Chapter 4
Structure Query Language (SQL)

Department: Computer
Course: DBMS
Faculty: Sana Shaikh

Source: Oracle Database 10g : SQL Fundamentals I (Volume 1)

Topics to be covered:

- Introduction to SQL
- Database Languages (DDL,DML, TCL, DCL)
- Data Definition Language(Create, Alter, Desc, Rename, Truncate, Drop)
- Data Manipulation Language (Select, Insert, Update, Delete)
- Transaction Control Language (Commit, Rollback, Savepoint)
- Data Control Language (Grant, Revoke)

Learning Outcomes:

Students should be able to:

- Explain in details what are DDL, DML, DCL, and TCL with examples.
- Learn the commands included in each of these categories with their purpose and explanation.
- Define the structure of the database and able to manage the data.

Relational Database Properties

A relational database:

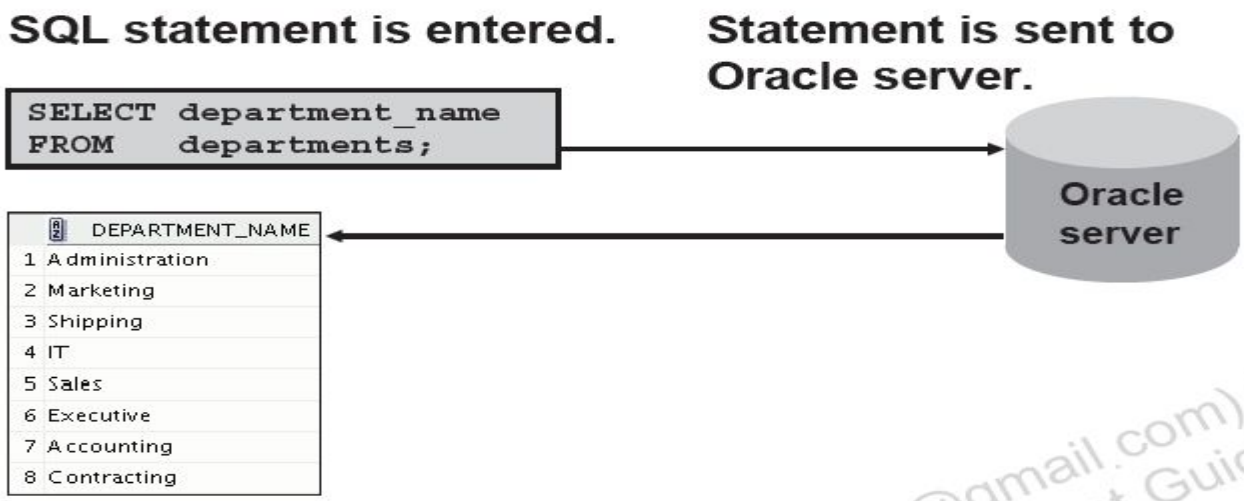
- Can be accessed and modified by executing structured query language (SQL) statements
- Contains a collection of tables with no physical pointers
- Uses a set of operators

Relational Database Properties

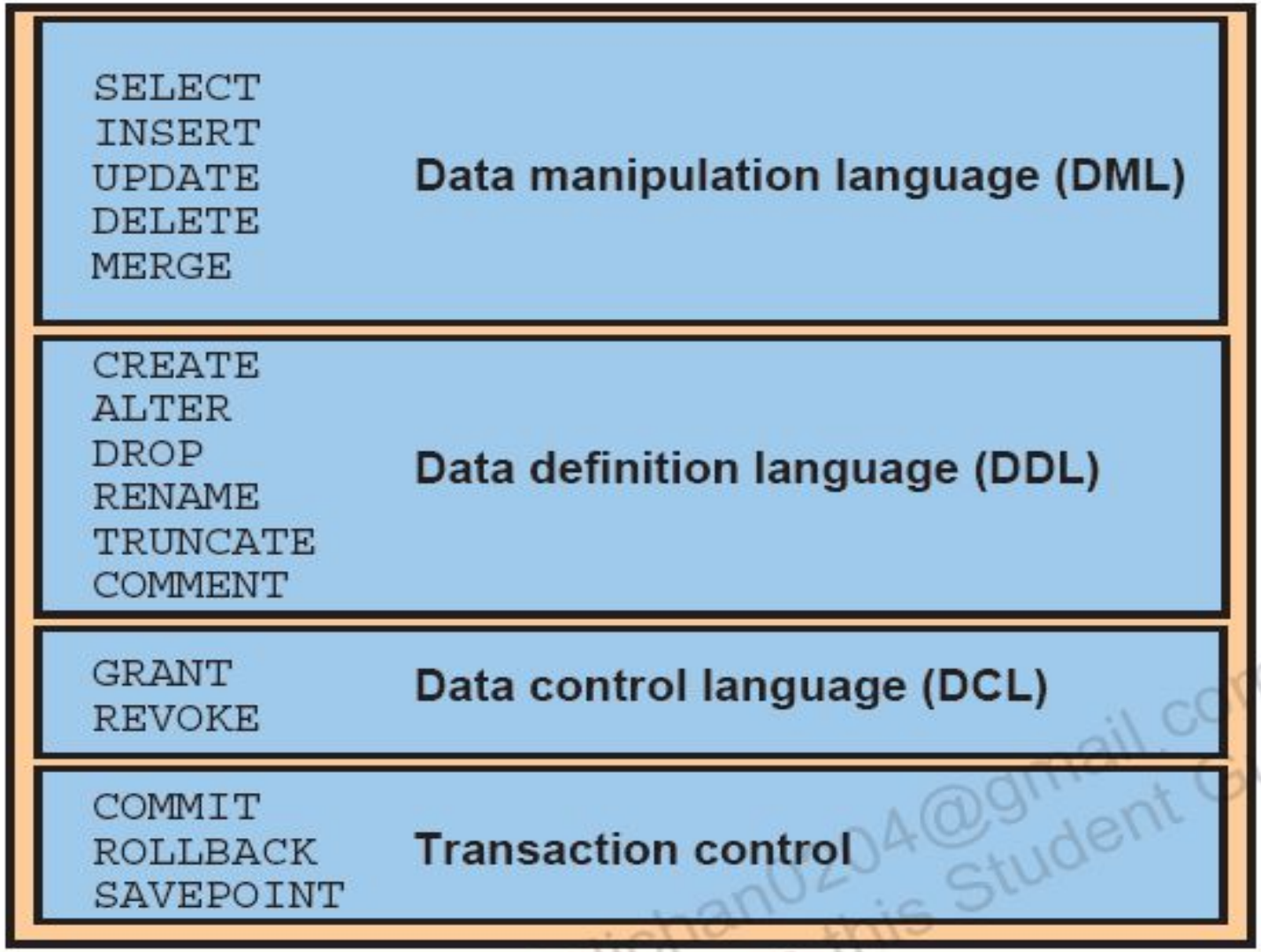
A relational database:

- Can be accessed and modified by executing structured query language (SQL) statements
- Contains a collection of tables with no physical pointers
- Uses a set of operators

Communicating with an RDBMS Using SQL



SQL Statements



Using DDL Statements to create and manage the Tables

Database Objects

Object	Description
Table	Basic unit of storage; composed of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

Creating and Selecting a Database

Find out what databases currently exist on the server:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

Creating and Selecting a Database

Find out what databases currently exist on the server:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

Show Current Database

SELECT DATABASE() FROM DUAL;

Creates your database:

mysql> CREATE DATABASE College;

Creating a database does not select it for use; you must do that explicitly

mysql> USE College;

CREATE TABLE statement

Naming Rules

Table names and column names:

- Must begin with a letter
- Must be 1–30 characters long
- Must contain only A–Z, a–z, 0–9, _, $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an Oracle server–reserved word

CREATE TABLE statement

DDL
Create

- You must have:
  - The `CREATE TABLE` privilege
  - A storage area
- You specify the:
  - Table name
  - Column name, column data type, and column size

Syntax

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

DEFAULT option

DDL
Create

- Specify a default value for a column during an insert.

```
… Dname varchar(10) Default ‘Comp’, ….
```

- Literal values, expressions, or SQL functions are legal values.
- Another column’s name or a pseudocolumn are illegal values.
- The default data type must match the column data type.

```
CREATE TABLE hire_dates
        (id             NUMBER(8),
         Dname varchar(10) Default ‘Comp’) ;
CREATE TABLE succeeded.
```

Creating Tables

DDL
Create

```
1  CREATE TABLE courses
2      (
3      name TEXT,
4      prereqs TEXT DEFAULT "None",
5      num_hours INTEGER DEFAULT 15);
6
7  INSERT INTO courses (name, num_hours) VALUES
8      ("Intro to JS", 25);
9  INSERT INTO courses (name) VALUES
10     ("Intro to HTML/CSS");
11 INSERT INTO courses (name) VALUES
12     ("Intro to SQL");
13 INSERT INTO courses (name, prereqs) VALUES
14     ("Advanced JS: Games & Viz", "Intro to JS");
15
16
17 SELECT * FROM courses;
18
```

Creating Tables

DDL
Create

```sql
CREATE TABLE courses
    (
    name TEXT,
    prereqs TEXT DEFAULT "None",
    num_hours INTEGER DEFAULT 15);

INSERT INTO courses (name, num_hours) VALUES
    ("Intro to JS", 25);
INSERT INTO courses (name) VALUES
    ("Intro to HTML/CSS");
INSERT INTO courses (name) VALUES
    ("Intro to SQL");
INSERT INTO courses (name, prereqs) VALUES
    ("Advanced JS: Games & Viz", "Intro to JS");

SELECT * FROM courses;
```

DATABASE SCHEMA

courses	4 rows
name	TEXT
prereqs	TEXT
num_hours	INTEGER

QUERY RESULTS

name	prereqs	num_hours
Intro to JS	None	25
Intro to HTML/CSS	None	15
Intro to SQL	None	15
Advanced JS: Games & Viz	Intro to JS	15

DDL
Create

Data Types

- **CHAR(SIZE)**
- **VARCHAR(SIZE)**
- **INTEGER(SIZE)**
- **FLOAT(size, d)**
- **DATE**
- **TIMESTAMP**
- **etc.**

Example: Creates a table called "Employee" that contains four columns: PersonID, FirstName, Address, and DOB:

Example: Creates a table called "Employee" that contains four columns: PersonID, FirstName, Address, and DOB:

```
CREATE TABLE Employee
(
    PersonID int,
    FirstName char(25),
    Address varchar(255),
    DOB Date
);
```

Create Table Using Another Table: Syntax:

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

Create Table Using Another Table: Syntax:

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

Example1: Create Employee backup table

Create Table Using Another Table: Syntax:

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

Example1: Create Employee backup table

```
CREATE TABLE  Emp_backup AS
    SELECT *
    FROM Employee;
```

Create Table Using Another Table: Syntax:

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

Example1: Create Employee backup table

```
CREATE TABLE  Emp_backup AS
    SELECT *
    FROM Employee;
```

Example2: Create separate table for those employees who live in Mumbai.

Create Table Using Another Table: Syntax:

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

Example1: Create Employee backup table

```
CREATE TABLE  Emp_backup AS
    SELECT *
    FROM Employee;
```

Example2: Create separate table for those employees who live in Mumbai.

```
CREATE TABLE  Emp_Mum AS
    SELECT *
    FROM Employee
    WHERE Address = 'Mumbai' ;
```

Displaying the Table Structure

DDL
Desc

- Use the DESCRIBE command to display the structure of a table.

```
DESC[RIBE]  tablename
```

Using the DESCRIBE Command

```
DESCRIBE employees
```

```
DESCRIBE employees
Name                          Null     Type
----------------------------- -------- ------------
EMPLOYEE_ID                   NOT NULL NUMBER(6)
FIRST_NAME                             VARCHAR2(20)
LAST_NAME                     NOT NULL VARCHAR2(25)
EMAIL                         NOT NULL VARCHAR2(25)
PHONE_NUMBER                           VARCHAR2(20)
HIRE_DATE                     NOT NULL DATE
JOB_ID                        NOT NULL VARCHAR2(10)
SALARY                                 NUMBER(8,2)
COMMISSION_PCT                         NUMBER(2,2)
MANAGER_ID                             NUMBER(6)
DEPARTMENT_ID                          NUMBER(4)

11 rows selected
```

DDL
Alter

ALTER TABLE Statement

Use the `ALTER TABLE` statement to:

- Add a new column
- Modify an existing column
- Define a default value for the new column
- Drop a column

ALTER TABLE Statement

DDL
Alter

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
1	149	Zlotkey	126000	29-JAN-00
2	174	Abel	132000	11-MAY-96
3	176	Taylor	103200	24-MAR-98

ALTER TABLE Statement

DDL
Alter

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
1	149	Zlotkey	126000	29-JAN-00
2	174	Abel	132000	11-MAY-96
3	176	Taylor	103200	24-MAR-98

- You use the ADD clause to add columns:

```
ALTER TABLE dept80
ADD         (job_id VARCHAR2(9));
ALTER TABLE succeeded.
```

ALTER TABLE Statement

DDL
Alter

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
1	149	Zlotkey	126000	29-JAN-00
2	174	Abel	132000	11-MAY-96
3	176	Taylor	103200	24-MAR-98

- You use the ADD clause to add columns:

```
ALTER TABLE dept80
ADD         (job_id VARCHAR2(9));
ALTER TABLE succeeded.
```

- The new column becomes the last column:

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
1	149	Zlotkey	126000	29-JAN-00	(null)
2	174	Abel	132000	11-MAY-96	(null)
3	176	Taylor	103200	24-MAR-98	(null)

ALTER TABLE Statement

DDL
Alter

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
             [, column datatype]...);
```

Guidelines

- You can increase the width or precision of a numeric column.
- You can increase the width of numeric or character columns.
- You can decrease the width of a column if:

ALTER TABLE Statement

DDL
Alter

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```

Guidelines

- You can increase the width or precision of a numeric column.
- You can increase the width of numeric or character columns.
- You can decrease the width of a column if:
  - The column contains only null values
  - The table has no rows
  - The decrease in column width is not less than the existing values in that column
- You can change the data type if the column contains only null values. The exception to this is CHAR-to-VARCHAR2 conversions, which can be done with data in the columns.
- You can convert a CHAR column to the VARCHAR2 data type or convert a VARCHAR2 column to the CHAR data type only if the column contains null values or if you do not change the size.
- A change to the default value of a column affects only subsequent insertions to the table.

ALTER TABLE Statement

DDL
Alter

Modifying a Column

- You can change a column's data type, size, and default value.

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2(30));
ALTER TABLE succeeded.
```

- A change to the default value affects only subsequent insertions to the table.

ALTER TABLE Statement

DDL
Alter

```
ALTER TABLE table
DROP        (column);
```

DDL
Truncate

TRUNCATE Statement

- Removes all rows from a table, leaving the table empty and the table structure intact
- Is a data definition language (DDL) statement rather than a DML statement; cannot easily be undone
- Syntax:

```
TRUNCATE TABLE table_name;
```

- Example:

```
TRUNCATE TABLE copy_emp;
```

Dropping a Table

DDL
Drop

- All data and structure in the table are deleted.
- Any pending transactions are committed.
- All indexes are dropped.
- All constraints are dropped.
- You *cannot* roll back the DROP TABLE statement.

```
DROP TABLE dept80;
DROP TABLE succeeded.
```

Guidelines

- All data is deleted from the table.
- Any views and synonyms remain but are invalid.
- Any pending transactions are committed.
- Only the creator of the table or a user with the DROP ANY TABLE privilege can remove a table.

Note: The DROP TABLE statement, once executed, is irreversible. The Oracle server does not question the action when you issue the DROP TABLE statement. If you own that table or have a high-level privilege, then the table is immediately removed. As with all DDL statements, DROP TABLE is committed automatically.

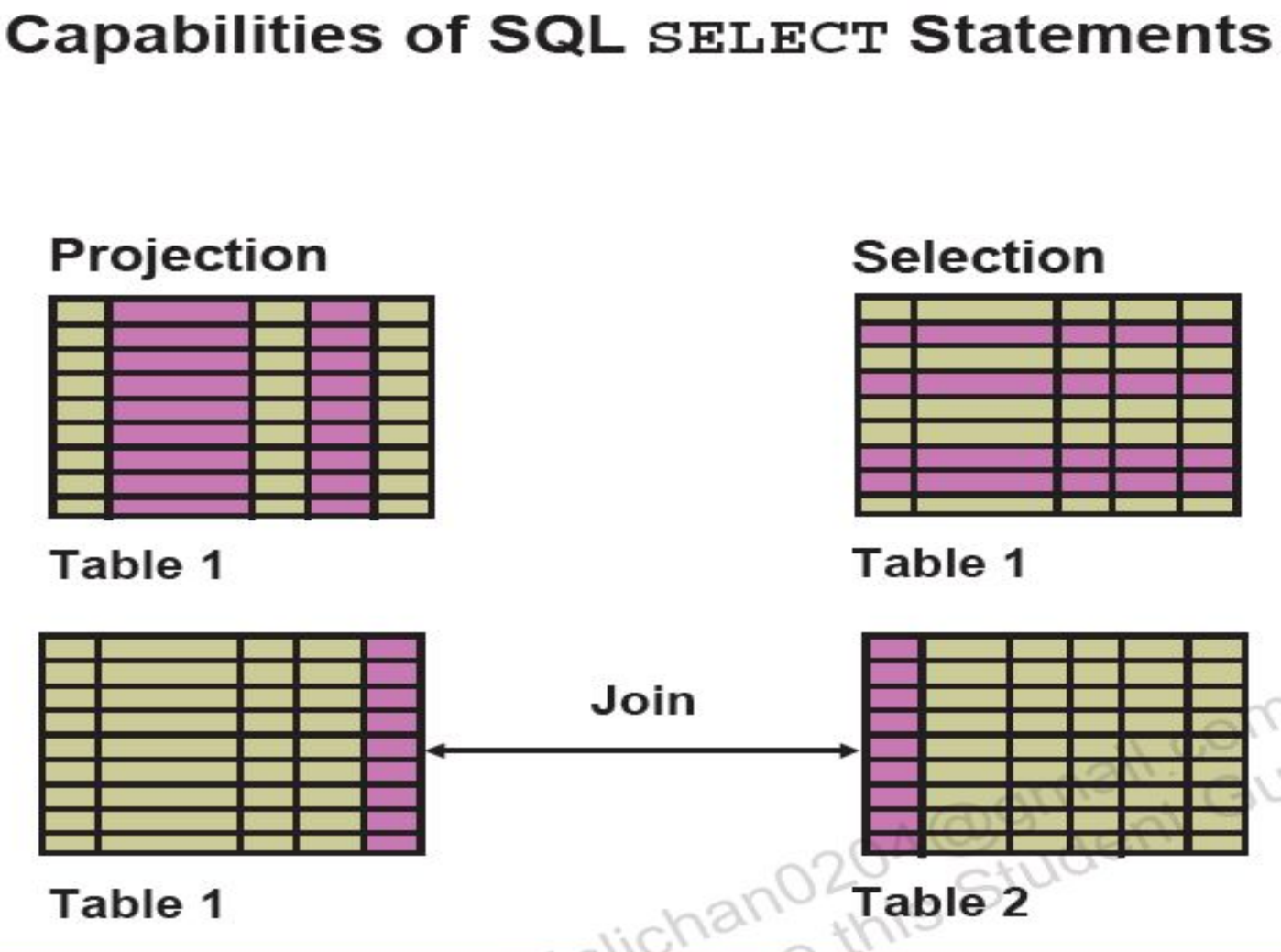
DML

Data Manipulation Language

- A DML statement is executed when you:
  - Add new rows to a table
  - Modify existing rows in a table
  - Remove existing rows from a table
- A *transaction* consists of a collection of DML statements that form a logical unit of work.

Retrieving the data using SQL SELECT statement

DML
Select



DML
Select

Basic SELECT Statement

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM    table;
```

- SELECT identifies the columns to be displayed.
- FROM identifies the table containing those columns.

DML

Select

Selecting All Columns

```
SELECT *
FROM   departments;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

DML
Select

Selecting Specific Columns

```
SELECT department_id, location_id
FROM   departments;
```

	DEPARTMENT_ID	LOCATION_ID
1	10	1700
2	20	1800
3	50	1500
4	60	1400
5	80	2500
6	90	1700
7	110	1700
8	190	1700

Writing SQL Statements

DML
Select

- SQL statements are not case-sensitive.
- SQL statements can be on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.
- In SQL Developer, SQL statements can optionally be terminated by a semicolon (;). Semicolons are required if you execute multiple SQL statements.
- In SQL*Plus, you are required to end each SQL statement with a semicolon (;).

DML
Select

Arithmetic Expressions

Create expressions with number and date data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

	LAST_NAME	SALARY	SALARY+300
1	Whalen	4400	4700
2	Hartstein	13000	13300
3	Fay	6000	6300
4	Higgins	12000	12300
5	Gietz	8300	8600
6	King	24000	24300
7	Kochhar	17000	17300
8	De Haan	17000	17300
9	Hunold	9000	9300
10	Ernst	6000	6300

...

DML
Select

Defining a Null Value

- A null is a value that is unavailable, unassigned, unknown, or inapplicable.
- A null is not the same as a zero or a blank space.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Whalen	AD_ASST	4400	(null)
2	Hartstein	MK_MAN	13000	(null)
3	Fay	MK_REP	6000	(null)
...				
17	Zlotkey	SA_MAN	10500	0.2
18	Abel	SA_REP	11000	0.3
19	Taylor	SA_REP	8600	0.2
20	Grant	SA_REP	7000	0.15

DML
Select

Null Values in Arithmetic Expressions

Arithmetic expressions containing a null value evaluate to null.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

	LAST_NAME	12*SALARY*COMMISSION_PCT
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)
4	Higgins	(null)

...

17	Zlotkey	25200
18	Abel	39600
19	Taylor	20640
20	Grant	12600

DML
Select

Defining a Column Alias

A column alias:

- Renames a column heading
- Is useful with calculations
- Immediately follows the column name (There can also be the optional `AS` keyword between the column name and alias.)
- Requires double quotation marks if it contains spaces or special characters, or if it is case-sensitive

DML
Select

Using Column Aliases

```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```

	NAME	COMM
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)

...

```
SELECT last_name "Name" , salary*12 "Annual Salary"
FROM   employees;
```

	Name	Annual Salary
1	Whalen	52800
2	Hartstein	156000
3	Fay	72000

...

DML
Select

Duplicate Rows

The default display of queries is all rows, including duplicate rows.

1

```
SELECT department_id
FROM   employees;
```

	DEPARTMENT_ID
1	10
2	20
3	20
4	110
5	110

...

2

```
SELECT DISTINCT department_id
FROM   employees;
```

	DEPARTMENT_ID
1	(null)
2	20
3	90
4	110
5	50

...

DML

Select

Restricting and Sorting data

DML
Select

Limiting the Rows That Are Selected

- Restrict the rows that are returned by using the WHERE clause:

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM   table
[WHERE condition(s)];
```

- The WHERE clause follows the FROM clause.

Limiting Rows Using a Selection

DML
Select

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20
4	205	Higgins	AC_MGR	110
5	206	Gietz	AC_ACCOUNT	110
6	100	King	AD_PRES	90
7	101	Kochhar	AD_VP	90
...				
20	178	Grant	SA_REP	(null)

"retrieve all employees in department 90"

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

Using the WHERE Clause

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

Character Strings and Dates

DML
Select

- Character strings and date values are enclosed in single quotation marks.
- Character values are case sensitive, and date values are format sensitive.
- The default date format is DD-MON-RR.

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	Whalen	AD_ASST	10

DML
Select

Comparison Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Logical Conditions

Operator	Meaning
AND	Returns TRUE if *both* component conditions are true
OR	Returns TRUE if *either* component condition is true
NOT	Returns TRUE if the following condition is false

Using Comparison Conditions

DML

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000 ;
```

	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

Using the BETWEEN Condition

Use the BETWEEN condition to display rows based on a range of values:

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Lower limit Upper limit

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

DML
Select

Using the IN Condition

Use the IN membership condition to test for values in a list:

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	201	Hartstein	13000	100
2	101	Kochhar	17000	100
3	102	De Haan	17000	100
4	124	Mourgos	5800	100
5	149	Zlotkey	10500	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

Using the LIKE Condition

DML
Select

- Use the LIKE condition to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers:
  - % denotes zero or many characters.
  - _ denotes one character.

```
SELECT    first_name
FROM      employees
WHERE     first_name LIKE 'S%' ;
```

	FIRST_NAME
1	Shelley
2	Steven

- You can combine pattern-matching characters:

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%' ;
```

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

- You can use the ESCAPE identifier to search for the actual % and _ symbols.

Using the NULL Conditions

DML
Select

Test for nulls with the IS NULL operator.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

	LAST_NAME	MANAGER_ID
1	King	(null)

DML
Select

Using the AND Operator

AND requires both conditions to be true:

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%' ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	201	Hartstein	MK_MAN	13000
2	149	Zlotkey	SA_MAN	10500

Using the OR Operator

OR requires either condition to be true:

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%' ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	201	Hartstein	MK_MAN	13000
2	205	Higgins	AC_MGR	12000
3	100	King	AD_PRES	24000
4	101	Kochhar	AD_VP	17000
5	102	De Haan	AD_VP	17000
6	124	Mourgos	ST_MAN	5800
7	149	Zlotkey	SA_MAN	10500
8	174	Abel	SA_REP	11000

Using the NOT Operator

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

Using the ORDER BY Clause

DML
Select

- Sort retrieved rows with the ORDER BY clause:
  - ASC: ascending order, default
  - DESC: descending order
- The ORDER BY clause comes last in the SELECT statement:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date ;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
...				
20	Zlotkey	SA_MAN	80	29-JAN-00

Sorting

DML
Select

- Sorting in descending order:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date DESC ;
```

1

- Sorting by column alias:

```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal ;
```

2

- Sorting by multiple columns:

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

3

DML
Insert

INSERT Statement Syntax

- Add new rows to a table by using the INSERT statement:

```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

- With this syntax, only one row is inserted at a time.

DML
Insert

Adding a New Row to a Table

70 Public Relations	100	1700

New row

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Insert new row into the DEPARTMENTS table

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700
9	190	Contracting	(null)	1700

DML
Insert

Inserting New Rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally, list the columns in the `INSERT` clause.

```
INSERT INTO departments(department_id,
       department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 rows inserted
```

- Enclose character and date values in single quotation marks.

DML
Insert

Inserting Rows with Null Values

- Implicit method: Omit the column from the column list.

```
INSERT INTO   departments (department_id,
                           department_name )
VALUES          (30, 'Purchasing');
1 rows inserted
```

- Explicit method: Specify the NULL keyword in the VALUES clause.

```
INSERT INTO   departments
VALUES          (100, 'Finance', NULL, NULL);
1 rows inserted
```

DML
Insert

Inserting Special Values

The SYSDATE function records the current date and time.

```
INSERT INTO employees (employee_id,
                       first_name, last_name,
                       email, phone_number,
                       hire_date, job_id, salary,
                       commission_pct, manager_id,
                       department_id)
VALUES                 (113,
                       'Louis', 'Popp',
                       'LPOPP', '515.124.4567',
                       SYSDATE, 'AC_ACCOUNT', 6900,
                       NULL, 205, 100);
1 rows inserted
```

DML
Insert

Inserting Specific Date Values

- Add a new employee.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'AC_ACCOUNT', 11000, NULL, 100, 30);
1 rows inserted
```

- Verify your addition.

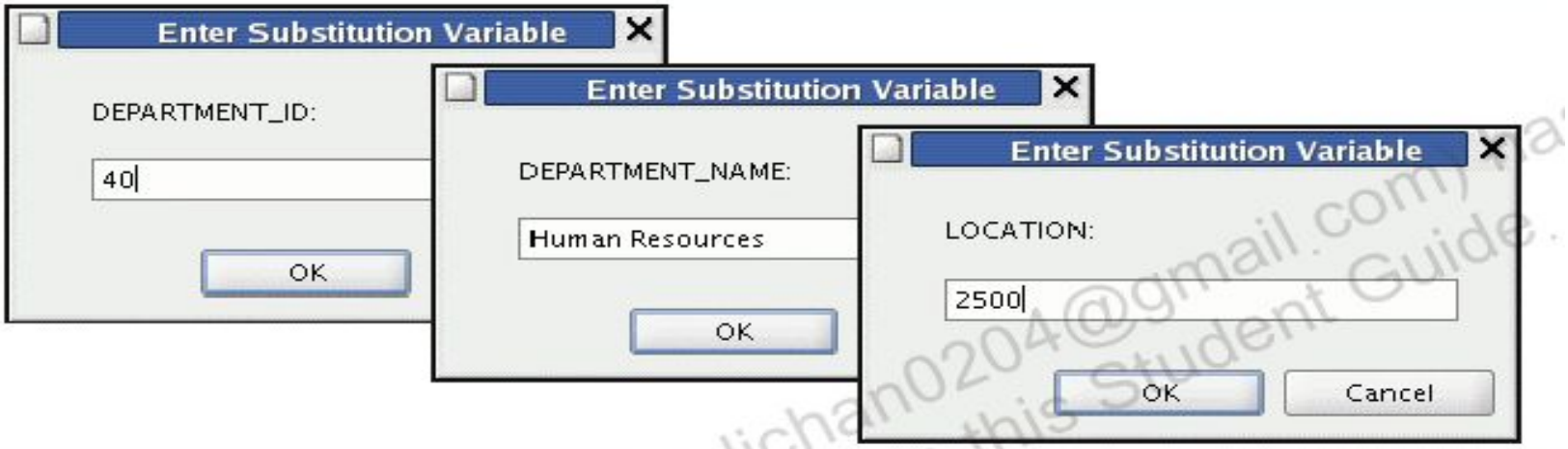
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
1	114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	AC_ACCOUNT	11000	(null)

Creating a Script

DML
Insert

- Use & substitution in a SQL statement to prompt for values.
- & is a placeholder for the variable value.

```
INSERT INTO departments
            (department_id, department_name, location_id)
VALUES      (&department_id, '&department name', &location);
```



DML
Update

UPDATE Statement Syntax

- Modify existing rows with the UPDATE statement:

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

- Update more than one row at a time (if required).

DML
Update

Changing Data in a Table

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_PCT
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	(null)
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	(null)
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	(null)
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	(null)
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	(null)
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	(null)
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	(null)

Update rows in the EMPLOYEES table:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_PCT
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	(null)
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	(null)
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	(null)
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	(null)
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	(null)
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	(null)
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	(null)

DML
Update

Updating Rows in a Table

- Specific row or rows are modified if you specify the WHERE clause:

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 113;
1 rows updated
```

- All rows in the table are modified if you omit the WHERE clause:

```
UPDATE   copy_emp
SET      department_id = 110;
22 rows updated
```

DML
Delete

Removing a Row from a Table

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	30	Purchasing	(null)	(null)
2	40	Human Resources	(null)	2500
3	10	Administration	200	1700
4	20	Marketing	201	1800
5	50	Shipping	124	1500
6	60	IT	103	1400

Delete a row from the DEPARTMENTS table:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	40	Human Resources	(null)	2500
2	10	Administration	200	1700
3	20	Marketing	201	1800
4	50	Shipping	124	1500
5	60	IT	103	1400

DELETE Statement

DML
Delete

You can remove existing rows from a table by using the DELETE statement:

```
DELETE [FROM]   table
[WHERE          condition];
```

Deleting Rows from a Table

DML
Delete

- Specific rows are deleted if you specify the `WHERE` clause:

```
DELETE FROM departments
WHERE  department_name = 'Finance';
1 rows deleted
```

- All rows in the table are deleted if you omit the `WHERE` clause:

```
DELETE FROM  copy_emp;
22 rows deleted
```

Transaction Control Language

TCL

Database Transactions

A database transaction consists of one of the following:

- DML statements that constitute one consistent change to the data
- One DDL statement
- One data control language (DCL) statement

TCL

Database Transactions

- Begin when the first DML SQL statement is executed
- End with one of the following events:
  - A `COMMIT` or `ROLLBACK` statement is issued.
  - A DDL or DCL statement executes (automatic commit).
  - The user exits SQL Developer or SQL*Plus.
  - The system crashes.

TCL

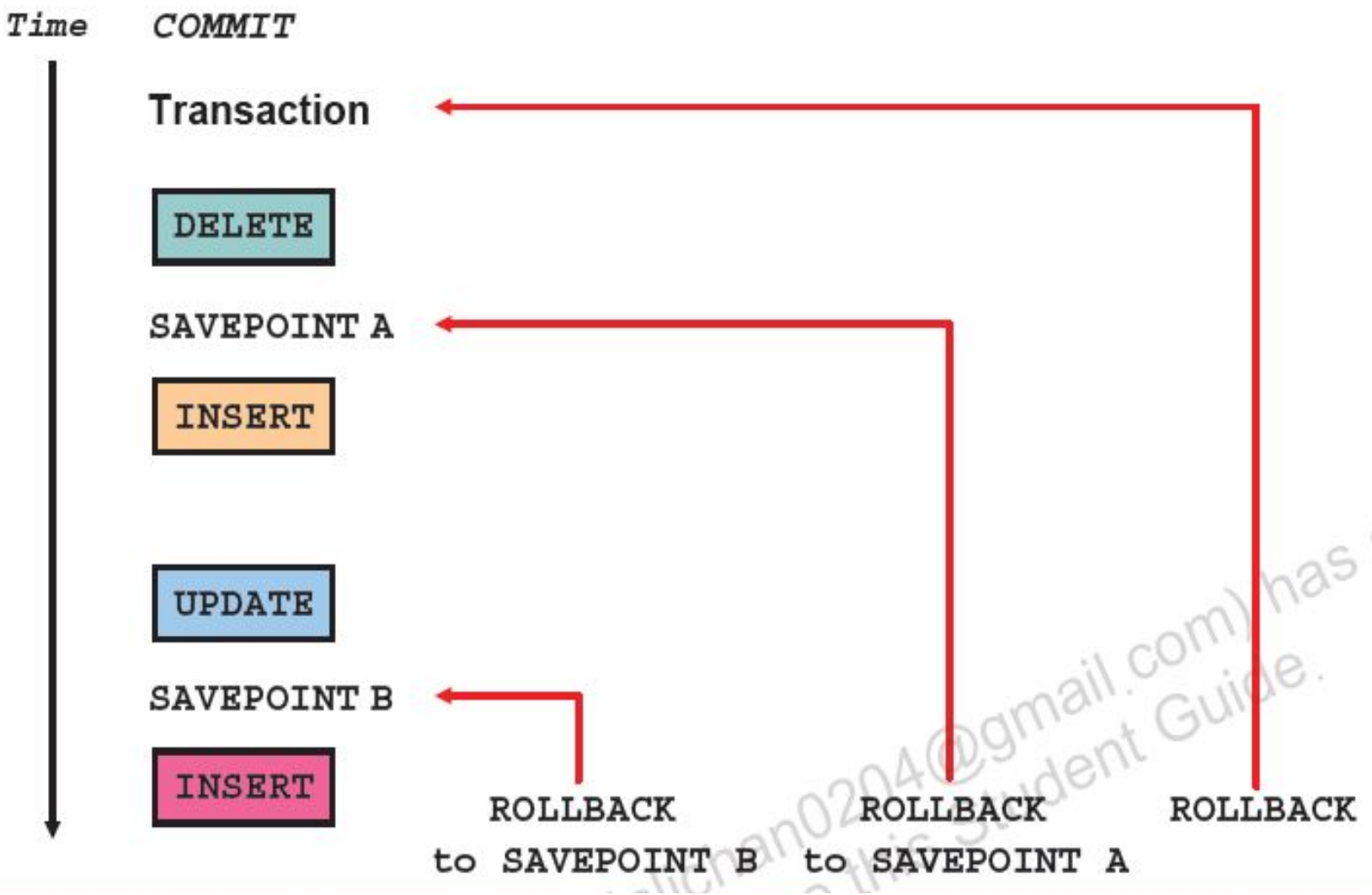
Advantages of COMMIT and ROLLBACK Statements

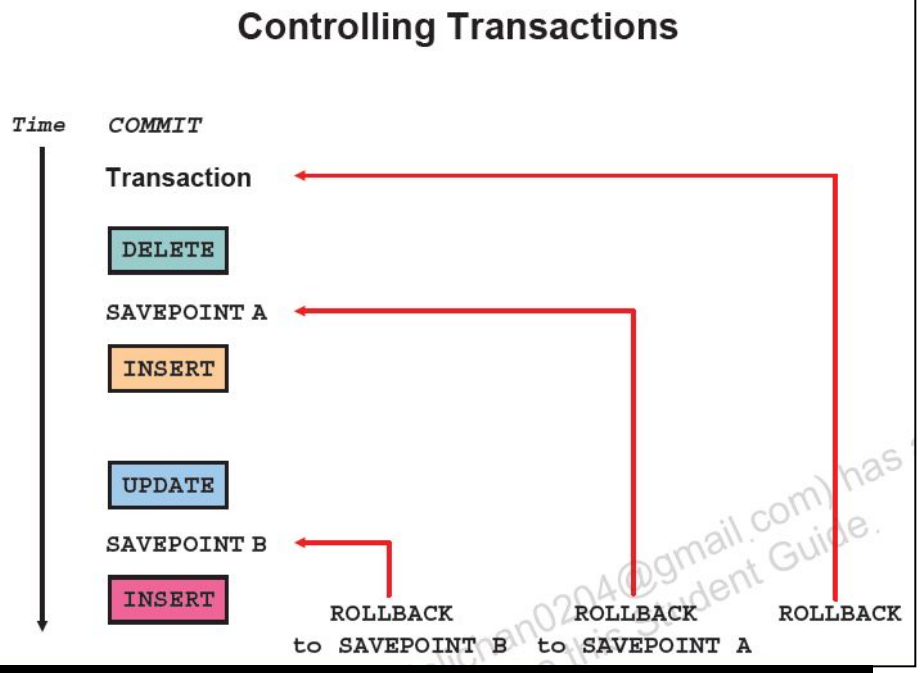
With COMMIT and ROLLBACK statements, you can:

- Ensure data consistency
- Preview data changes before making changes permanent
- Group logically related operations

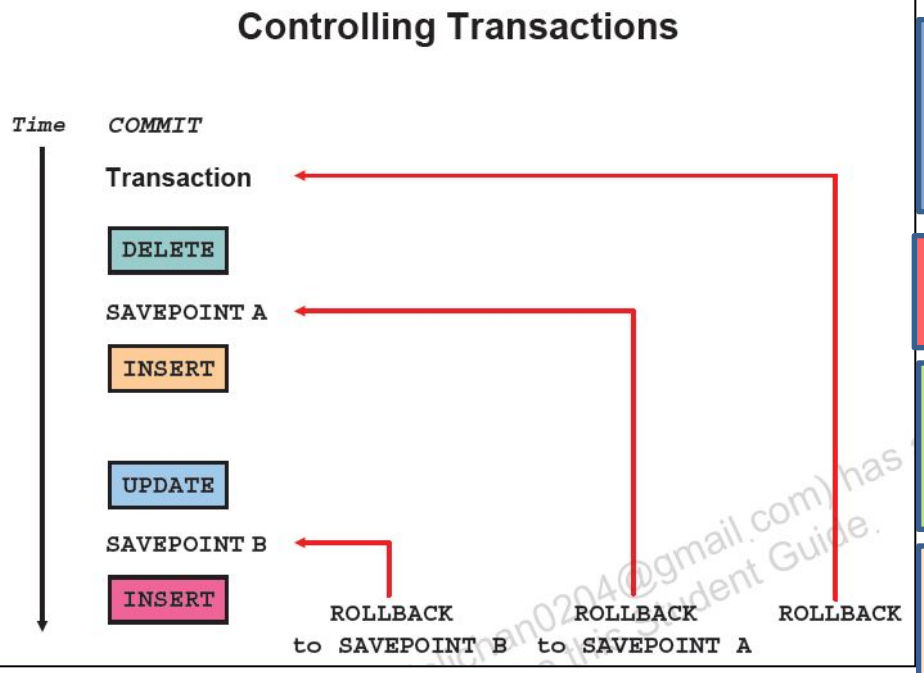
Controlling Transactions

TCL





Student		
Roll No	Name	Marks
1	Sami	50
2	Reema	60
3	Heena	45



Student		
Roll No	Name	Marks
1	Sami	50
2	Reema	60
3	Heena	45

Delete for Rollno=2

1	Sami	50
3	Heena	45

Savepoint A

Insert one record

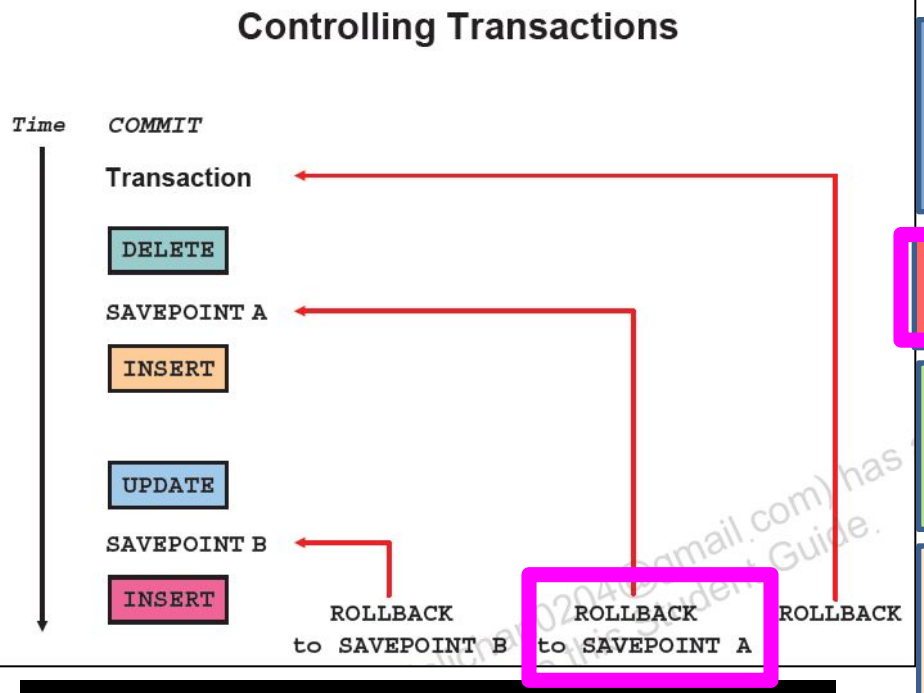
1	Sami	50
3	Hina	45
4	Sana	80

Update one record

Savepoint B

Insert one record

1	Sami	50
3	Hina	45
4	Sana	80
5	Saba	67



Student		
Roll No	Name	Marks
1	Sami	50
2	Reema	60
3	Heena	45

Delete for Rollno=2

1	Sami	50
3	Heena	45

Savepoint A

Insert one record

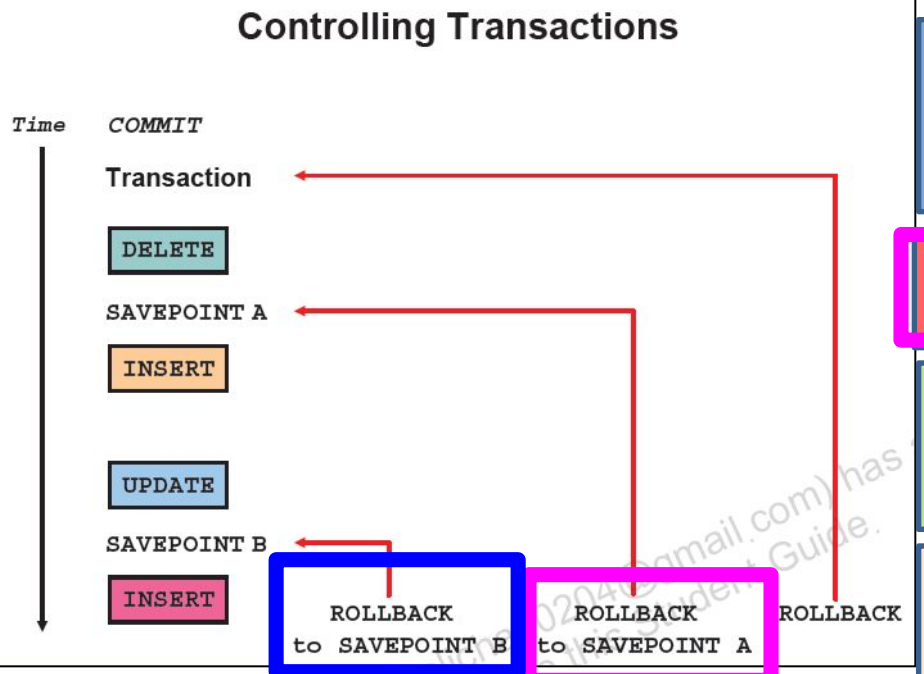
1	Sami	50
3	Hina	45
4	Sana	80

Update one record

Savepoint B

Insert one record

1	Sami	50
3	Hina	45
4	Sana	80
5	Saba	67



Student		
Roll No	Name	Marks
1	Sami	50
2	Reema	60
3	Heena	45

Delete for Rollno=2

1	Sami	50
3	Heena	45

Savepoint A

Insert one record

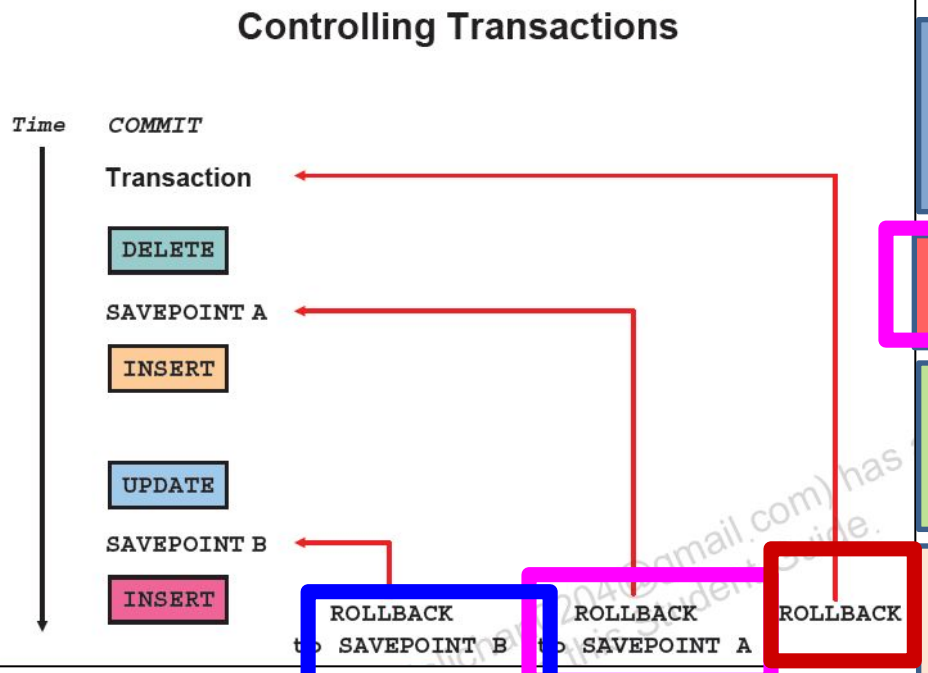
1	Sami	50
3	Hina	45
4	Sana	80

Update one record

Savepoint B

Insert one record

1	Sami	50
3	Hina	45
4	Sana	80
5	Saba	67



Student

Roll No	Name	Marks
1	Sami	50
2	Reema	60
3	Heena	45

Delete for Rollno=2

1	Sami	50
3	Heena	45

Savepoint A

Insert one record

1	Sami	50
3	Hina	45
4	Sana	80

Update one record

Savepoint B

Insert one record

1	Sami	50
3	Hina	45
4	Sana	80
5	Saba	67

Rolling Back Changes to a Marker

TCL

Savepoint

- Create a marker in a current transaction by using the `SAVEPOINT` statement.
- Roll back to that marker by using the `ROLLBACK TO SAVEPOINT` statement.

```
UPDATE...
SAVEPOINT update_done;
SAVEPOINT update_done succeeded
INSERT...
ROLLBACK TO update_done;
ROLLBACK TO succeeded
```

Implicit Transaction Processing

TCL

- An automatic commit occurs under the following circumstances:
  - DDL statement is issued
  - DCL statement is issued
  - Normal exit from SQL Developer or SQL*Plus, without explicitly issuing `COMMIT` or `ROLLBACK` statements
- An automatic rollback occurs under an abnormal termination of SQL Developer or SQL*Plus or a system failure.

TCL

State of the Data Before COMMIT or ROLLBACK

- The previous state of the data can be recovered.
- The current user can review the results of the DML operations by using the `SELECT` statement.
- Other users *cannot* view the results of the DML statements by the current user.
- The affected rows are *locked*; other users cannot change the data in the affected rows.

TCL
Commit

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.

Committing Data

TCL

Commit

- Make the changes:

```
DELETE FROM employees
WHERE  employee_id = 99999;
1 rows deleted

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 rows inserted
```

- Commit the changes:

```
COMMIT;
Commit complete
```

State of the Data After ROLLBACK

TCL
Rollback

Discard 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.

```
DELETE FROM copy_emp;
20 rows deleted
ROLLBACK ;
Rollback complete
```

Data Control Language

DCL

Privileges

- Database security:
  - System security
  - Data security
- System privileges: Gaining access to the database
- Object privileges: Manipulating the content of the database objects
- Schemas: Collection of objects such as tables, views, and sequences

DCL

System Privileges

- More than 100 privileges are available.
- The database administrator has high-level system privileges for tasks such as:
  - Creating new users
  - Removing users
  - Removing tables
  - Backing up tables

DCL

Creating Users

The DBA creates users with the `CREATE USER` statement.

```
CREATE USER user
IDENTIFIED BY   password;
```

```
CREATE USER  USER1
IDENTIFIED BY   USER1;
CREATE USER succeeded.
```

User System Privileges

DCL
Grant

- After a user is created, the DBA can grant specific system privileges to that user.

```
GRANT privilege [, privilege...]
TO user [, user| role, PUBLIC...];
```

- An application developer, for example, may have the following system privileges:
  - CREATE SESSION
  - CREATE TABLE
  - CREATE SEQUENCE
  - CREATE VIEW
  - CREATE PROCEDURE

DCL

Grant

Granting System Privileges

The DBA can grant specific system privileges to a user.

```
GRANT   create session, create table,
        create sequence, create view
TO      scott;
GRANT CREATE succeeded.
```

DCL

Revoke

Revoking Object Privileges

As user Alice, revoke the SELECT and INSERT privileges given to user Scott on the DEPARTMENTS table.

```
REVOKE   select, insert
ON       departments
FROM     scott;
REVOKE succeeded.
```