it is easy:
    - to return the data to it's state before the function began, and
    - to re-run the function with revised parameters or perform a RECOVERY action.
- After a rollback to a savepoint, Oracle releases the data locks obtained by rolled back statements. As a result:
  - Other transactions that were waiting for the previously locked resources can proceed.
  - Other transactions that want to update previously locked rows can do so.
- When a transaction is rolled back to a savepoint, there is an occurrence of the following sequence:
  1. Oracle rolls back only the statements run after the savepoint.
  2. Oracle preserves the specified savepoint, but all savepoints that were established after the specified savepoint are lost.
  3. Oracle releases all table and row locks acquired since that savepoint, however, it retains all data locks acquired previous to the savepoint.

  The transaction remains active and can be continued.



Examples of Rollback and Savepoints:
In the example in the above slide, after execution of 'ROLLBACK TO A' command, 'INSERT INTO department_master VALUES (70, 'PERSONNEL')' statement got committed (stored in a table permanently) and all other statements after the SAVEPOINT A, will get rolledback (undone).

Advantages of COMMIT and ROLLBACK statements:
With COMMIT and ROLLBACK statements, you can:
- ensure data consistency
- preview data changes before making changes permanent
- group logically related operations

State of the Data after COMMIT:
- Data changes are made permanent in the database.
- The previous state of the data is permanently lost.
- All users can view the results.
- Locks on the affected rows are released. Those rows are available for other users to manipulate.
- All savepoints are erased.

State of the Data after ROLLBACK:
DBMS discards all pending changes by using the ROLLBACK statement:
- Data changes are undone.
- Previous state of the data is restored.
- Locks on the affected rows are released

Syntax:

```
UPDATE...
COMMIT;
```

Example of rolling back changes to a Marker:
- Create a marker in a current transaction by using the SAVEPOINT statement.
- Roll back to that marker by using the ROLLBACK TO SAVEPOINT statement.

```
INSERT...
ROLLBACK;
```

```
UPDATE...
SAVEPOINT update_done;
Savepoint created.
INSERT...
ROLLBACK TO update_done;
Rollback complete.
```

Summary

- In this lesson, you have learnt:
  - Transactions
    - Statement execution
    - Transaction control
    - Commit Transactions
    - Rollback transactions
    - Savepoints

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Review – Questions

- Question 1 : ___ is a logical unit of work.

- Question 2: A transaction is committed when the user issues a DDL statement.
  - True/False
- Question 3: A transaction is rolled back when ___.
  - Option 1: rollback statement is issued
  - Option 2: the user session is abruptly terminated
  - Option 3: an error occurs in DML statement
  - Option 4: none of the above
- Question 4:In a transaction, DDL statement after DML statement commits the changes done by DML.
  - True/False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Oracle

(DBMS SQL)

Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes
05-Feb-2009	0.1D	Rajita Dhumal	Content Creation
09-Feb-2009		CLS team	Review
02- Jun-2011	2.0	Anu Mitra	Integration Refinements
30-Nov-2012	3.0	Karthikeyan R	Revamp of lab assignments
26-Feb-2015	4.0	Kavita Arora	Rearranging of lab assignments
07-May-2016	5.0	Kavita Arora	Integration Refinements

Table of Contents

Getting Started

Overview

This lab book is a guided tour for learning DBMS SQL. It comprises 'To Do' assignments. Follow the steps provided and work out the 'To Do' assignments.

Setup Checklist for DBMS SQL

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 32MB of RAM (64MB or more recommended)

Please ensure that the following is done:

- Oracle Client installed.
- Connectivity to Oracle Server

Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory <DBMS SQL>_assgn. For each lab exercise create a directory as lab <lab number>.

Learning More (Bibliography if applicable)

NA

Problem Statement / Case Study

Table descriptions

Emp

Name	Null?	Type
Empno	Not Null	Number(4)
Ename	-	Varchar2(10)
job		Varchar2(9)
mgr		Number(4)
Hiredate		Date
Sal	-	Number(7,2)
Comm	-	Number(7,2)
Deptno	-	Number(2)

Designation_Master

Name	Null?	Type
Design_code	Not Null	Number(3)
Design_name		Varchar2(50)

Department_Master

Name	Null?	Type
Dept_Code	Not Null	Number(2)
Dept_name		Varchar2(50)

Student_Master

Name	Null?	Type
Student_Code	Not Null	Number(6)
Student_name	Not Null	Varchar2(50)
Dept_Code		Number(2)
Student_dob		Date
Student_Address		Varchar2(240)

Student_Marks

Name	Null?	Type
Student_Code		Number(6)
Student_Year	Not Null	Number
Subject1		Number(3)
Subject2		Number(3)
Subject3		Number(3)

Staff_Master

Name	Null?	Type
Staff_code	Not Null	Number(8)
Staff_Name	Not Null	Varchar2(50)
Design_code		Number
Dept_code		Number
HireDate		Date
Staff_dob		Date
Staff_address		Varchar2(240)
Mgr_code		Number(8)
Staff_sal		Number (10,2)

Book_Master

Name	Null?	Type
Book_Code	Not Null	Number(10)
Book_Name	Not Null	Varchar2(50)
Book_pub_year		Number

Book_pub_author	Not Null	Varchar2(50)

Book_Transactions

Name	Null?	Type
Book_Code		Number
Student_code		Number
Staff_code		Number
Book_Issue_date	Not Null	Date
Book_expected_return_date	Not Null	Date
Book_actual_return_date		Date

Data Query Language

Goals	• Query the database • Usage of various operators in select statements
Time	45 mins

1.1: Data Query Language

1. List the Name and Designation code of the staff who have joined before Jan 2003 and whose salary range is between 12000 and 25000. Display the columns with user defined Column headers. Hint: Use As clause along with other operators
2. List the staff code, name, and department number of the staff who have experience of 18 or more years and sort them based on their experience.
3. Display the staff details who do not have manager. Hint: Use is null
4. Display the Book details that were published during the period of 2001 to 2004. Also display book details with Book name having the character '&' anywhere.
5. List the names of the staff having '_' character in their name.

Single Row And Group Functions

Goals	• Querying tables using single row functions • Querying tables using date functions • Querying tables using number functions • Querying tables using group functions
Time	2 hrs

2.1: Single Row Functions:

1. Create a query which will display Staff Name, Salary of each staff. Format the salary to be 15 characters long and left padded with '$'.
2. Display name and date of birth of students where date of birth must be displayed in the format similar to "January, 12 1981" for those who were born on Saturday or Sunday.
3. Display each Staff name and number of months they worked for the organization. Label the column as 'Months Worked'. Order your result by number of months employed. Also Round the number of months to closest whole number.
4. List the details of the staff who have joined in first half of December month (irrespective of the year).
5. Write a query that displays Staff Name, Salary, and Grade of all staff. Grade depends on the following table.

Salary	Grade
Salary >=50000	A
Salary >= 25000 < 50000	B
Salary>=10000 < 25000	C
OTHERS	D

6. Display the Staff Name, Hire date and day of the week on which staff was hired. Label the column as DAY. Order the result by the day of the week starting with Monday. Hint :Use to_char with hiredate and formats 'DY' and 'D'

7. Write a query to find the position of third occurrence of 'i' in the given word 'Mississippi'.
8. Write a query to find the pay date for the month. Pay date is the last Friday of the month. Display the date in the format "Twenty Eighth of January, 2002". Label the heading as PAY DATE. Hint: use to_char, next_day and last_day functions
9. Display Student code, Name and Dept Name. Display "Electricals" if dept code = 20, "Electronics" if Dept code =30 and "Others" for all other Dept codes in the Dept Name column. Hint : Use Decode

2.2: Group Functions:

1. Display the Highest, Lowest, Total & Average salary of all staff. Label the columns Maximum, Minimum, Total and Average respectively for each Department code. Also round the result to the nearest whole number.
2. Display Department code and number of managers working in that department. Label the column as 'Total Number of Managers' for each department.
3. Get the Department number, and sum of Salary of all non-managers where the sum is greater than 20000.

JOINS AND SUBQUERIES

Goals	• Querying multiple tables using joins • Querying tables using subqueries
Time	2 hr 30 min

3.1: Joins and Subqueries

1. Write a query which displays Staff Name, Department Code, Department Name, and Salary for all staff who earns more than 20000.
2. Display Staff Code, Staff Name, Department Name, and his manager's number and name. Label the columns Staff#, Staff, Mgr#, Manager.
3. Create a query that will display Student Code, Student Name, Book Code, and Book Name for all students whose expected book return date is today.
4. Create a query that will display Staff Code, Staff Name, Department Name, Designation name, Book Code, Book Name, and Issue Date for only those staff who have taken any book in last 30 days. . If required, make changes to the table to create such a scenario.
5. Generate a report which contains the following information.

   Staff Code, Staff Name, Designation Name, Department, Book Code, Book Name,

   Author, Fine For the staff who has not returned the book. Fine will be calculated as Rs. 5 per day.

   Fine = 5 * (No. of days = Current Date – Expected return date). Include records in the table to suit this problem statement
6. List Staff Code, Staff Name, and Salary for those who are getting less than the average salary of organization.
7. Display Author Name, Book Name for those authors who wrote more than one book.

8. Display Staff Code, Staff Name, and Department Name for those who have taken more than one book.
9. Display the Student Code, Student Name, and Department Name for that department in which there are maximum number of student studying.
10. Display Staff Code, Staff Name, Department Name, and Designation name for those who have joined in last 3 months.
11. Display the Manager Name and the total strength of his/her team.
12. Display the details of books that have not been returned and expected return date was last Monday. Book name should be displayed in proper case.. Hint: You can change /add records so that the expected return date suits this problem statement
13. Write a query to display number of people in each Department. Output should display Department Code, Department Name and Number of People.

Database Objects

Goals	Following set of questions are designed to implement the following concepts • Creating Database objects like tables, views , etc • Modifying Database objects • Deleting Database objects • Usage of Data Dictionary tables
Time	2 hr 30 min

4.1: Database Objects

1. Create the Customer table with the following columns.

   CustomerId Number(5)

   Cust_Name varchar2(20)

   Address1 Varchar2(30)

   Address2 Varchar2(30)

2. Modify the Customer table Cust_Name column of datatype with Varchar2(30), rename the column to CustomerName and it should not accept Nulls.

3. a) Add the following Columns to the Customer table.

   Gender Varchar2(1)

   Age Number(3)

   PhoneNo Number(10)

   b) Rename the Customer table to Cust_Table

4. Insert rows with the following data in to the Customer table.

   Insert into customer values: (1000, ‘Allen’, ‘#115 Chicago’, ‘#115 Chicago’, ‘M’, ‘25, 7878776’)

In similar manner, add the below records to the Customer table:

1001, George, #116 France, #116 France, M, 25, 434524

1002, Becker, #114 New York, #114 New York, M, 45, 431525

5. Add the Primary key constraint for CustomerId with the name CustId_Prim.
6. Insert the row given below in the Customer table and see the message generated by the Oracle server.

   1002, John, #114 Chicago, #114 Chicago, M, 45, 439525

7. Disable the constraint on CustomerId, and insert the following data:

   1002, Becker, #114 New York, #114 New york , M, 45, 431525

   1003, Nanapatekar, #115 India, #115 India , M, 45, 431525

8. Enable the constraint on CustomerId of the Customer table, and see the message generated by the Oracle server.
9. Drop the constraint CustId_Prim on CustomerId and insert the following Data. Alter Customer table, drop constraint Custid_Prim.

   1002, Becker, #114 New York, #114 New york , M, 45, 431525, 15000.50

   1003, Nanapatekar, #115 India, #115 India , M, 45, 431525, 20000.50

10. Delete all the existing rows from Customer table, and let the structure remain itself using TRUNCATE statement.
11. In the Customer table, add a column E_mail.
12. Drop the E_mail column from Customer table.
13. Create the Suppliers table based on the structure of the Customer table. Include only the CustomerId, CustomerName, Address1, Address2, and phoneno columns.

    Name the columns in the new table as SuppID, SName, Addr1, Addr2, and Contactno respectively.

14. Drop the above table and recreate the following table with the name CustomerMaster.

    CustomerId Number(5) Primary key(Name of constraint is CustId_PK)

    CustomerName Varchar2(30) Not Null

    Addressl Varchar2(30) Not Null

    Address2 Varchar2(30)

Gender	Varchar2(l)
Age	Number(3)
PhoneNo	Number(10)

15. Create the AccountsMaster table with the following Columns. Use sequence to generate Account number

CustomerId	Number(5)
AccountNumber	Number(10,2) Primary key(Name of constraint is Acc_PK)
AccountType	Char(3)
LedgerBalance	Number(10,2) Not Null

16. Relate AccountsMaster table and CustomerMaster table through CustomerId column with the constraint name Cust_acc.
17. Insert the following rows to the CustomerMaster table:

    1000, Allen, #115 Chicago, #115 Chicago, M, 25, 7878776

    1001, George, #116 France, #116 France, M, 25, 434524

    1002, Becker, #114 New York, #114 New York, M, 45, 431525
18. Modify the AccountMaster table with the Check constraint to ensure AccountType should be either NRI or IND.
19. Modify the AccountsMaster table keeping a Check constraint with the name Balance_Check for the Minimum Balance which should be greater than 5000.
20. Modify the AccountsMaster table such that if Customer is deleted from Customer table then all his details should be deleted from AccountsMaster table.
21. Create Backup copy for the AccountsMaster table with the name ‘AccountDetails’.
22. Create a view ‘Acc_view’ with columns CustomerId, CustomerName, AccountNumber, AccountType, and LedgerBalance from AccountsMaster. In the view Acc_view, the column names should be CustomerCode, AccountHolderName, AccountNumber, Type, and Balance for the respective columns from AccountsMaster table.
23. Create a view on AccountsMaster table with name vAccs_Dtls. This view should list all customers whose AccountType is ‘IND’ and their balance amount should

not be less than 10000. Using this view any DML operation should not violate the view conditions.


Hint: Use the With Check Option constraint.

24. Create a view accsvw10 which will not allow DML statement against it.
25. Create a Sequence with the name Seq_Dept on Deptno column of Department_Masters table. It should start from 40 and stop at 200. Increment parameter for the sequence Seq_Dept should be in step of 10.
26. Insert three sample rows by using the above sequence in Department_Masters table.
27. Drop the Seq_Dept sequence.
28. Get information on the index No_Name from the Data Dictionary.
29. Create synonym synEmp for the EMP table.
30. Get Information on synonym synEmp from the Data Dictionary.
31. Note: Perform this after creating the Employee Table mentioned in the next Lab assignment. Create Index on HireDate column and give the name as idx_emp_hiredate for this object.
32. Create a Sequence with the name Seq_Emp on Empno column of Employee table. It should start from 1001. Try to set Minimum value for this sequence which is less than / greater than 1001, use the sequence to generate Empno while inserting records in Employee table and check the values generated.

Data Manipulation Language

Goals	Creating table to do the following DML operations • Insert Records • Delete Records • Update Records
Time	30 mins

5.1: Data Manipulation Language

1. Create Employee table with same structure as EMP table.

   SQL>Create table employee as select * from emp where 1=3

   SQL>desc employee

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(50)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

SQL>select * from employee

2. Write a query to populate Employee table using EMP table's empno, ename, sal, deptno columns.

SQL>select * from employee

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH				800		20
7499	ALLEN				1600		30
7521	WARD				1250		30
7566	JONES				2975		20
7654	MARTIN				1250		30
7698	BLAKE				2850		30
7782	CLARK				2450		10
7788	SCOTT				3000		20
7839	KING				5000		10
7844	TURNER				1500		30
7876	ADAMS				1100		20
7900	JAMES				950		30
7902	FORD				3000		20
7934	MILLER				1300		10

14 rows selected.

3. Write a query to change the job and deptno of employee whose empno is 7698 to the job and deptno of employee having empno 7788.

4. Delete the details of department whose department name is ‘SALES’.

5. Write a query to change the deptno of employee with empno 7788 to that of employee having empno 7698.

6. Insert the following rows to the Employee table through parameter substitution.

- 1000,Allen, Clerk,1001,12-jan-01, 3000, 2,10
- 1001,George, analyst, null, 08 Sep 92, 5000,0, 10
- 1002, Becker, Manager, 1000, 4 Nov 92, 2800,4, 20
- 1003, 'Bill', Clerk, 1002, 4 Nov 92,3000, 0, 20

Transaction Control Language Statements

Goals	• Creating a transaction • Creating a savepoint • Roll back and commit the transaction to the savepoint
Time	30 min

6.1: Transaction Control Language Statements

1. Insert rows with the following data into the Customer table. 6000, John, #115 Chicago, #115 Chicago, M, 25, 7878776, 10000

- 6001, Jack, #116 France, #116 France, M, 25, 434524, 20000
- 6002, James, #114 New York, #114 New York, M, 45, 431525, 15000.50

Use parameter substitution.

2. Create a Savepoint named ‘SP1’ after third record in the Customer table .
3. Insert the below row in the Customer table.

   6003, John, #114 Chicago, #114 Chicago, M, 45, 439525, 19000.60

4. Execute rollback statement in such a way that whatever manipulations done before Savepoint sp1 are permanently implemented, and the ones after Savepoint SP1 are not stored as a part of the Customer table.

Appendices

Appendix A: DBMS SQL Standards

Key points to keep in mind:

NA

How to follow DBMS SQL standards:

- Decide upon a database naming convention, standardize it across your organization and be consistent in following it. It helps make your code more readable and understandable.
- Tables:
  - Tables represent the instances of an entity. For example, you store all your customer information in a table. Here “customer” is an entity and all the rows in the customers table represent the instances of the entity “customer”. So name your table using the entity it represents, namely “Customer”. Since the table is storing “multiple instances” of customers, make your table name a plural word.
  - Keeping this in mind:
    - name your customer table as “Customers'”
    - name your order storage table as “Orders”
    - name your error messages table as “ErrorMessages”
  - Suppose your database deals with different logical functions and you want to group your tables according to the logical group they belong to. It will help prefixing your table name with a two or three character prefix that can identify the group.
  - For example: Your database has tables which store information about Sales and Human resource departments, then you can name all your tables related to Sales department as shown below:
    - SL_NewLeads
    - SL_Territories
    - SL_TerritoriesManagers

- You can name all your tables related to Human resources department as shown below:
    - HR_Candidates
    - HR_PremierInstitutes
    - HR_InterviewSchedules
- Prefix the table names with owner names, as this improves readability, and avoids any unnecessary confusion.

- Primary keys:
  - Primary key is the column(s) that can uniquely identify each row in a table. So just use the column name prefixed with “pk_” + “Table name” for naming primary keys.
  - Given below is an example of how we can name the primary key on the CustomerID column of Customers table:
    - pk_Customers_CustomerID
  - Consider concatenating the column names in case of composite primary keys.
- Foreign keys:
  - Foreign keys are used to represent the relationships between tables which are related. So a foreign key can be considered as a link between the “column of a referencing table” and the “primary key column of the referenced table”.
  - We can use the following naming convention for foreign keys:
    - fk_referencing table + referencing column_referenced table + referenced column
  - Based on the above convention, we can name the foreign key, which references the CustomerID column of the Customers table from the Order’s tables CustomerID column as shown below:
    - fk_OrdersCustomerID_CustomersCustomerID
  - Foreign key can be composite too. In that case, consider concatenating the column names of referencing and referenced tables while naming the foreign key. This might make the name of the foreign key lengthy. However, you

should not be worried about it, as you will never reference this name from your code, except while creating/dropping these constraints.

- Default and Check constraints:
  - Use the column name to which the defaults /check constraints are bound to. Prefix it with “def” and “chk” prefixes respectively for Default and Check constraints.
  - We can name the default constraint for OrderDate Column as def_OrderDate, and the check constraint for OrderDate column as chk_OrderDate.
- Do not use any reserved words for naming my database objects, as that can lead to some unpredictable situations.
- Do not depend on undocumented functionality. The reasons are:
  - you will not get support, when something goes wrong with your undocumented code
  - undocumented functionality is not guaranteed to exist (or behave the same) in a future release or service pack, thereby breaking your code
- Make sure you normalize your data at least till third normal form. At the same time, do not compromise on query performance. A little de-normalization helps queries perform faster.

Appendix B: Coding Best Practices

Given below are a few best practices in DBMS SQL:

- Do not use SELECT * in your queries. Always write the required column names after the SELECT statement, as shown below:
  SELECT CustomerID, CustomerFirstName, City

  This technique results in less disk IO, less network traffic, and hence better performance.

- Try to avoid wildcard characters at the beginning of a word while searching by using the LIKE keyword. This is because this arrangement results in an index scan, which is defeating the purpose of having an index. The following statement results in an index scan, while the second statement results in an index seek:
  - 1. SELECT LocationID FROM Locations WHERE Specialities LIKE '%pples'
  - 2. SELECT LocationID FROM Locations WHERE Specialities LIKE 'A%s'

  Also avoid searching with not equals operators (<> and NOT) as they result in table and index scans. If you must do heavy text-based searches, consider using the Full-Text search feature of SQL Server for better performance.

- Use char data type for a column, only when the column is non-nullable. If a char column is nullable, it is treated as a fixed length column in SQL Server 7.0+. So a char(100), when NULL, will eat up 100 bytes, resulting in space wastage. So use varchar(100) in this situation. Of course, variable length columns do have a very little processing overhead over fixed length columns. Carefully choose between char and varchar depending upon the length of the data you are going to store.

- Always use a column list in your INSERT statements. This helps in avoiding problems when the table structure changes (like adding a column).

- Do not use the column numbers in the ORDER BY clause as it impairs the readability of the SQL statement. Further, changing the order of columns in the SELECT list has no impact on the ORDER BY when the columns are referred by names instead of numbers. Consider the following example, in which the second query is more readable than the first one:

```
SELECT OrderID, OrderDate
FROM Orders
ORDER BY 2

SELECT OrderID, OrderDate
FROM Orders
ORDER BY OrderDate
```


Appendix C: Debugging Examples

1. Identify the mistake in the query (if any) in terms of computed value.

   - SELECT empno, ename, sal+comm as Total FROM emp;

2. CREATE TABLE ITEM_MASTER

   (ITEM_NO VARCHAR2(4) PRIMARY KEY,ITEM_DESC VARCHAR2(25));

```
INSERT INTO ITEM_MASTER VALUES ('1001', 'pen');
INSERT INTO ITEM_MASTER VALUES ('1002', 'Pencil  ');
INSERT INTO ITEM_MASTER VALUES ('1003', 'Eracer');
INSERT INTO ITEM_MASTER VALUES ('1104', 'Keyboard MF');
INSERT INTO ITEM_MASTER VALUES ('1005', 'Gel');
INSERT INTO ITEM_MASTER VALUES ('1006', 'chart');
INSERT INTO ITEM_MASTER VALUES ('1007', 'Map');
INSERT INTO ITEM_MASTER VALUES ('1008', 'note book');
INSERT INTO ITEM_MASTER VALUES ('1009', 'STAPLER');
INSERT INTO ITEM_MASTER VALUES ('1010', '  story book');
INSERT INTO ITEM_MASTER VALUES ('1011', 'Calculator');
INSERT INTO ITEM_MASTER VALUES ('2012', 'Writing Pad(S)');
INSERT INTO ITEM_MASTER VALUES ('1013', 'Geometry box');
INSERT INTO ITEM_MASTER VALUES ('1014', 'Id card holder');
INSERT INTO ITEM_MASTER VALUES ('1015', 'FILE');
INSERT INTO ITEM_MASTER VALUES ('1016', 'Bag');
INSERT INTO ITEM_MASTER VALUES ('1017', 'Mobile Pouch');
INSERT INTO ITEM_MASTER VALUES ('1018', 'Punching Machine');
```

a) Correct the mistake in the below query to get the list in ascending order of item_no:
SELECT item_no, item_desc FROM item_master order by item_no;

b) Identify the mistake, if any to get the list of records correctly:
SELECT item_no, item_desc FROM item_master WHERE item_desc like 'k%' or like 'p%' or item_desc like 'S%';

Hint: use Conversion Functions for above Qs

3. Rewrite the subquery below to correct the errors(if any)

```
SELECT staff_name,dept_name,staff_sal
    FROM  staff_master s,department_master d
     WHERE s.dept_code = sm.dept_code AND   --d
       staff_sal < ALL (SELECT AVG(staff_sal) FROM staff_master sm WHERE
department_code = s.dept_code);
```

4. Complete the below query to increase the salary for those people whose salary is less than their Department's average

```
 ---
UPDATE emp a
SET sal = (select avg(sal) FROM enmp b where
....your query....
)
WHERE SAL <
....your query...
```

