ORACLE SQL

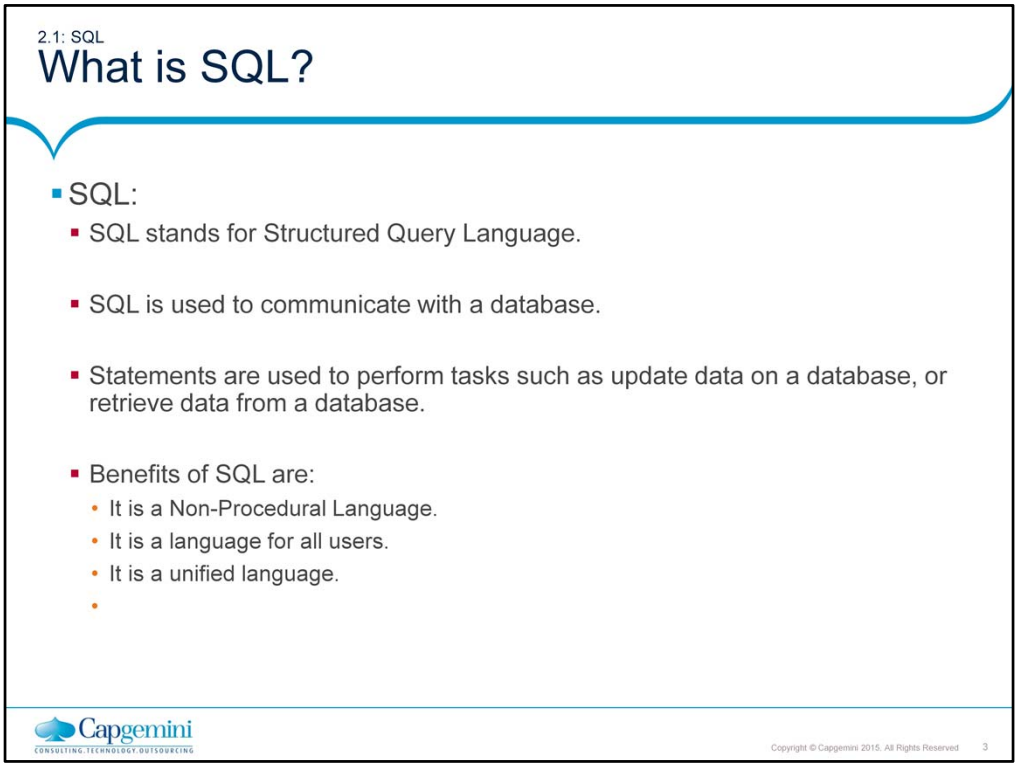
Lesson 01: SQL Basic Commands

Lesson Objectives

- To understand the following topics:
  - SQL Language
  - DDL
  - DML
  - DRL
  - TCL
  - DCL

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

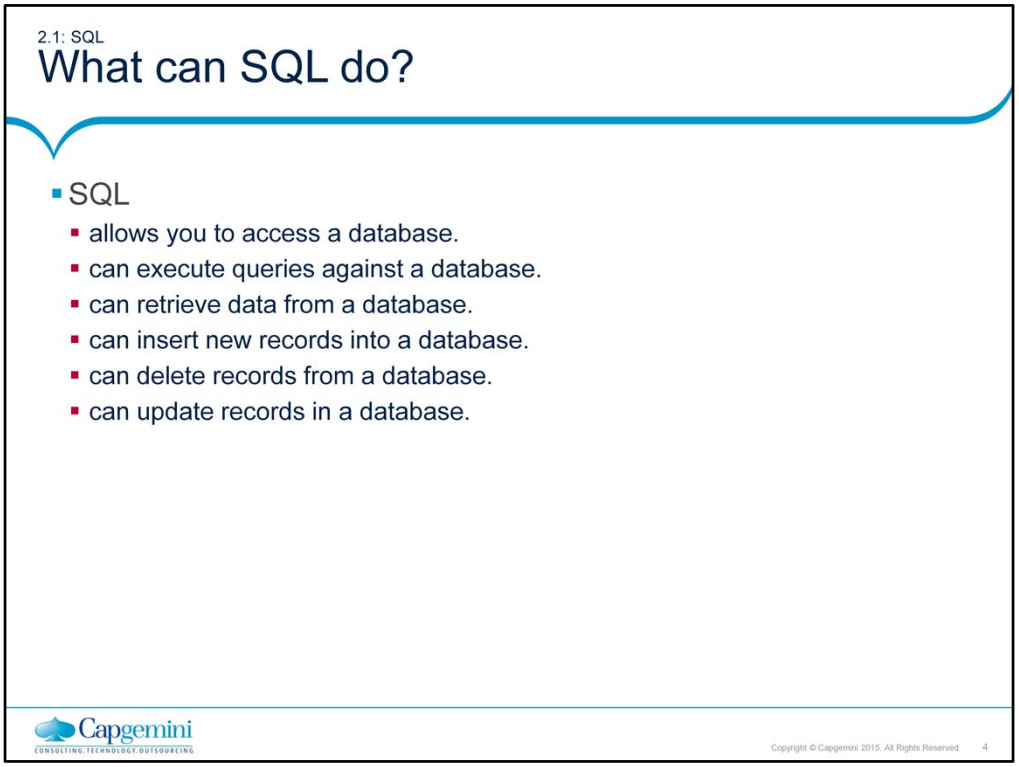
Copyright © Capgemini 2015. All Rights Reserved 2



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

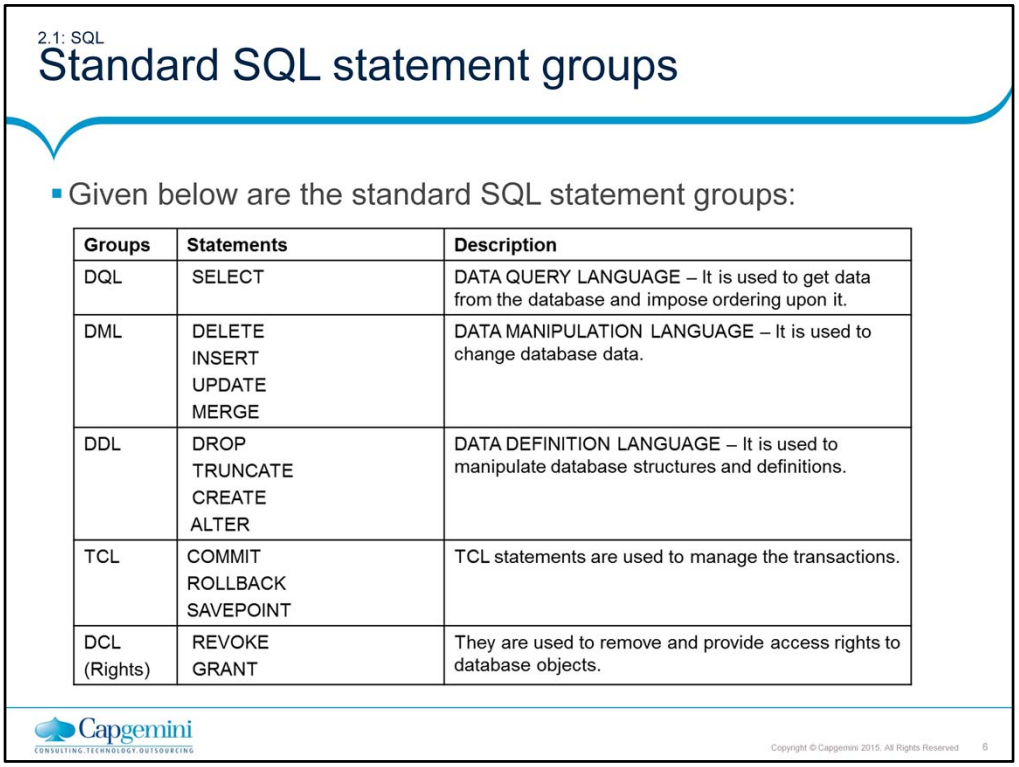
2.1: SQL

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

Copyright © Capgemini 2015. All Rights Reserved 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.

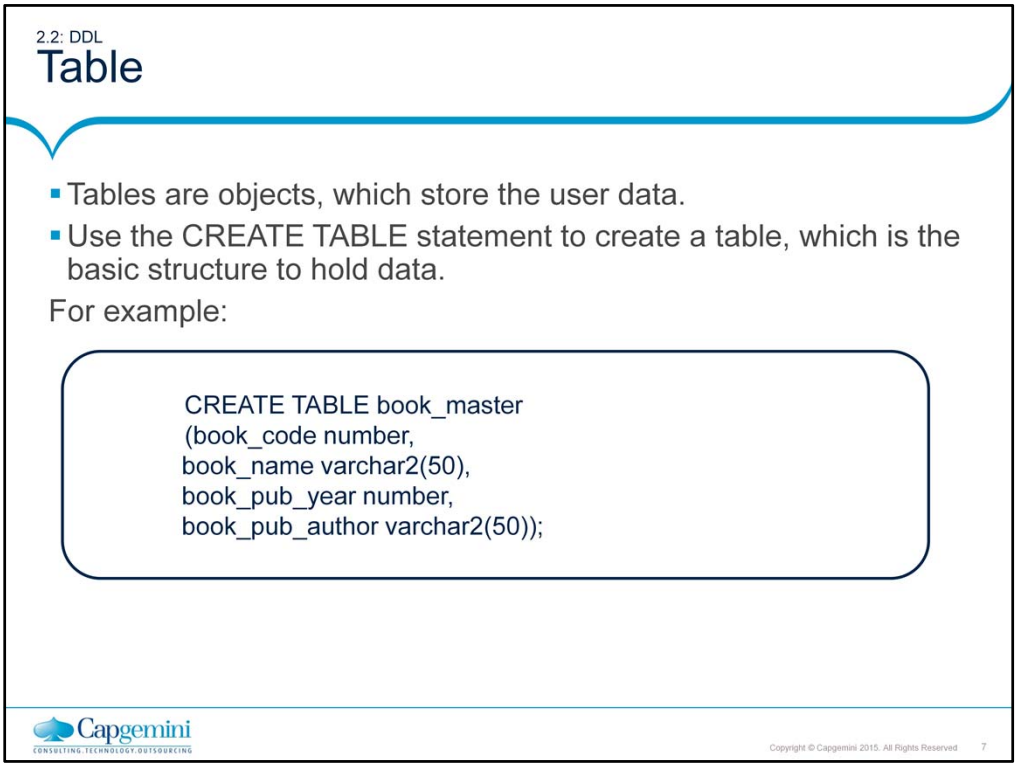
Standard SQL Statement groups:

•"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.

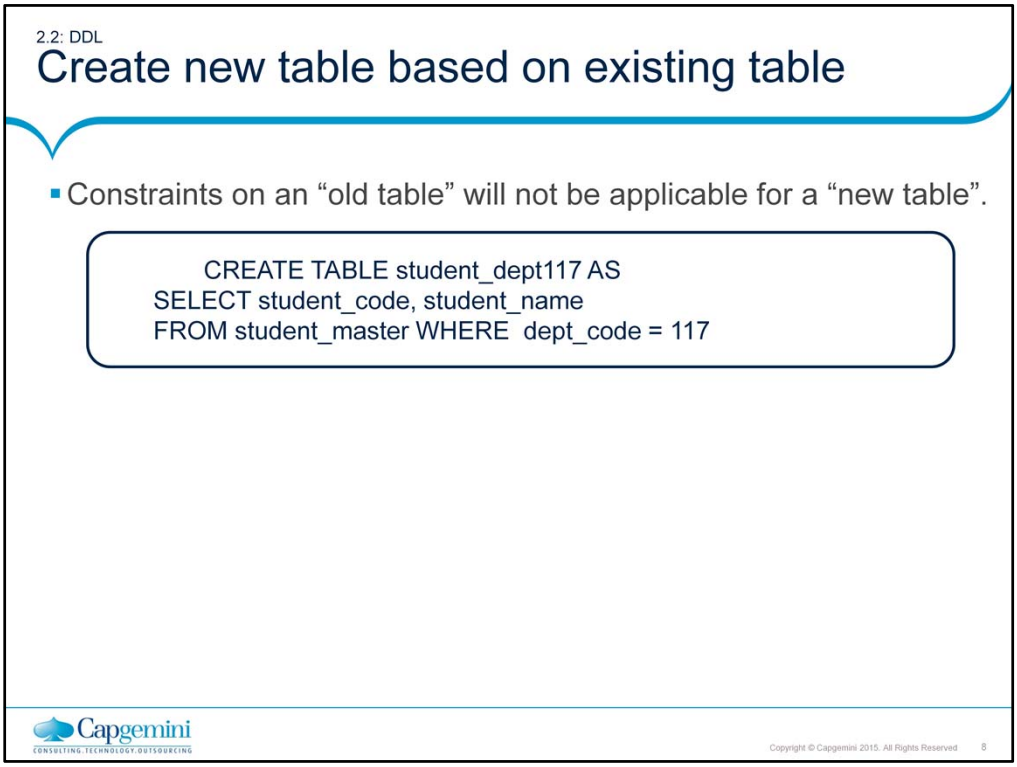
For example: You can create tables, add data to tables, or modify data, drop the table, set permissions for users, etc.



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:

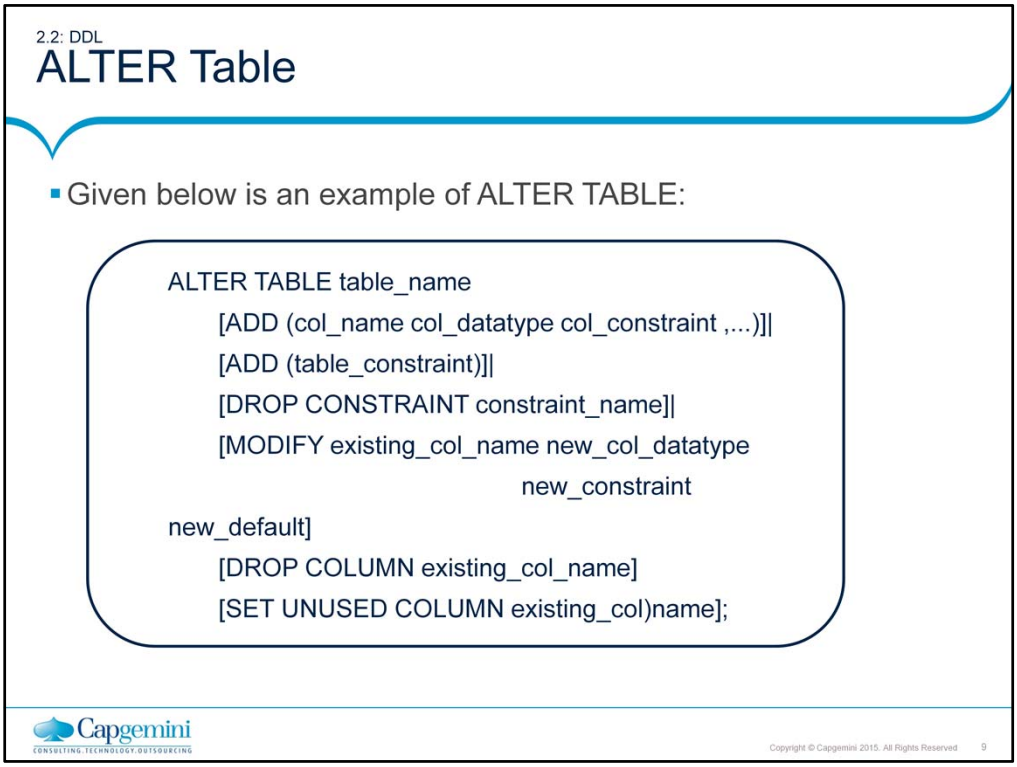
```
CREATE TABLE table_name
(
{col_name.col_datatype [[CONSTRAINT
const_name][col_constraint]]},...
[table_constraint],...
)
[AS query]
```

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.



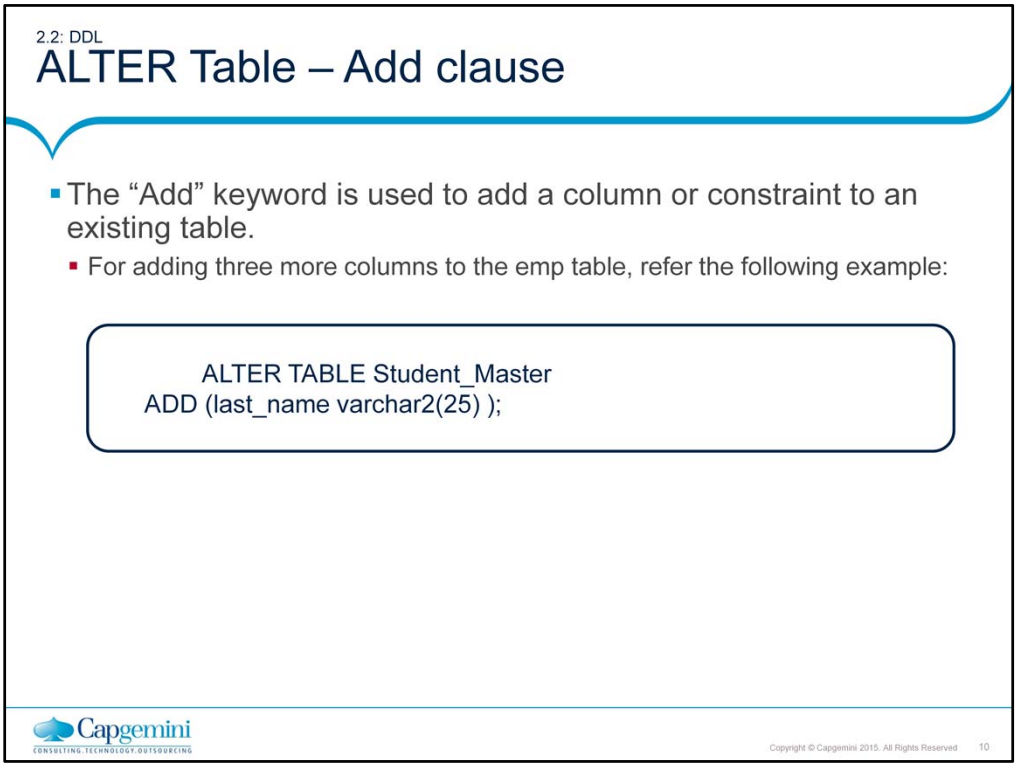
Creating new table based on an existing table:
The example in the slide shows how to create a new table based on an existing table.

When the new table will be created, it will contain student information from department 117.
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 cannot 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);
```

2.2: DDL

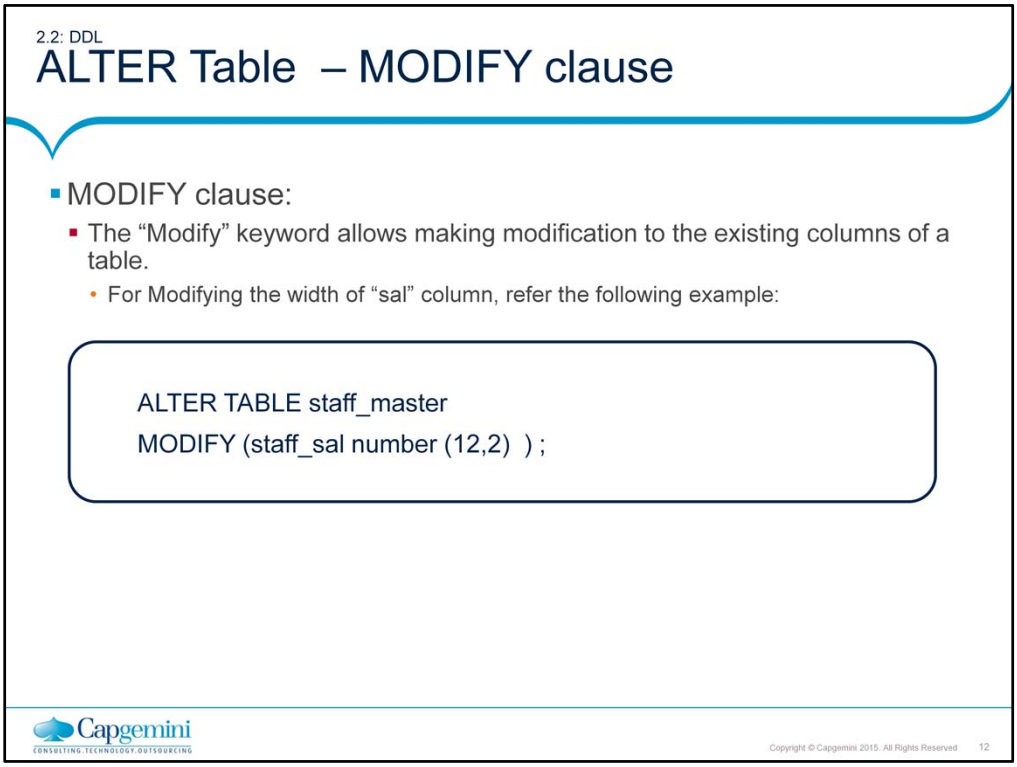
ALTER Table – Add clause

- 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

Copyright © Capgemini 2015. All Rights Reserved 11



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".

2.2: DDL

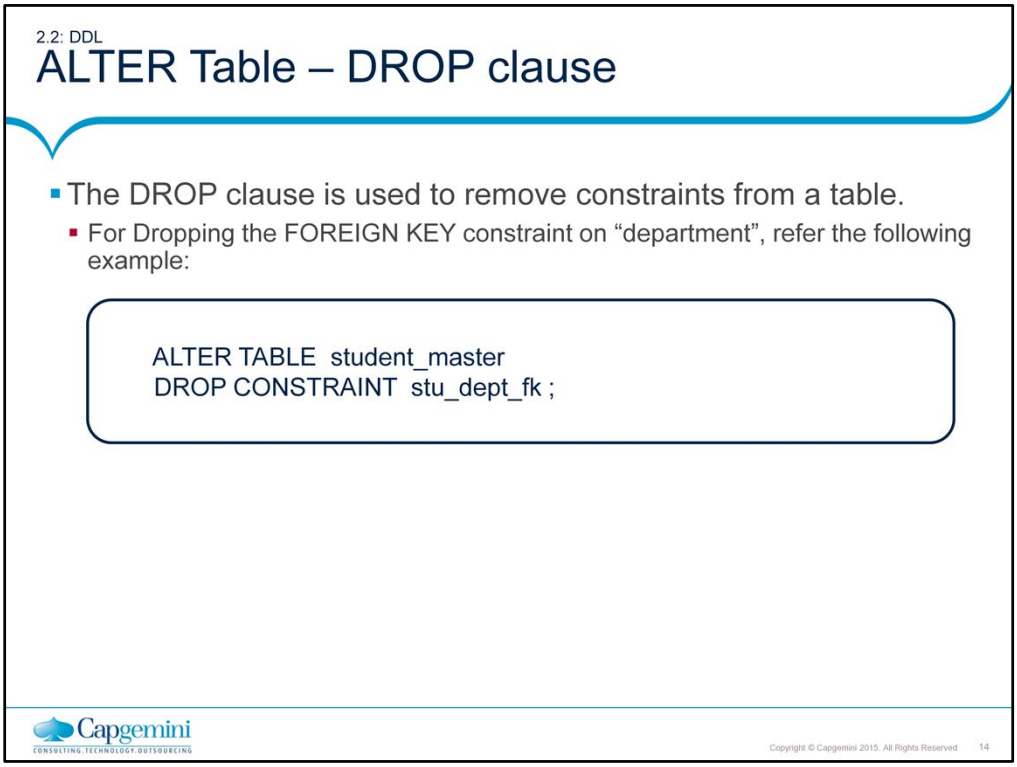
ALTER Table – Enable | Disable clause

- 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;
```

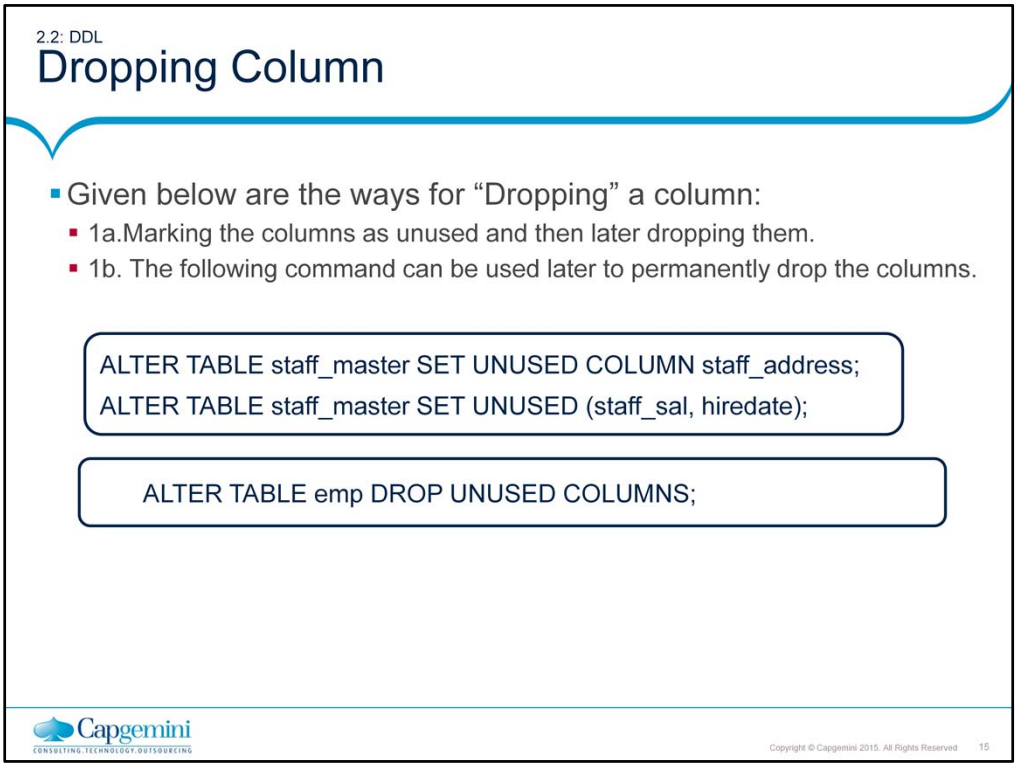
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 13



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:
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.

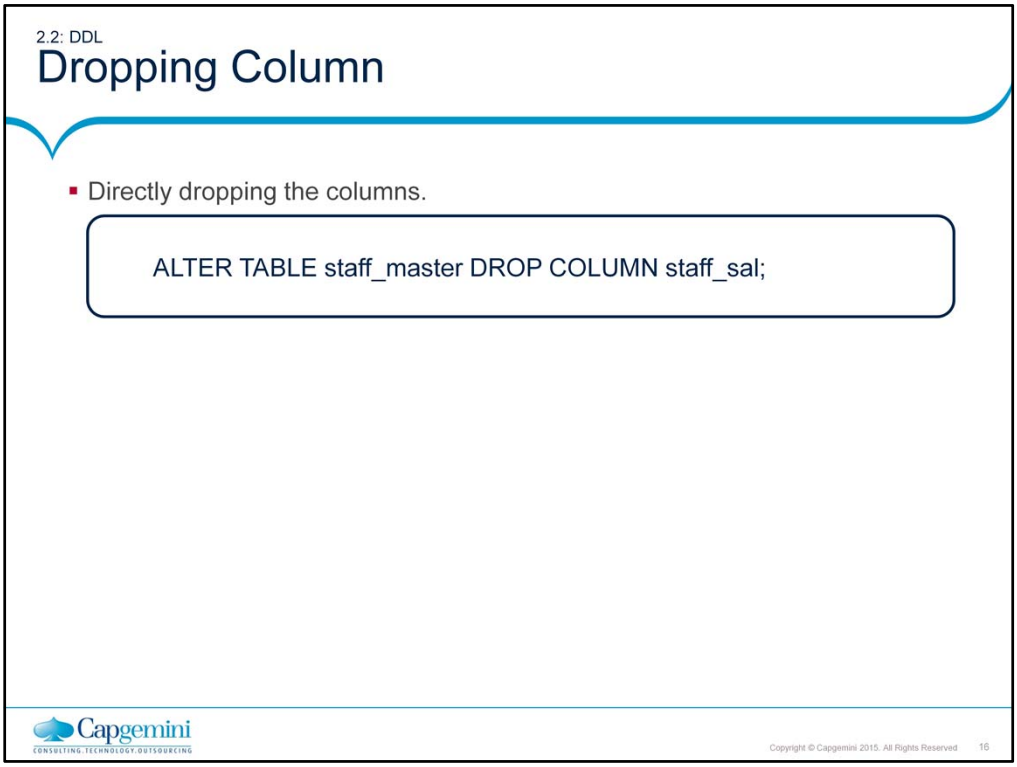
This feature drops the column from a table and releases any space back to the segment.

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.

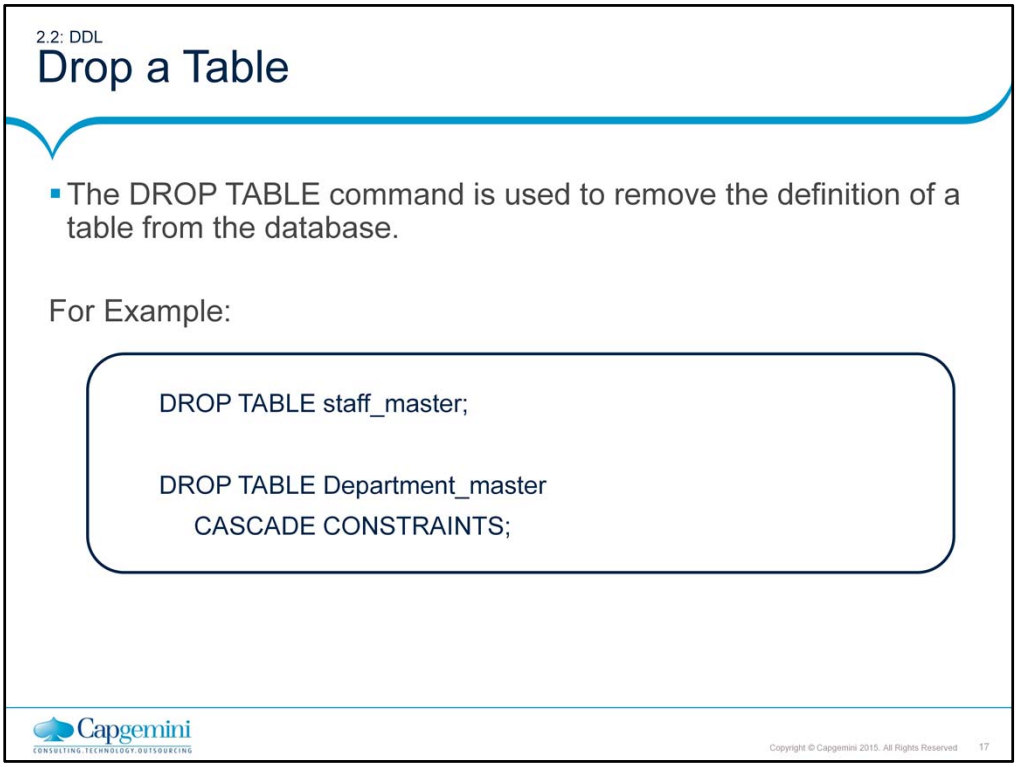
The advantage of the marking column as “unused” and then dropping them is that marking the columns is much faster process than dropping the columns.
You can refer to the data dictionary table USER_UNUSED_COL_TABS to get information regarding the tables with columns marked as unused.



Ways for “Dropping” a column (contd.):
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

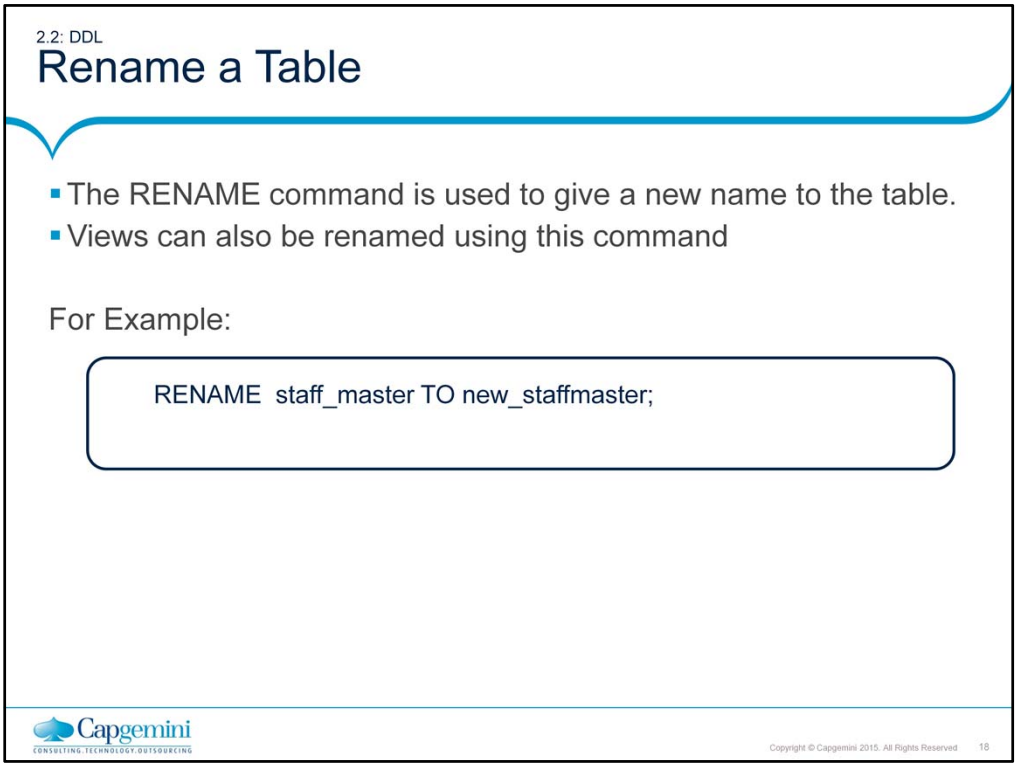


Deleting Database Objects: Tables
Deleting objects that exist in the database is an easy task. Just say:

```
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-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.

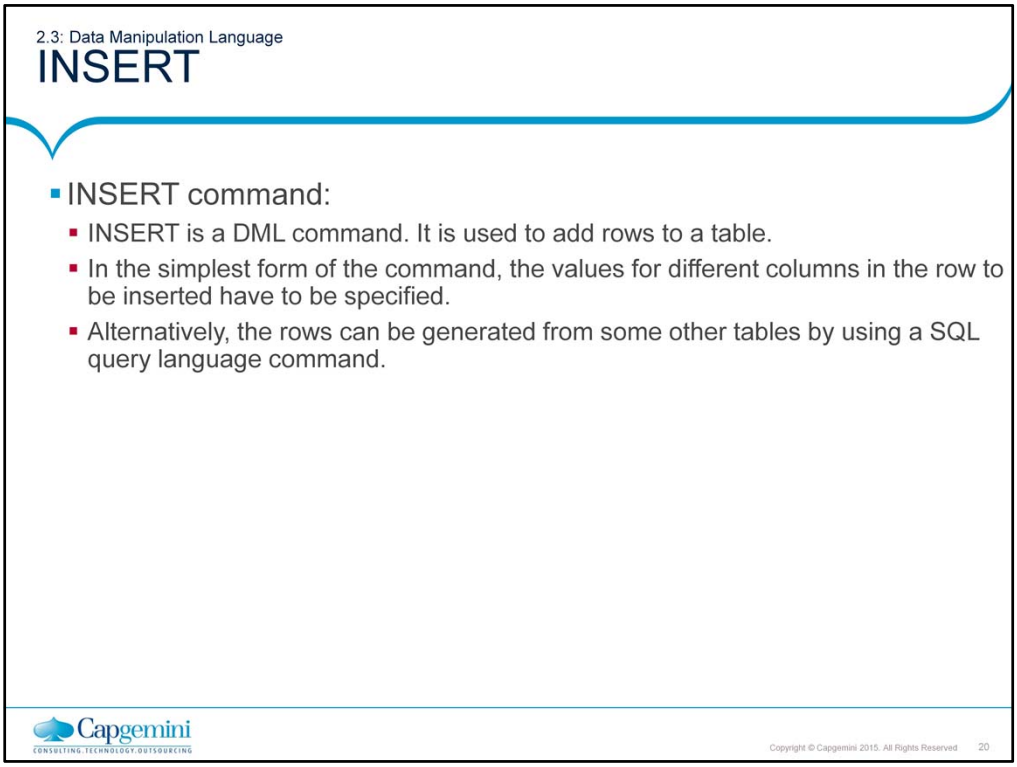
2.3: Data Manipulation Language

Concept of 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.

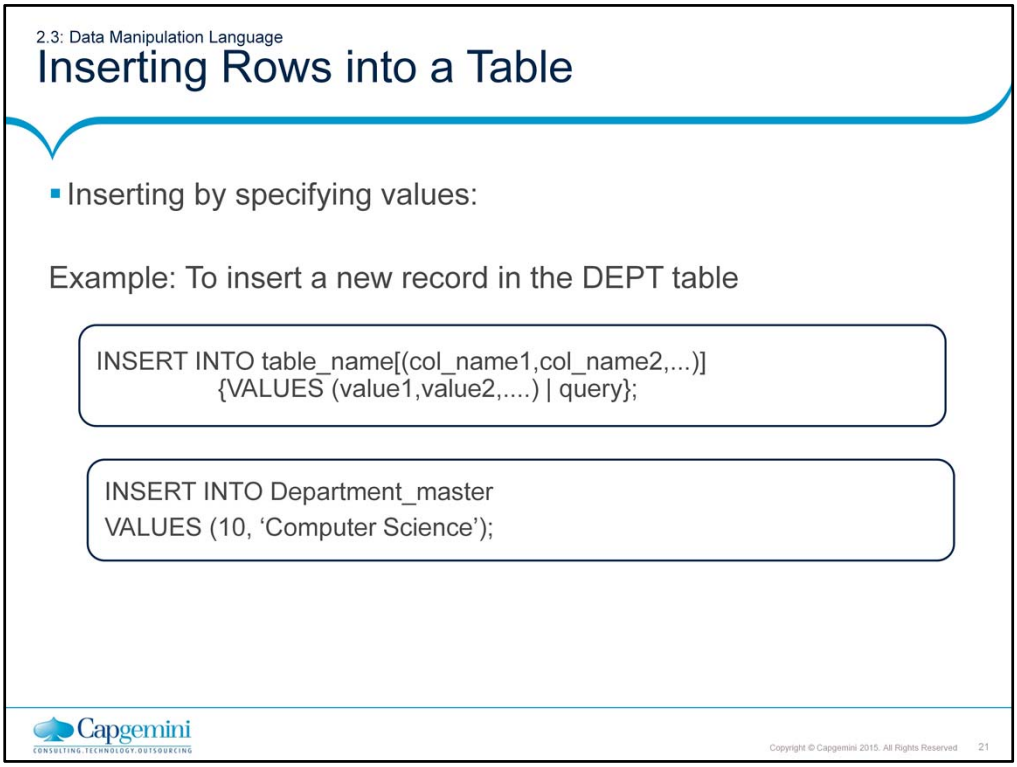
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 19



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:

The “view” cannot have a GROUP BY, CONNECT BY, START WITH, DISTINCT, UNION, INTERSECT, or MINUS clause or a join.
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.

2.3: Data Manipulation Language

Inserting Rows into a Table

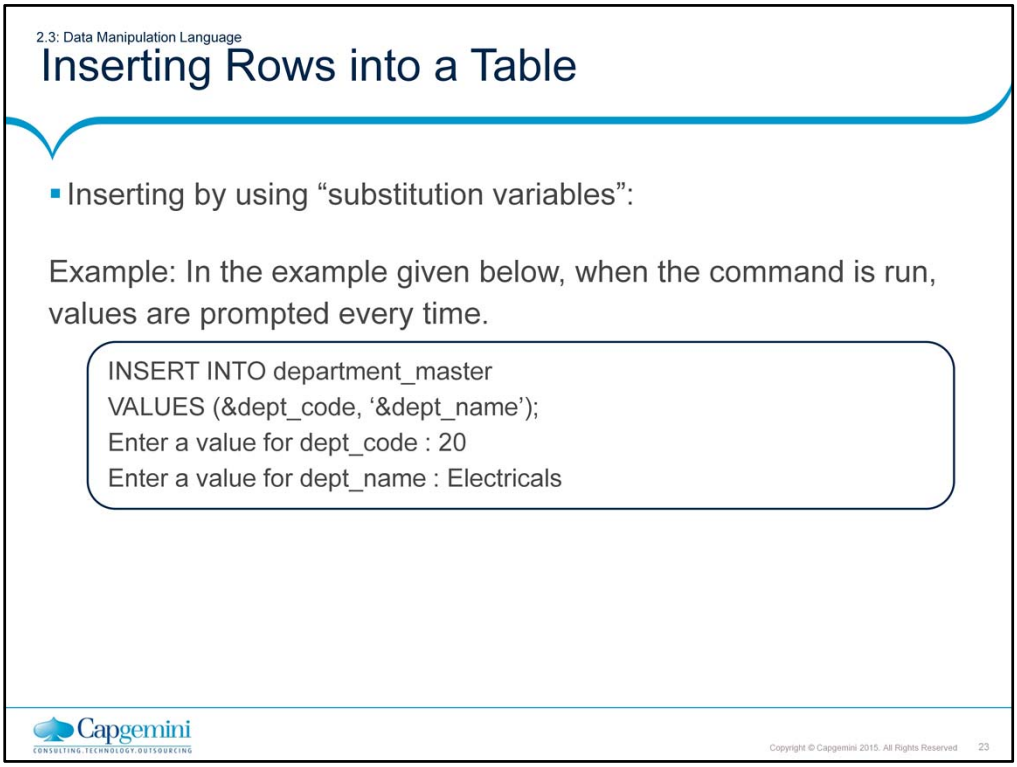
- Inserting rows in a table from another table using Subquery:

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

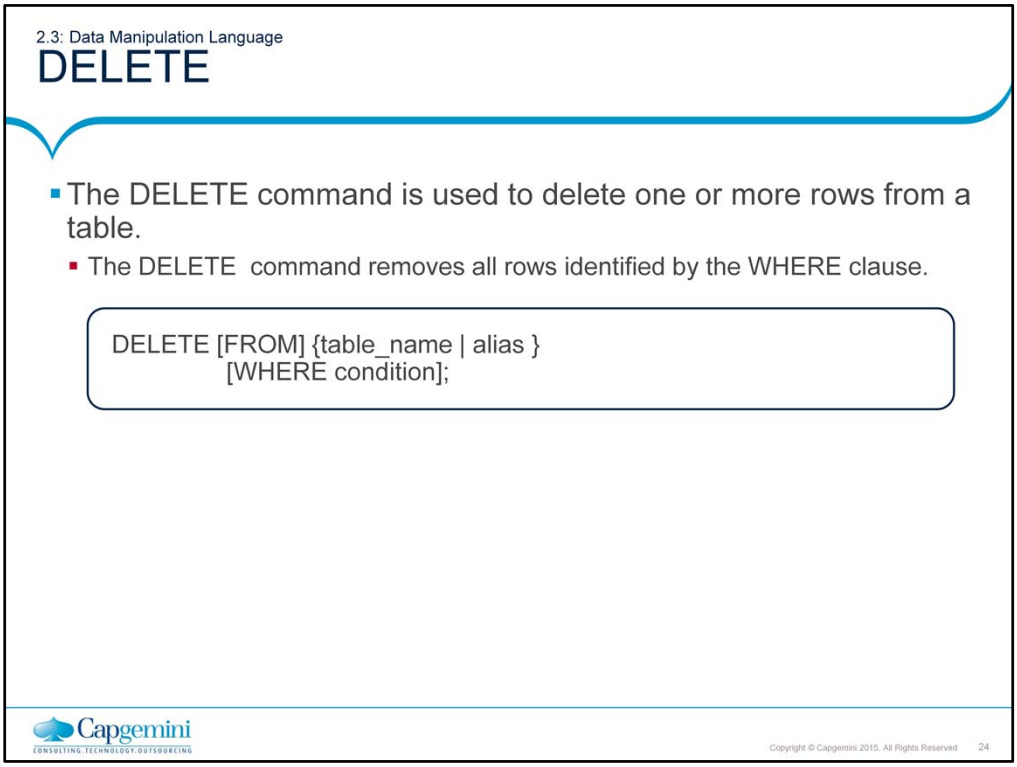
```
INSERT INTO new_staff_table
SELECT * FROM staff_master
WHERE staff_master.hiredate > '01-jan-82';
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

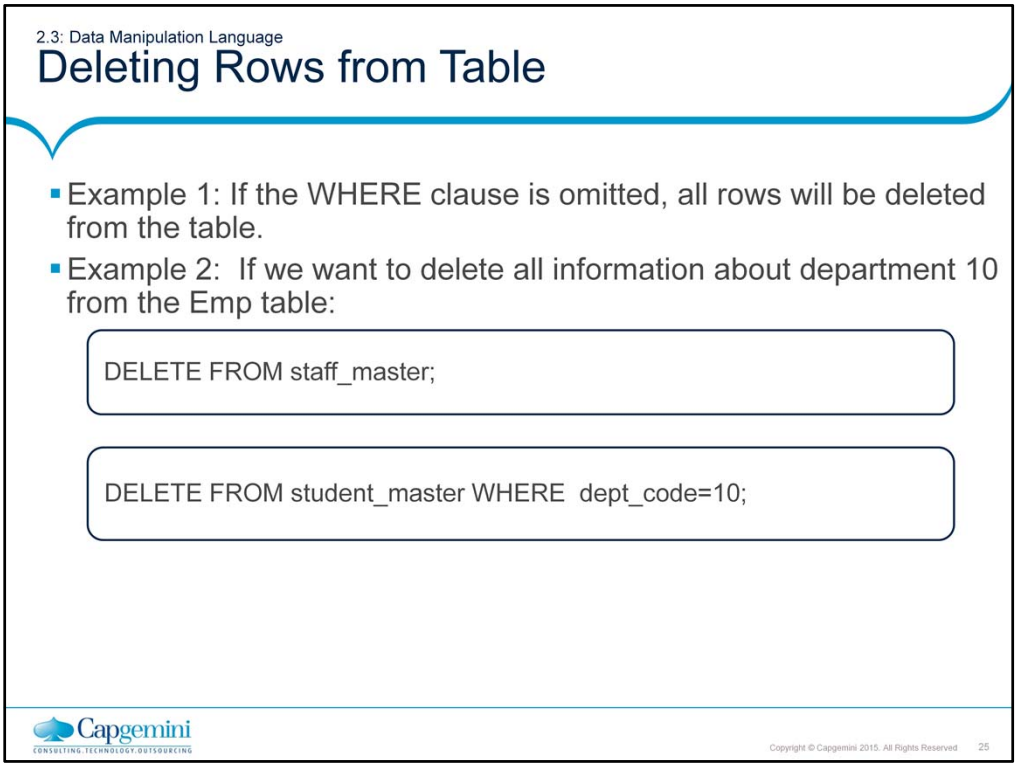
Copyright © Capgemini 2015. All Rights Reserved 22



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.

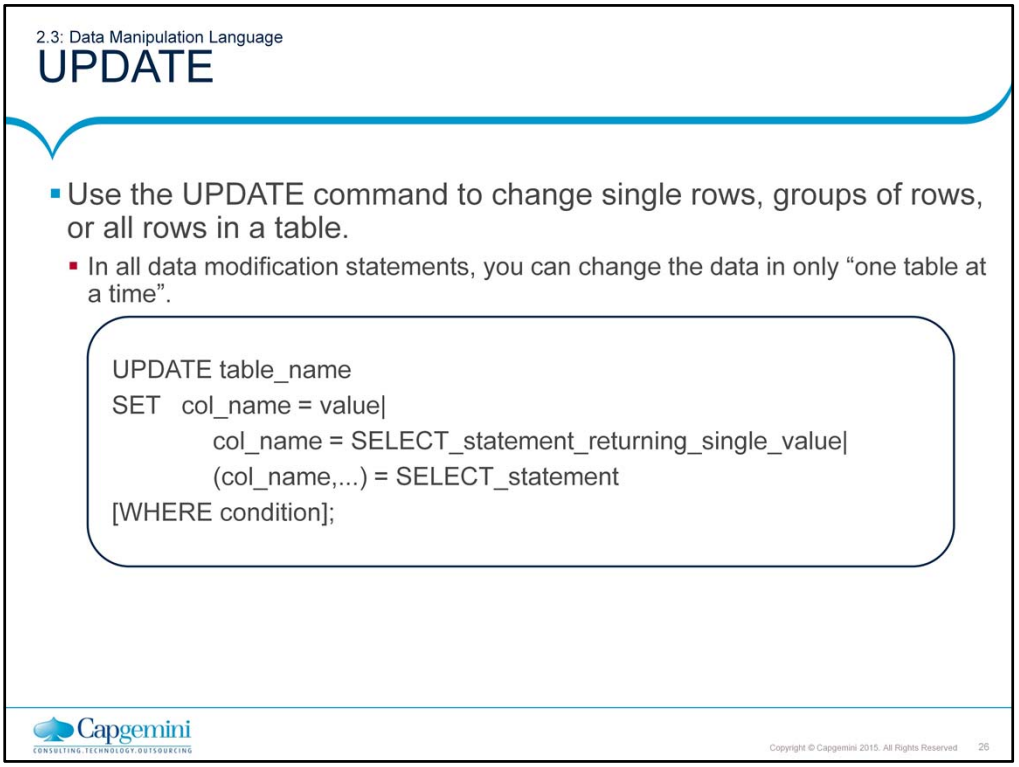


Deletion of Data from Tables
The table_name can be a “table” or a “view”.
The DELETE command is used to delete one or more rows from a table.
The DELETE statement removes all rows identified by the WHERE clause.
This is another DML, which means we can rollback the deleted data, and that to make our changes 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.

2.3: Data Manipulation Language

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

Copyright © Capgemini 2015. All Rights Reserved 27

2.3: Data Manipulation Language

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

Copyright © Capgemini 2015. All Rights Reserved 28

2.3: Data Manipulation Language

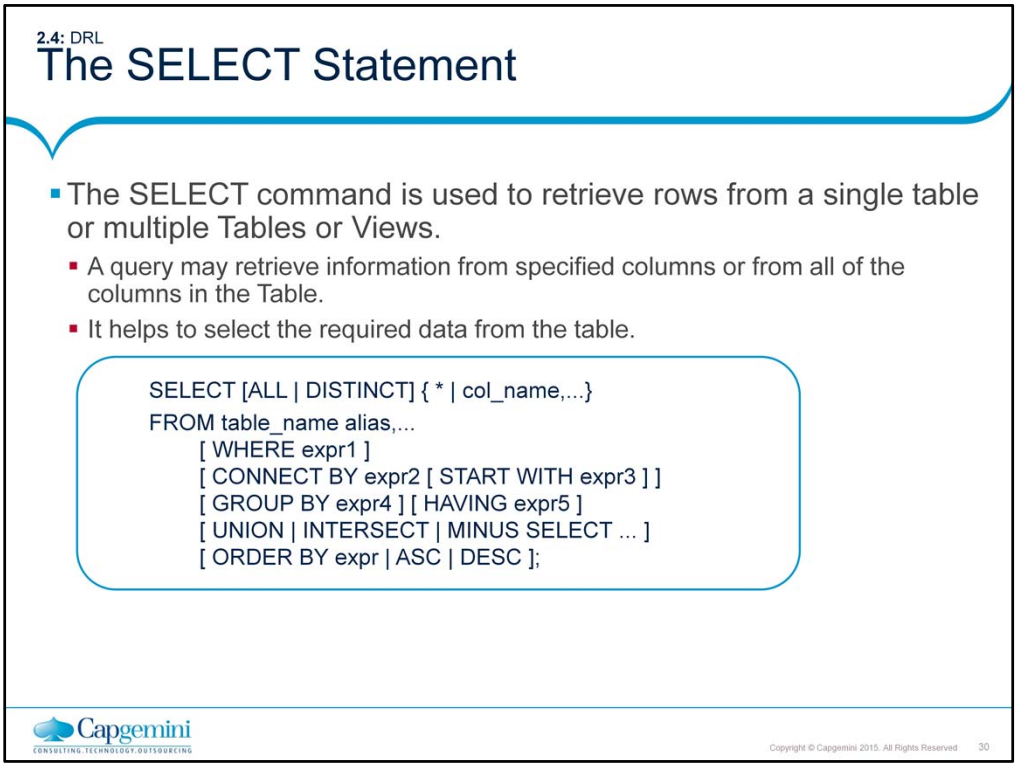
Using a Subquery to do an Update

- 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

Copyright © Capgemini 2015. All Rights Reserved 29



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, UNION are optional.
All the examples that follow are based on EMP and DEPT tables that are already available.

2.4: DRL

Selecting Columns

- Displays all the columns from the student_master table

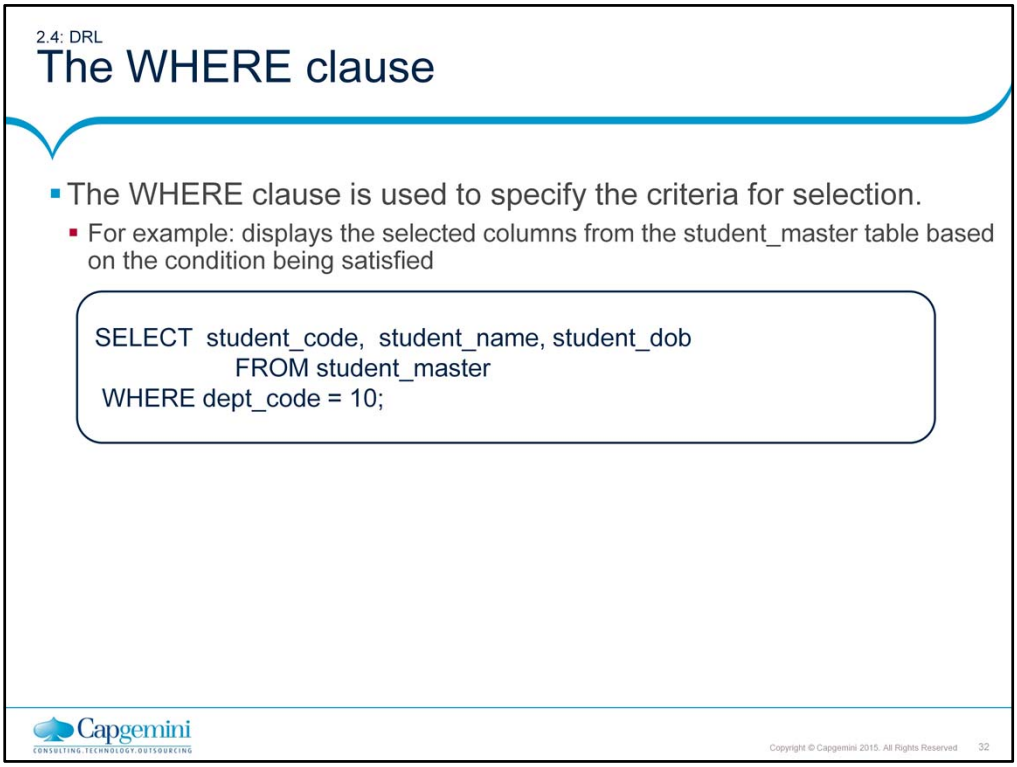
```
SELECT  *  FROM student_master;
```

- 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 31



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:

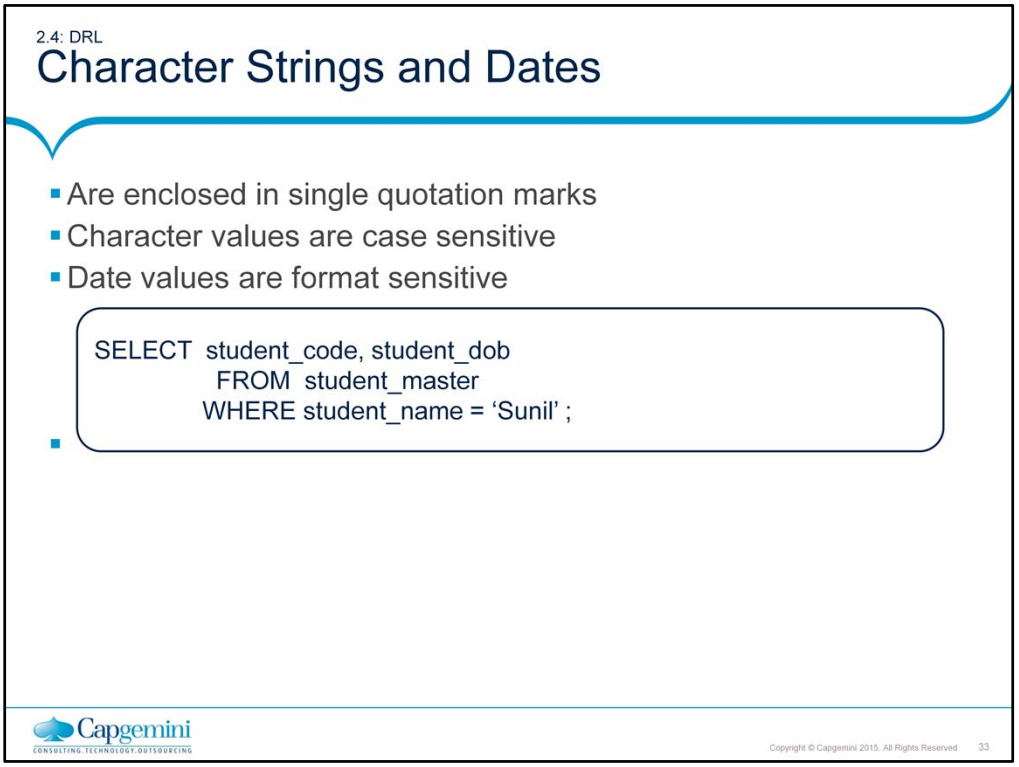
Comparison Predicates:
The Comparison Predicates specify the comparison of two values.
It is of the form:

< Expression> < operator > < Expression>
< Expression> <operator> <subquery>

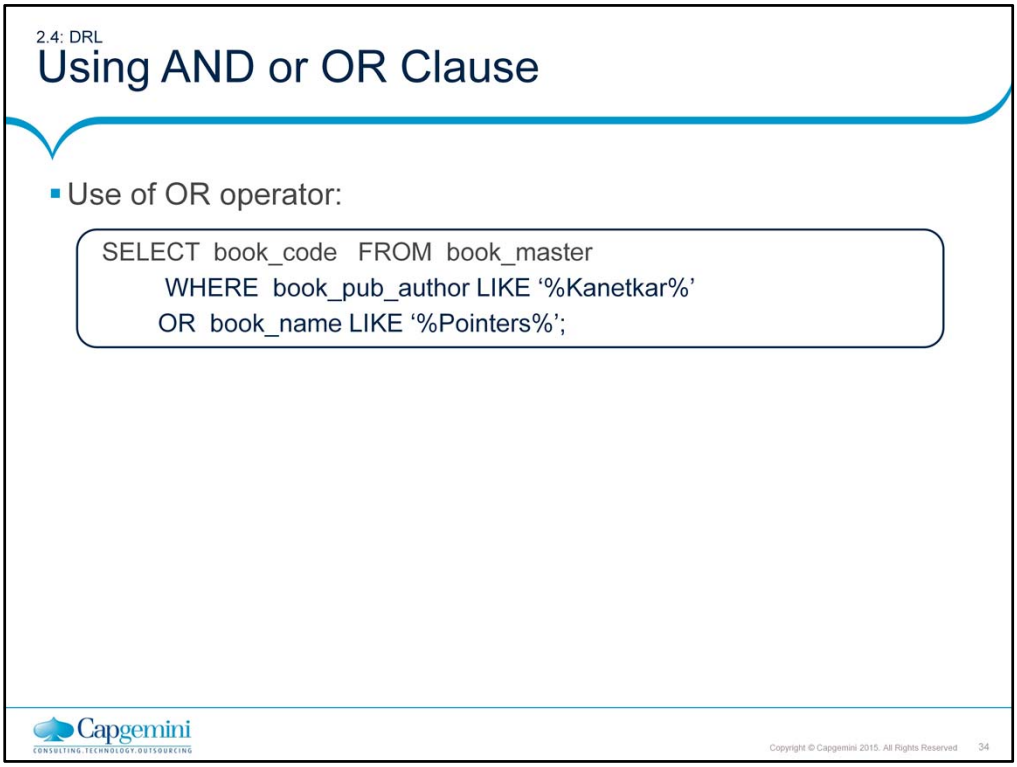
The operators used are shown on the next slide:

```
WHERE <search condition>
```

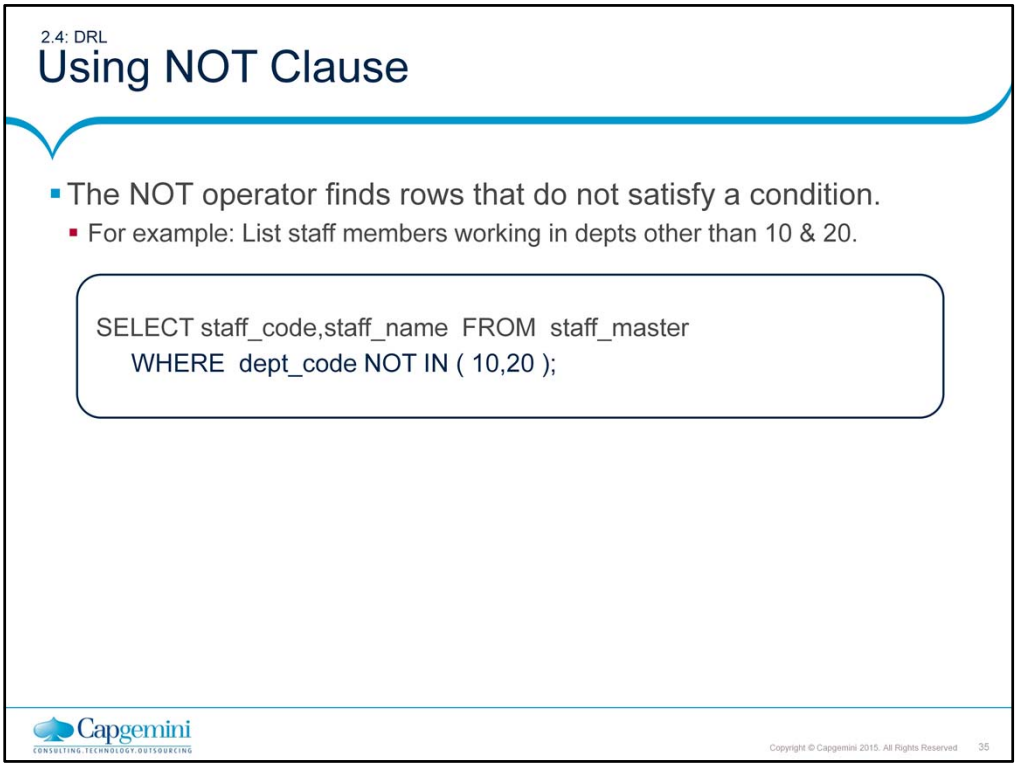
contd.



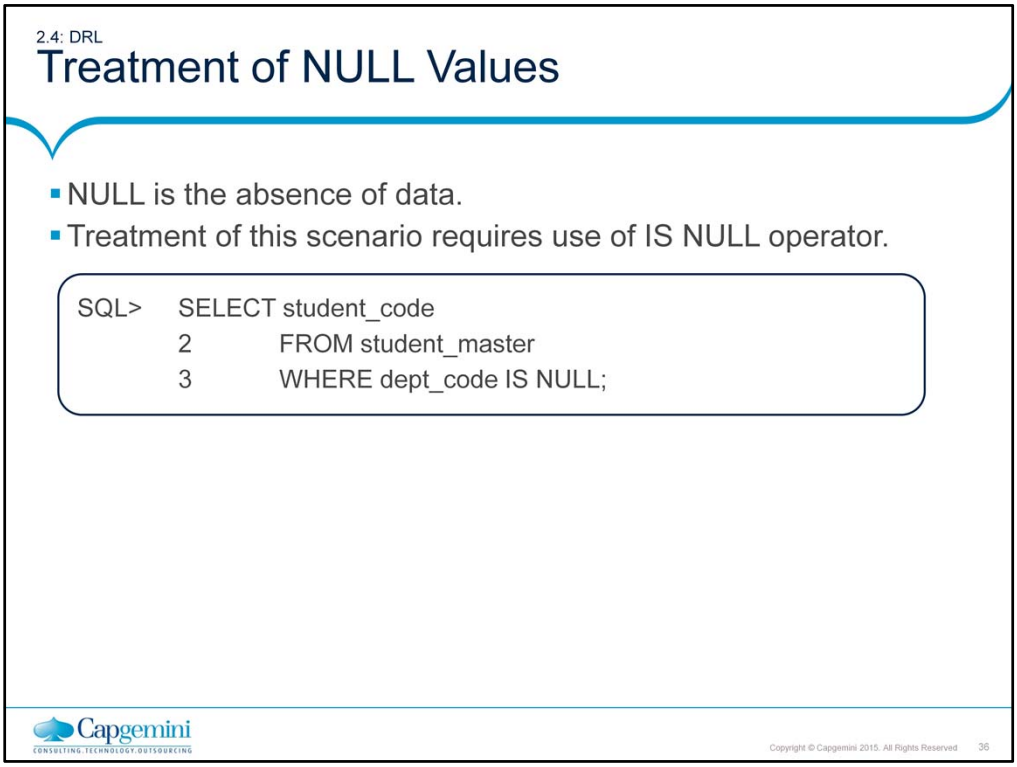
Oracle Database store dates in an internal numeric format, representing the century, year, month, day, hours, minutes, and seconds.
The date datatype is covered in detail later.



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).

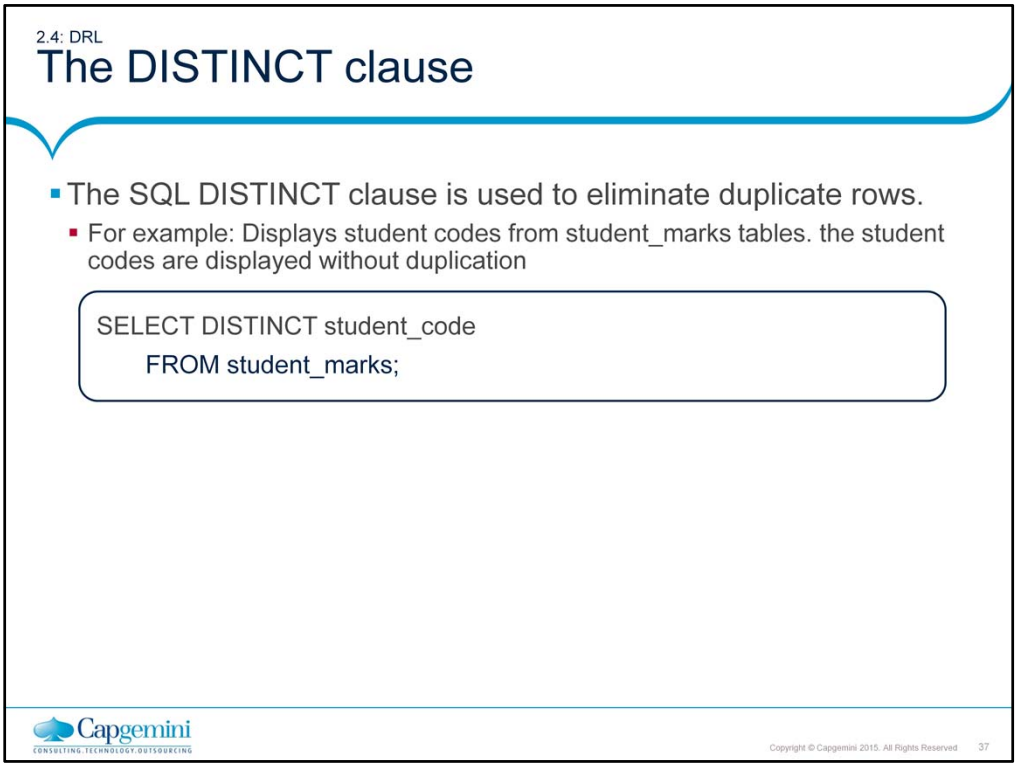


Note: NOT is a negation operator.



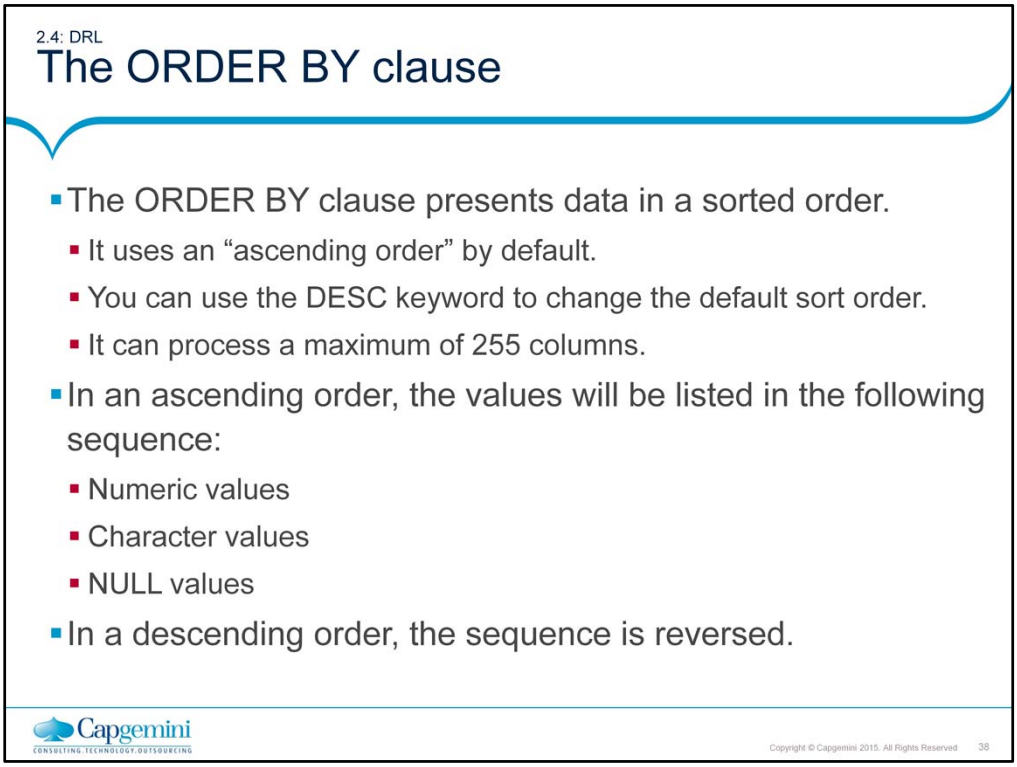
NULL predicate:
The NULL predicate specifies a test for NULL values. The form for NULL predicate is:

< 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.



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.

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.



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.

2.4: DRL

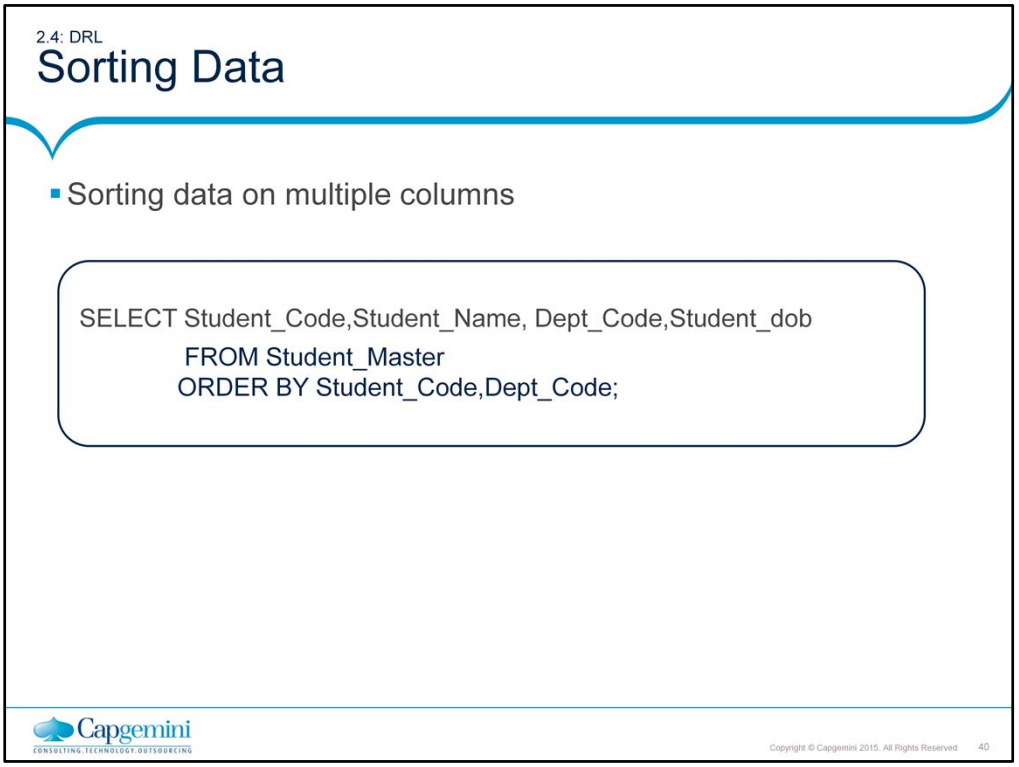
Sorting Data

- 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 ;
```

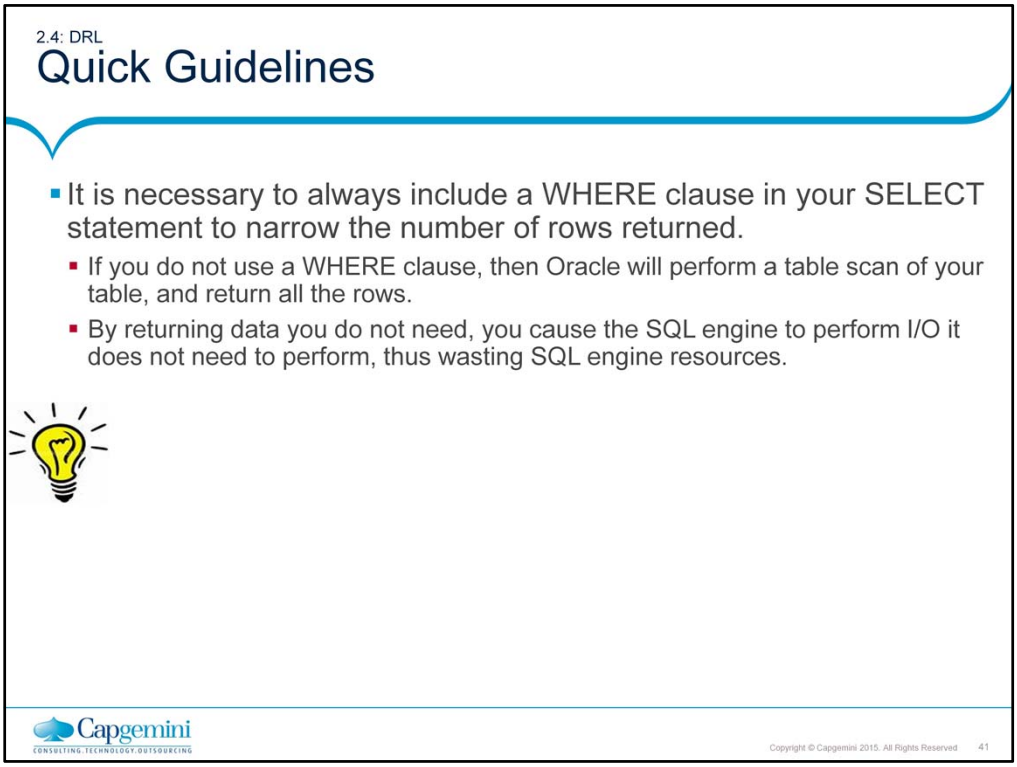
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 39



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

Copyright © Capgemini 2015. All Rights Reserved 42

Quick Guidelines

- Carefully evaluate whether the SELECT query requires the DISTINCT clause or not.
  - The DISTINCT clause should only be used in SELECT statements.
    - This is mandatory if you know that “duplicate” returned rows are a possibility, and that having duplicate rows in the result set would cause problems with your application.
  - The DISTINCT clause creates a lot of extra work for SQL Server.
    - The extra load reduces the “physical resources” that other SQL statements have at their disposal.
  - Hence, use the DISTINCT clause only if it is necessary.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 43

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.

Quick Guidelines

- If you use LIKE in your WHERE clause, try to use one or more leading character in the clause, if at all possible.
  - **For example**: Use LIKE 'm%' not LIKE '%m'
- Certain operators in the WHERE clause prevents the query optimizer from using an Index to perform a search.
  - **For example**: "IS NULL", "<>", "!=", "!>", "!<", "NOT", "NOT EXISTS", "NOT IN", "NOT LIKE", and "LIKE '%500'"

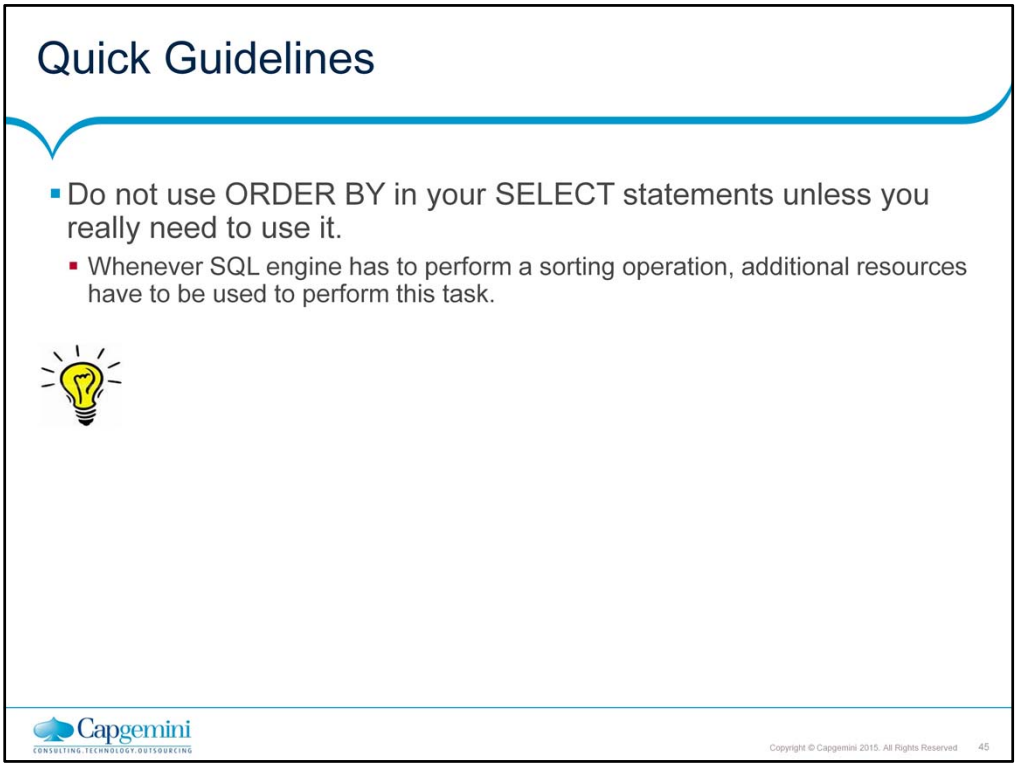
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 44

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.):
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.

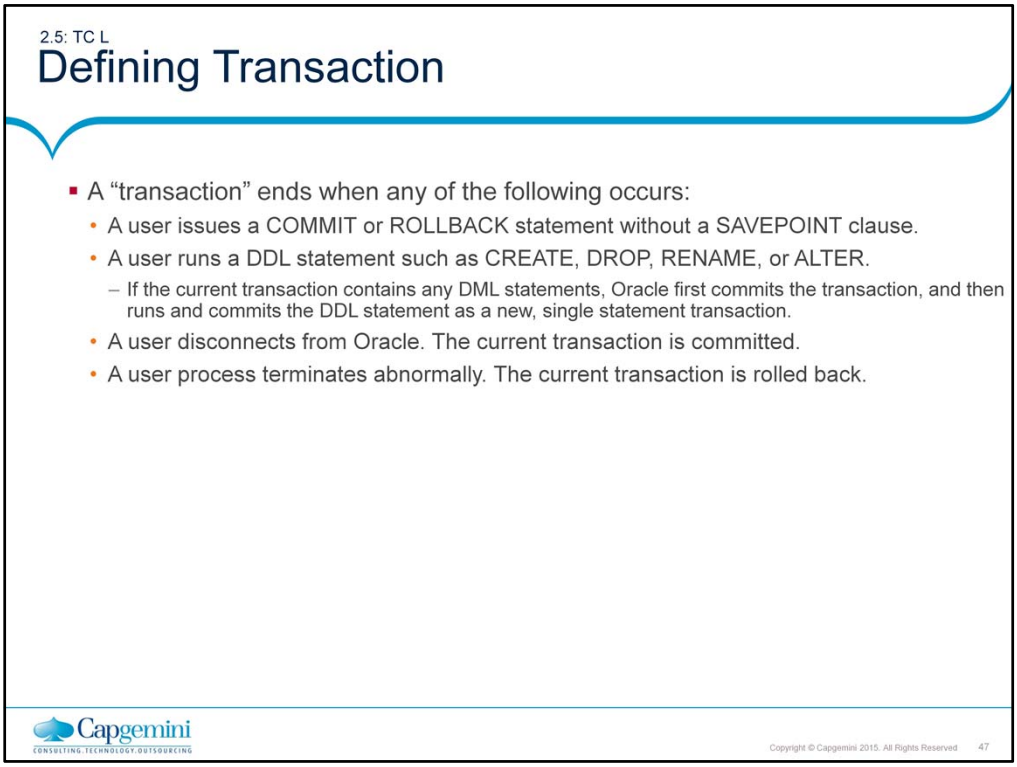
2.5: TC L

Defining Transaction

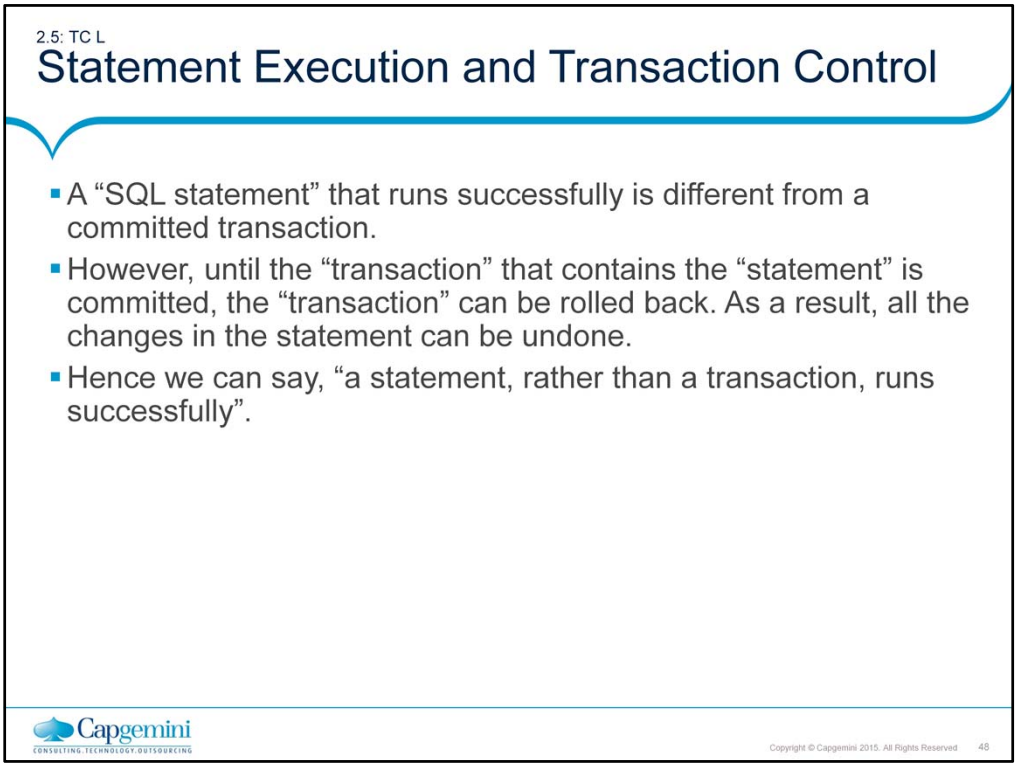
- A “transaction” is a logical unit of work that contains one or more SQL statements.
  - “Transaction” is an atomic unit.
  - The effects of all the SQL statements in a transaction can be either:
    - all committed (applied to the database), or
    - all rolled back (undone from the database)
  - A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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 46



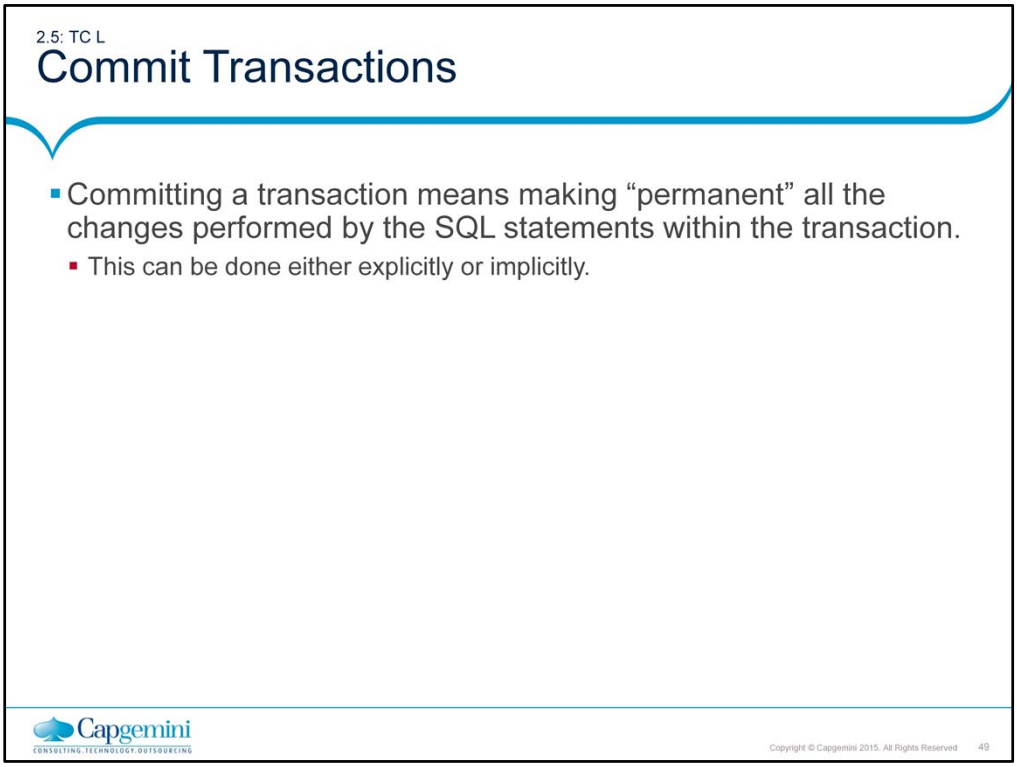
Note:
After one transaction ends, the next executable SQL statement automatically starts the subsequent transaction.



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.

2.5: TC L

Commit Transactions

- COMMIT statement makes “permanent” all the changes that are performed in the current transaction.
- Syntax:

```
COMMIT [WORK];
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 51

2.5: TC L

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

Copyright © Capgemini 2015. All Rights Reserved 52

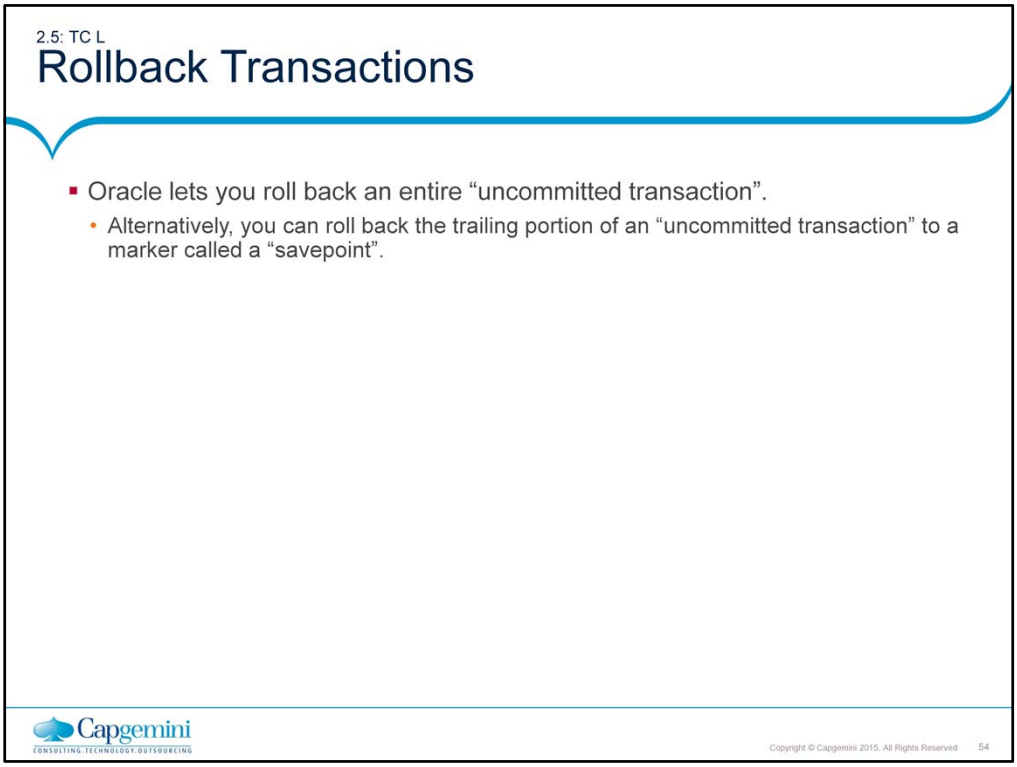
2.5: TC L

Rollback Transactions

- Rolling back a transaction means “undoing changes” to data that have been performed by SQL statements within an “uncommitted transaction”.
  - Oracle uses “undo tablespaces” (or rollback segments) to store old values.
  - Oracle also uses the “redo log” that contains a record of changes.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 53



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:

- Oracle undoes all changes made by all the SQL statements in the transaction by using the corresponding undo tablespace.
- Oracle releases all the locks of data for the transaction.
- The transaction ends.

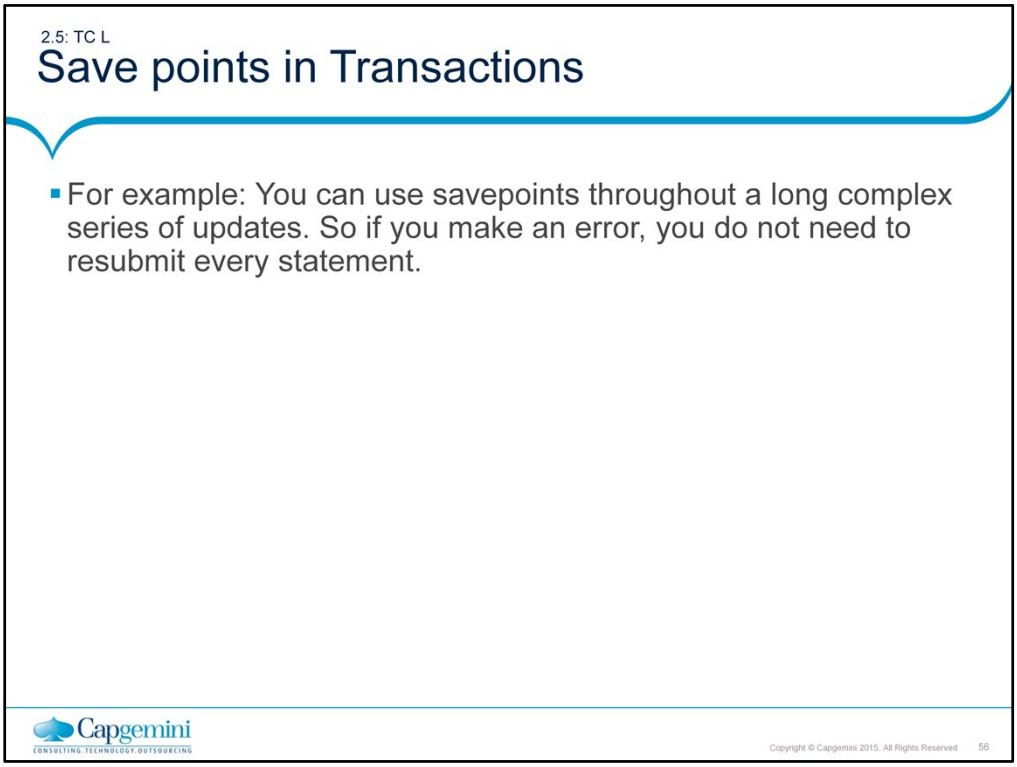
2.5: TC L

Save points in Transactions

- In a transaction, you can declare intermediate markers called “savepoints” within the context of a transaction.
  - By using “savepoints”, you can arbitrarily mark your work at any point within a long transaction.
  - In this manner, you can keep an option that is available later to roll back the work performed, however:
    - before the current point in the transaction, and
    - after a declared savepoint within the transaction

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 55



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.

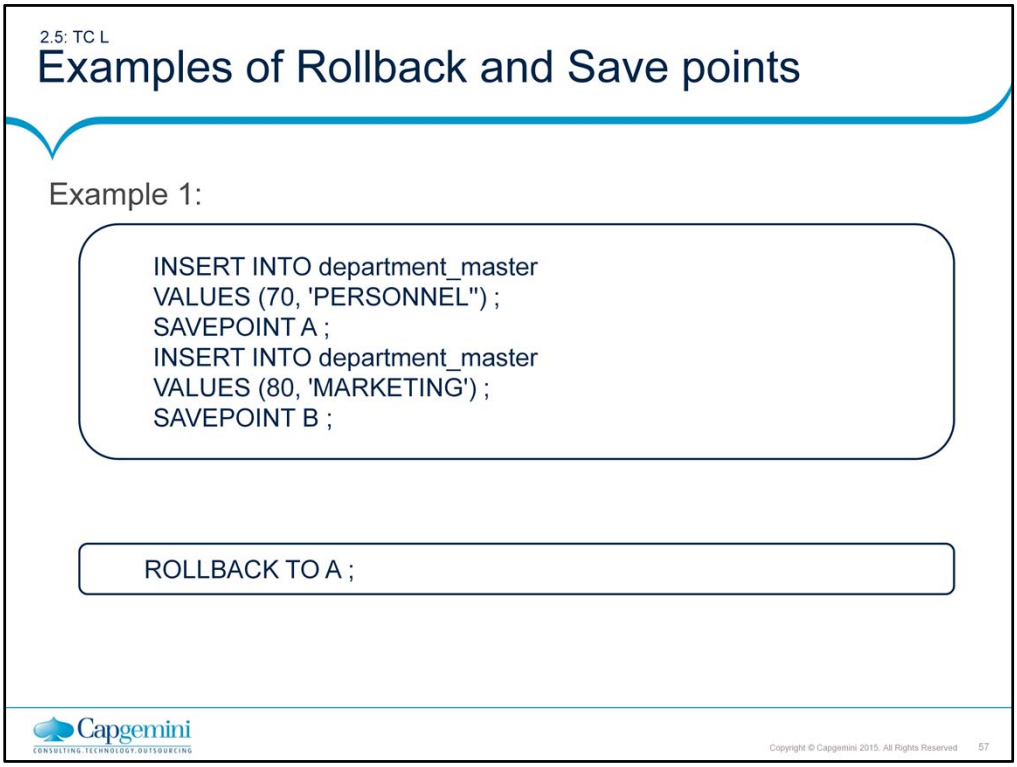
When a transaction is rolled back to a savepoint, there is an occurrence of the following sequence:

Oracle rolls back only the statements run after the savepoint.

Oracle preserves the specified savepoint, but all savepoints that were established after the specified savepoint are lost.

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

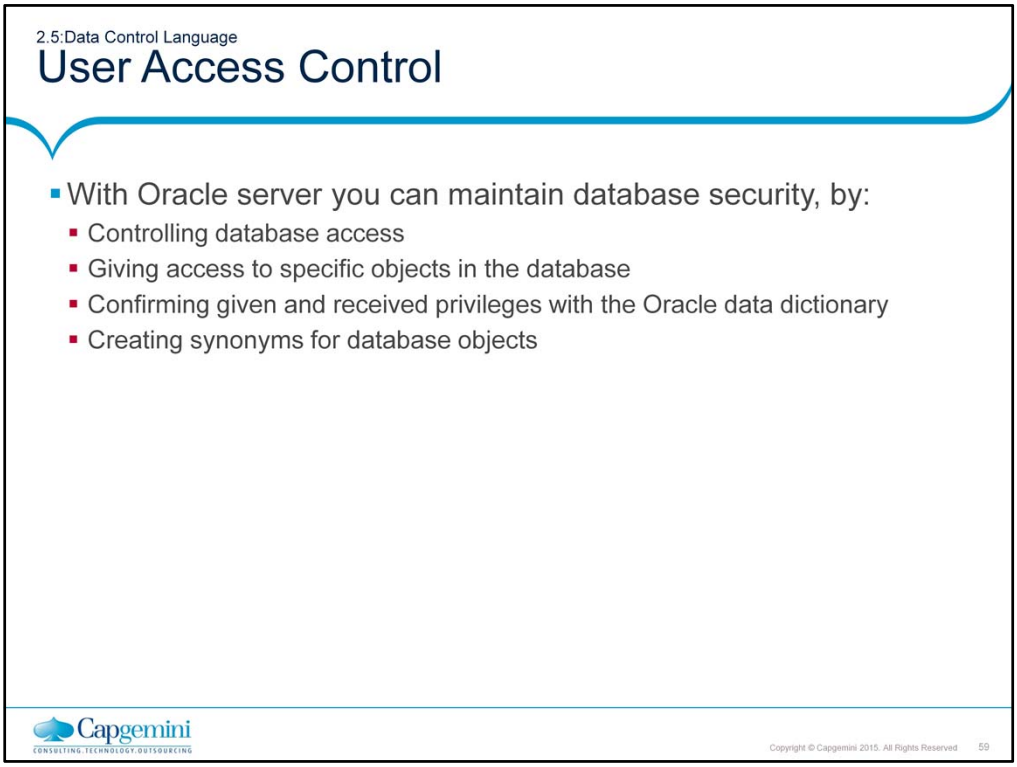
Syntax:

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

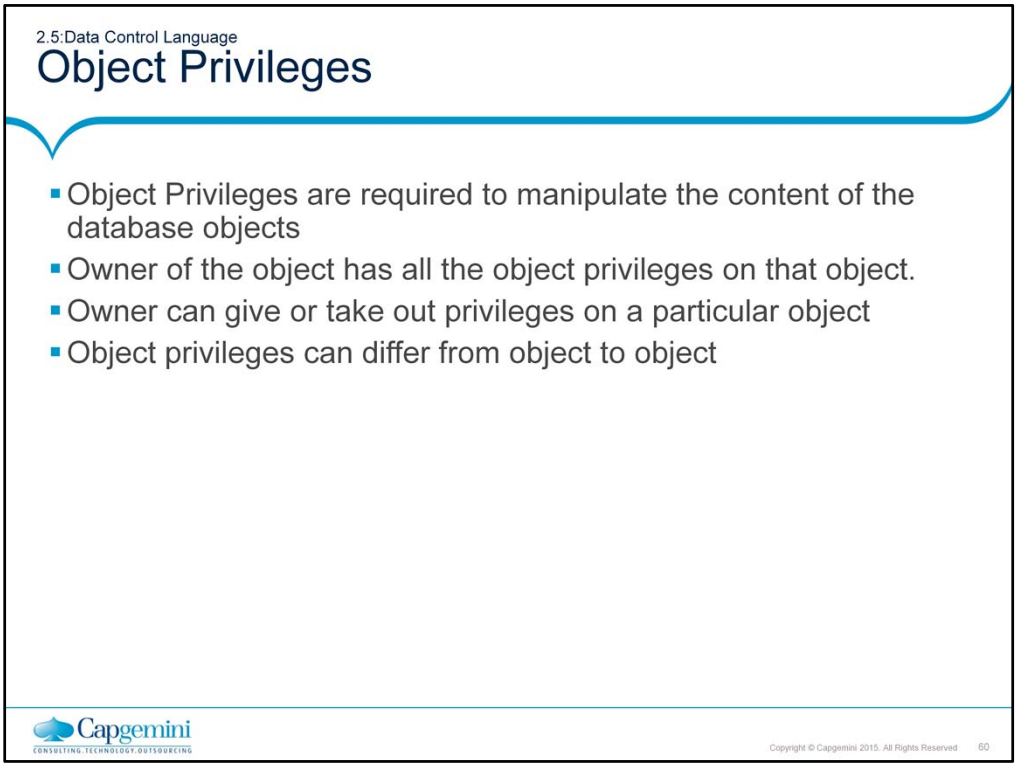
Example of rolling back changes to a Marker:
- Create a marker in a current transaction by using the SAVEPOINT statement.
- Roll back to that marker by using the ROLLBACK TO SAVEPOINT statement.

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

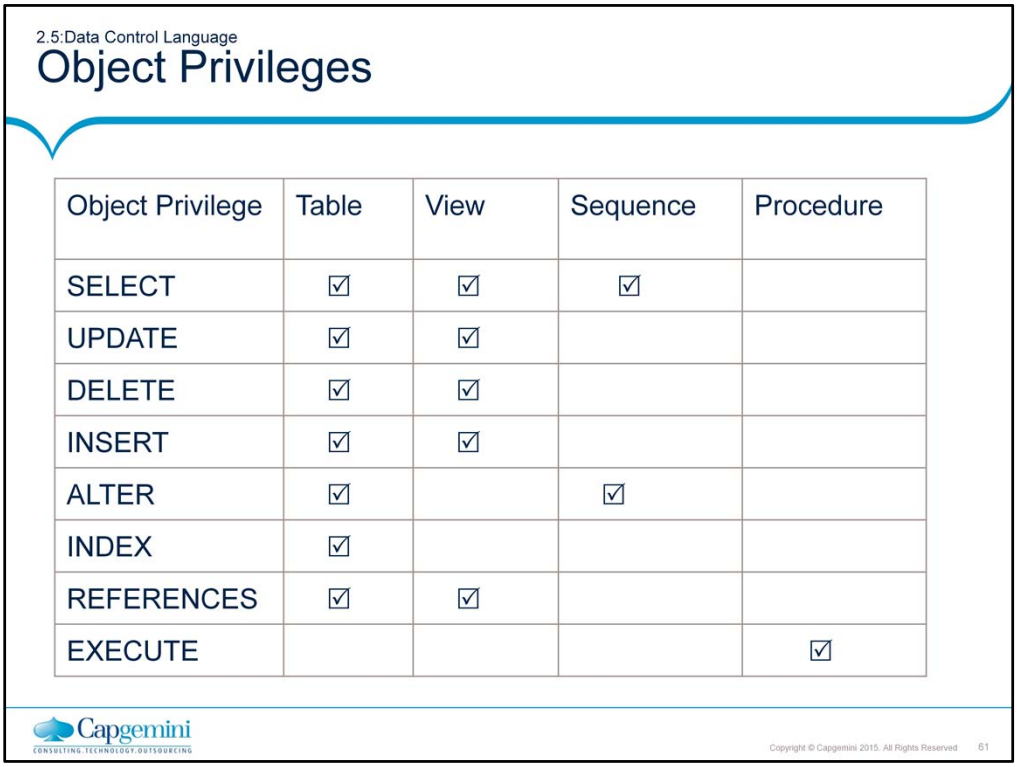
```
UPDATE...
SAVEPOINT update_done;
Savepoint created.
INSERT...
ROLLBACK TO update_done;
Rollback complete.
```



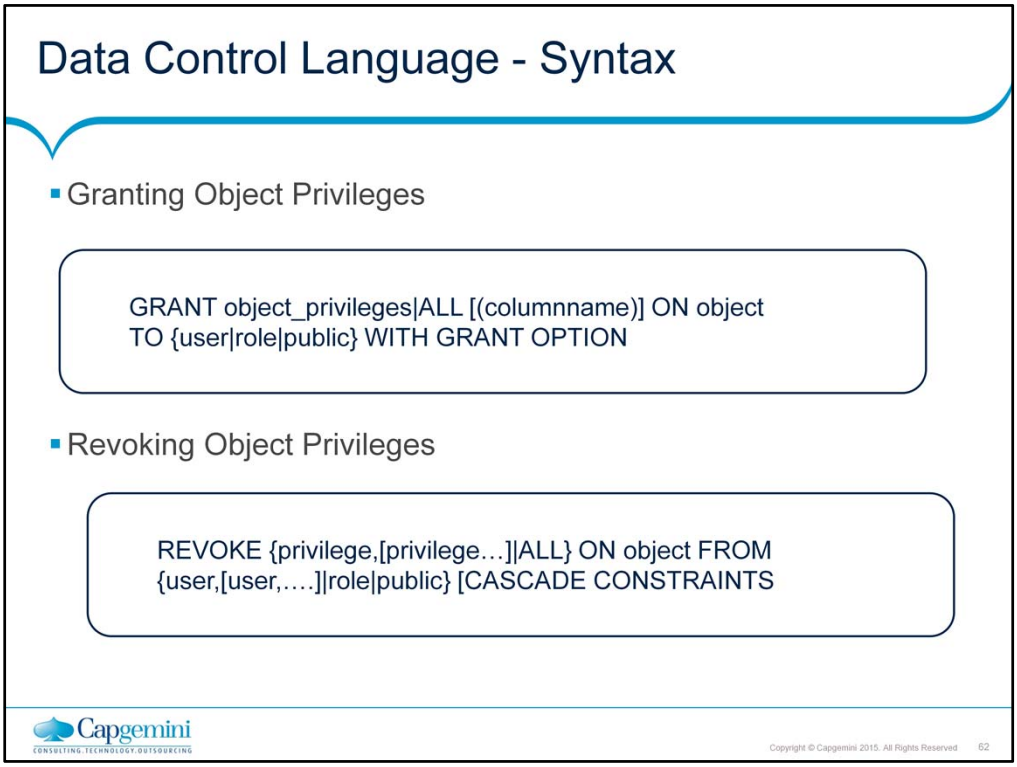
In a typical multi-user environment, you need to maintain security of the database access and use. As far as Database Security is concerned it can be at System Level and Data Level. System Security concerns items in the database such as users, usage of disk space etc… Data Security would mean access and usage of database objects.



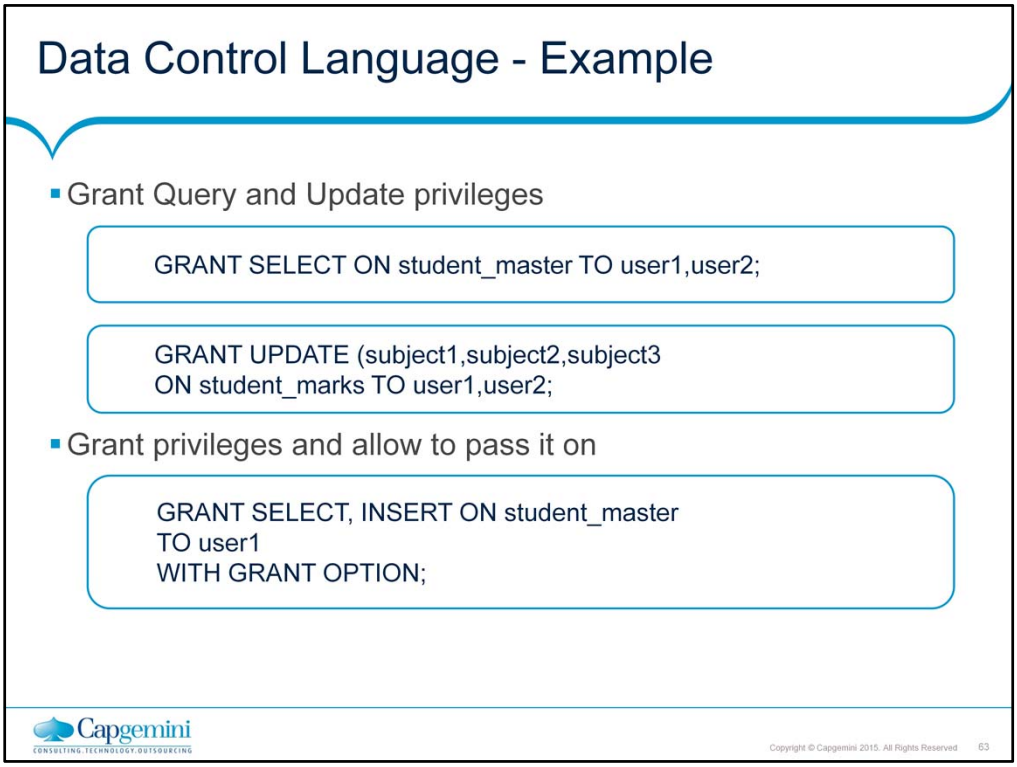
Object Privilege:
The right to execute a particular SQL statement is known as a Privilege. The DBA is a high level user who has access to all the objects in the database.
Object Privilege is a right to manipulate the content of database objects. In other words, object privilege is right to perform a particular action on a specific table, view etc.. Different object privileges are available for different types of objects. The owner of the object automatically has all object privileges on that object.



Each object has a set of privileges which can be granted. The lists of privileges which can be granted on a particular object is shown on the slide.
You can refer to USER_TAB_PRIVS_RECD table to know which object privileges are granted to the user. Similarly you can also refer to USER_TAB_PRIVS_MADE for knowing which object privileges are granted on user's objects.

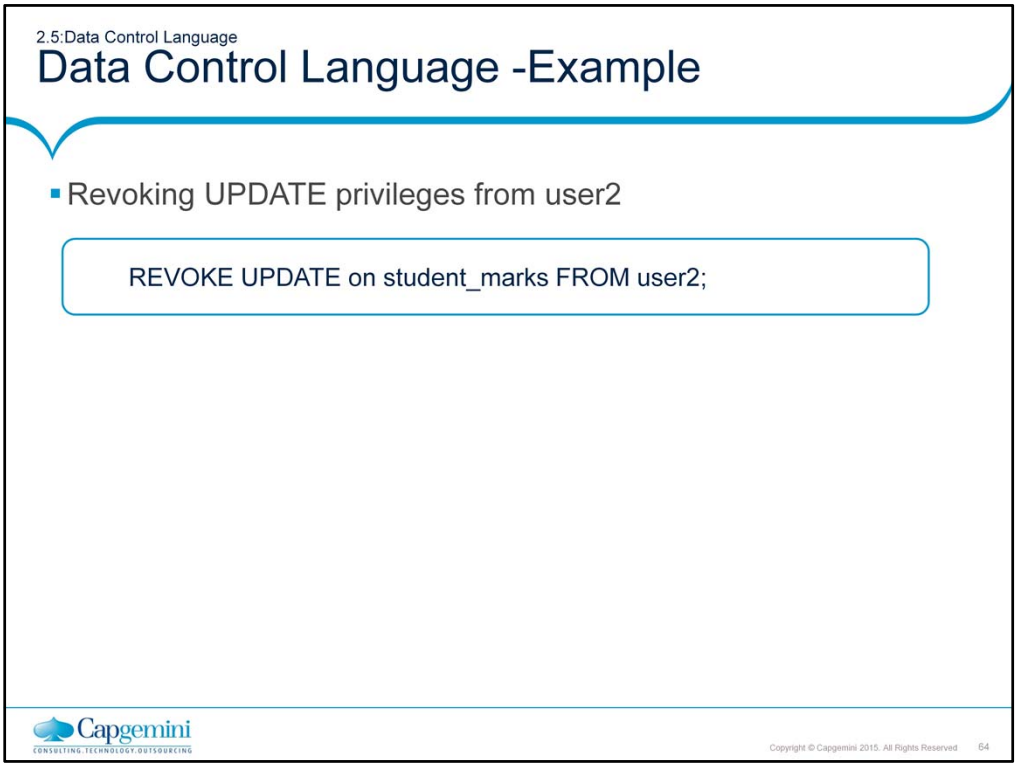


Data Control Language:
The syntax of both the Grant and Revoke statements is given on the slide. A user can give privileges using the GRANT statement
Understanding GRANT Statement:
Object_privileges : the right/privilege that is to be given
ALL : specifies/indicates all object privileges
columnname : specifies the column name on which privilege is to be given
ON object : the object on which privilege is to be given
TO user|role|public: The username or rolename to which the privilege is to be given. ALL grants privilege on the object to all users.
WITH GRANT OPTION : allows the grantee to grant the object privileges to other users and roles
To remove privileges the user can use REVOKE statement. This removes privileges that you mention from users specified and any other users to whom those privileges were granted through the WITH GRANT OPTION clause.

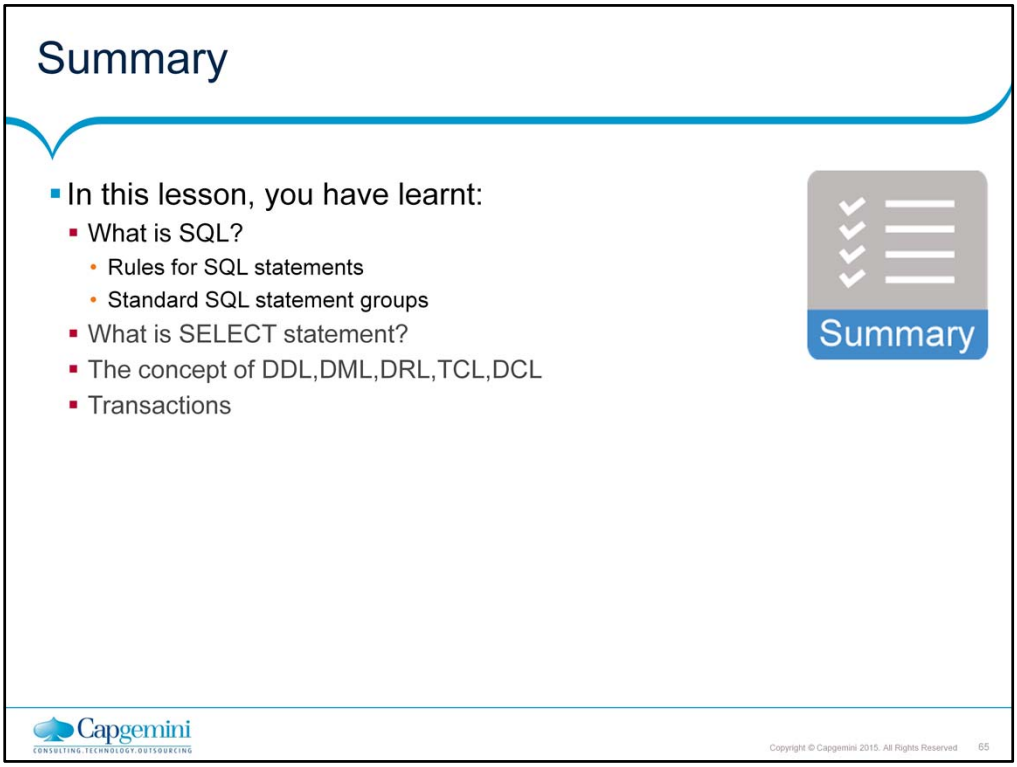


The owner of the object can grant access to all users using the Public keyword
In the following example all the users will be able to query student_master table

```
GRANT SELECT ON student_master
TO public;
```



You can also revoke multiple privileges from multiple users/roles.



Add the notes here.

Review Question

- Question 1: SQL categories are ___.
  - Option 1: DDL
  - Option 2: DML
  - Option 3: DSL
  - Option 4: DQL
  - Option 5: TCL
  - Option 6: TDL
- Question 2: A transaction is committed when the user issues a DDL statement.
  - True/False
- Question 3: All DML statements are auto committed
  - True/False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 66

Add the notes here.