Database Management System Lab (CS-2094)

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

School of Computer Engineering

1 Credit

Dr. Jayanta Mondal

Lab Contents

Sr #	Major and Detailed Coverage Area	Lab#
1 2 3	More on DQL and DML Operators Row ID	2

DQL

A **query** is an inquiry into the database using the **SELECT** statement. A query is used to extract data from the database in a readable format according to the user's request. For instance, if you have an employee table, you might issue an SQL statement that returns the employee who is paid the most. This request to the database for usable employee information is a typical query that can be performed in a relational database.

The SELECT statement, the command that represents **Data Query Language (DQL)**, is the basic statement used to construct database queries. The **SELECT** statement is not a standalone statement, which means that one or more additional clauses (elements) are required for a syntactically correct query. The **FROM** clause is a mandatory clause and must always be used in conjunction with the SELECT statement. There are four keywords, or clauses, that are valuable parts of a SELECT statement. These keywords are as follows:

- ❑ SELECT
- ❑ FROM
- ❑ WHERE
- ❑ ORDER BY

DQL

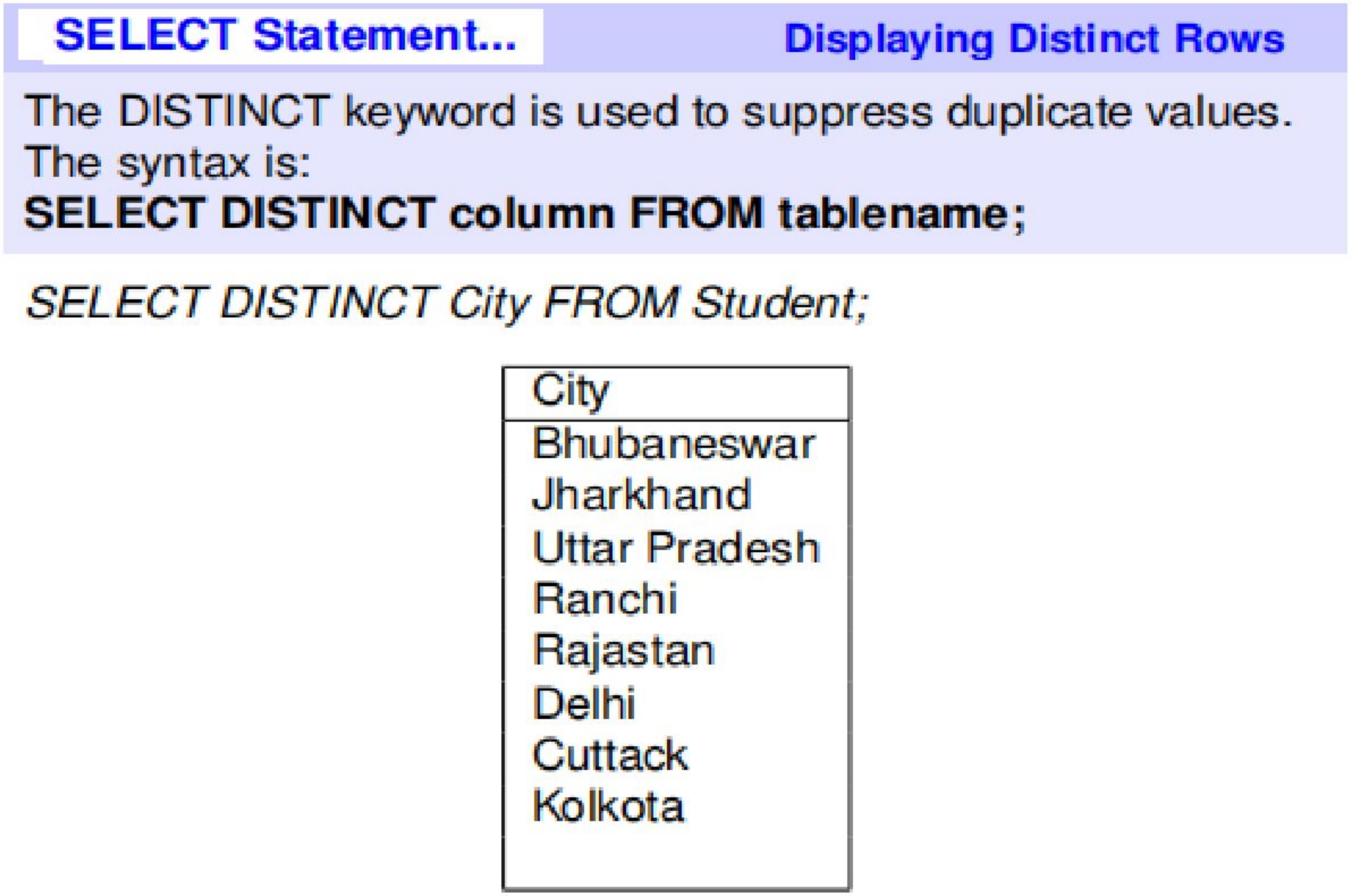
SELECT statement is used to retrieve data from the underlying table. The syntax is:
SELECT columns FROM tablename;

If the user wants to see all the columns in a table, * can be used in place of columns

SELECT Roll, CGPA FROM Student;

Roll	CGPA
101	9.0
102	6.7
103	8.97
104	8.5
105	9.2
106	7.9
210	8.99
211	8.6
212	5.98
165	9.15

DQL



SELECT Statement... **Displaying Distinct Rows**

The DISTINCT keyword is used to suppress duplicate values.
The syntax is:
SELECT DISTINCT column FROM tablename;

SELECT DISTINCT City FROM Student;

City
Bhubaneswar
Jharkhand
Uttar Pradesh
Ranchi
Rajastan
Delhi
Cuttack
Kolkota

DQL

Use of Arithmetic Expressions

The arithmetic expressions are used to display mathematically calculated data. The syntax is:

SELECT column, expression FROM tablename;

SELECT Name, Age, Age+3 FROM Student;

Name	Age	Age+3
Ram	19	22
Uday	20	23
Vikas	19	22
Sweta	19	22
Yogesh	18	21
Smriti	20	23
Sudam	21	24
Vikas	23	26
Manish	19	22

DQL

Use of Alias

The column aliases are used to rename a table's columns for the purpose of a particular SQL query. The syntax is:
SELECT column1, column2 [AS] Alias FROM tablename;

SELECT Name, Age, Age+3 "Passing Age"FROM Student;

Name	Age	Passing Age
Ram	19	22
Uday	20	23
Vikas	19	22
Sweta	19	22
Yogesh	18	21
Smriti	20	23
Sudam	21	24
Vikas	23	26
Manish	19	22

Column Aliases

Column aliases are used to **temporarily rename a table's columns** for the purpose of a particular query. The following syntax illustrates the use of column

aliases:

*SELECT COLUMN_NAME **AS** ALIAS_NAME FROM TABLE_NAME;*

Example:

SELECT ID, DESC **AS** PROD_DESC FROM PRODUCT;

SELECT ID, DESC **AS** "Prod Description" FROM PRODUCT;

Class Work

- Select distinct last name of the employee with appropriate aliases
- Select distinct first name of the employee with appropriate aliases

DQL

Concatenation

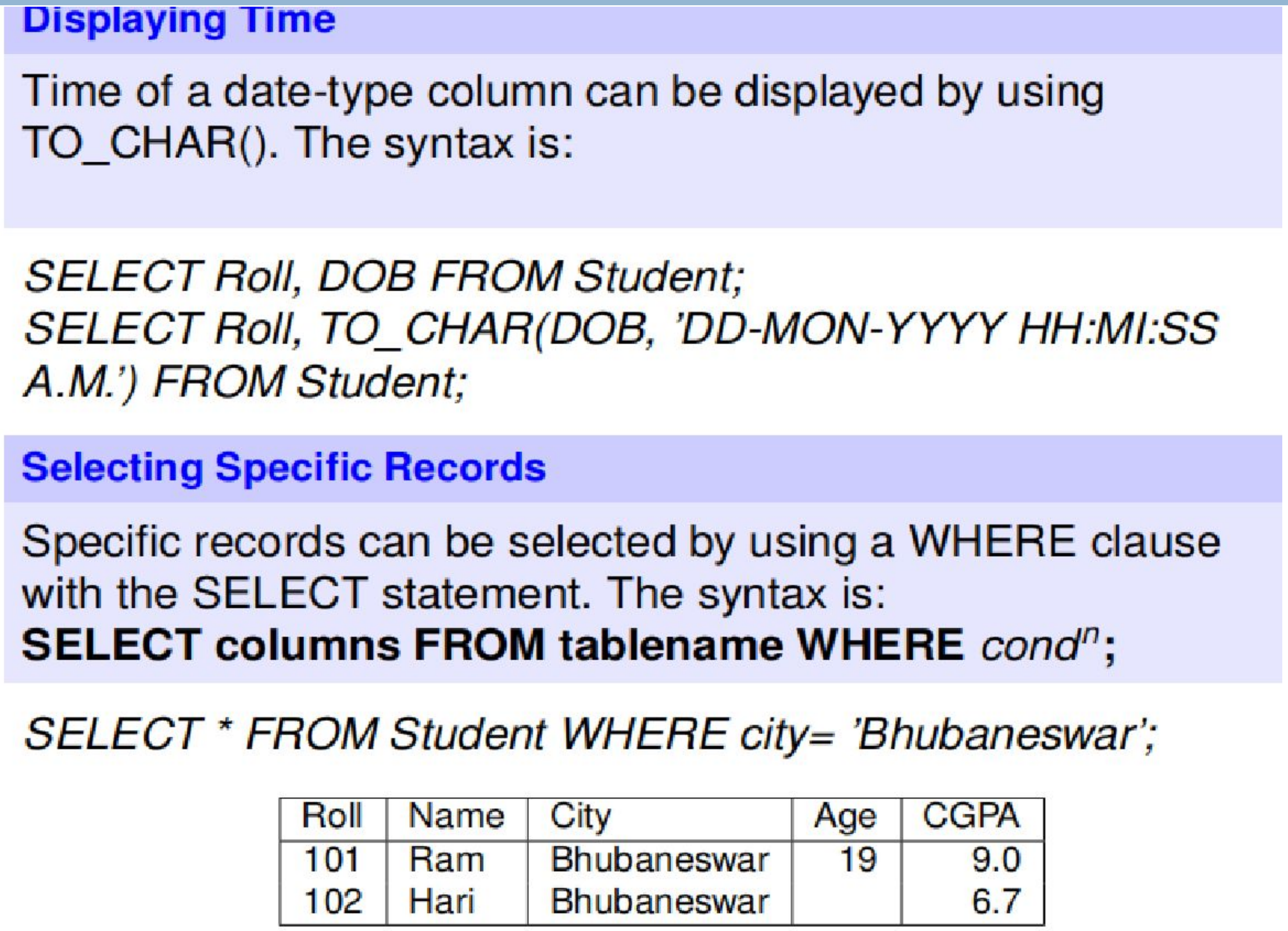
Concatenation joins a column or a character string to another column. The syntax is:
SELECT column1||' '||column2 [AS] ALIAS FROM tablename;

SELECT Name||' '||City FROM Student;
SELECT Name||' '||City AS "Address"FROM Student;

Name||' '||City
Ram Bhubaneswar
Hari Bhubaneswar
Uday Jharkhand
Vikas Uttar Pradesh
Sweta Ranchi
Yogesh Rajastan
Smriti Delhi
Sudam Cuttack
Vikas Kolkota
Manish

Address
Ram Bhubaneswar
Hari Bhubaneswar
Uday Jharkhand
Vikas Uttar Pradesh
Sweta Ranchi
Yogesh Rajastan
Smriti Delhi
Sudam Cuttack
Vikas Kolkota
Manish

DQL



Displaying Time

Time of a date-type column can be displayed by using TO_CHAR(). The syntax is:

SELECT Roll, DOB FROM Student;
SELECT Roll, TO_CHAR(DOB, 'DD-MON-YYYY HH:MI:SS A.M.') FROM Student;

Selecting Specific Records

Specific records can be selected by using a WHERE clause with the SELECT statement. The syntax is:
SELECT columns FROM tablename WHERE *cond^n***;**

*SELECT * FROM Student WHERE city= 'Bhubaneswar';*

Roll	Name	City	Age	CGPA
101	Ram	Bhubaneswar	19	9.0
102	Hari	Bhubaneswar		6.7

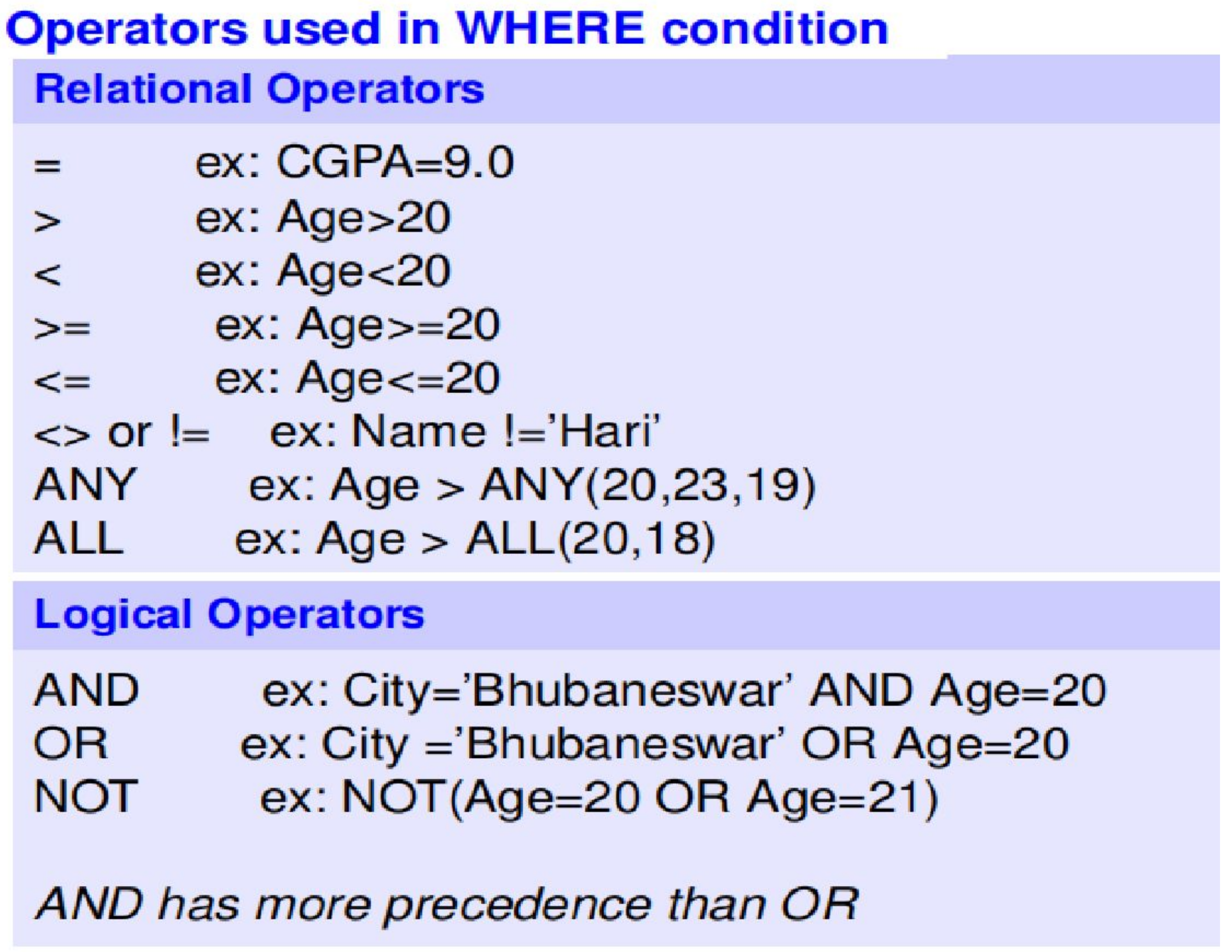
Operators

An operator is a reserved word or a character used primarily in an SQL statement's **WHERE** clause to perform operation(s), such as comparisons and arithmetic operations. Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

Commonly used operators are:

- Comparison operators
- Logical operators
- Operators used to negate conditions
- Arithmetic operators

DQL



Operators used in WHERE condition

Relational Operators	
=	ex: CGPA=9.0
>	ex: Age>20
<	ex: Age<20
>=	ex: Age>=20
<=	ex: Age<=20
<> or !=	ex: Name !='Hari'
ANY	ex: Age > ANY(20,23,19)
ALL	ex: Age > ALL(20,18)

Logical Operators	
AND	ex: City='Bhubaneswar' AND Age=20
OR	ex: City ='Bhubaneswar' OR Age=20
NOT	ex: NOT(Age=20 OR Age=21)

AND has more precedence than OR

Comparison Operators

Comparison operators test single values in an SQL statement. The comparison operators discussed consist of =, <>, <, and >. These operators are used to test:

- Equality
- Non-equality
- Less-than values
- Greater-than values

Equality

The equal operator compares single values to one another in an SQL statement.

The equal sign (=) symbolizes equality. When testing for equality, the compared values must match exactly, or no data is returned. If two values are equal during a comparison for equality, the returned value for the comparison is TRUE; the returned value is FALSE if equality is not found. This Boolean value (TRUE/FALSE) is used to determine whether data is returned according to the condition. This operator can be used by itself or combine with other operators.

Example - SELECT * FROM PRODUCT WHERE ID = ‘2345’;

Comparison Operators cont…

Non-equality

Opposite of equality operator

Example - SELECT * FROM PRODUCT WHERE ID <> ‘2345’;

SELECT * FROM JOB WHERE SALARY <> 20000

Less Than, Greater Than

Example –

SELECT * FROM PRODUCT WHERE COST > 20;

SELECT * FROM PRODUCT WHERE COST < 20;

Class Work

- Select the employees who are not staying in “Bhubaneswar”
- Select the employees who are staying in “Lucknow”

Logical Operators

Logical operators are those operators to make comparisons. Commonly used operators are:

- IS NULL
- IN
- EXISTS
- BETWEEN
- LIKE

IS NULL

The NULL operator is used to compare a value with a NULL value.

Example –

SELECT ID, LAST_NAME, FIRST_NAME, PAGER FROM EMPLOYEE WHERE PAGER **IS NULL**;

BETWEEN

The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value. The minimum and maximum values are included as part of the conditional set. **Example** –

SELECT * FROM PRODUCT WHERE COST BETWEEN 5.95 AND 14.5;

Logical Operators cont...

IN

The **IN** operator compares a value to a list of literal values that have been specified. For TRUE to be returned, the compared value must match at least one of the values in the list. **Example** –

SELECT * FROM PRODUCT WHERE ID IN (‘13’, ’9’, ’87’, ’119’);

LIKE

The **LIKE** operator is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator:

- The percent sign (**%**) - represents **zero, one, or multiple** characters.
- The underscore (_) - represents a **single** number or character

Example –

SELECT DESC FROM PRODUCT WHERE DESC LIKE ‘**%**S’;

SELECT DESC FROM PRODUCT WHERE DESC LIKE ‘_S’;

Logical Operators cont...

EXISTS

The **EXISTS** operator is used to search for the presence of a row in a specified table that meets certain criteria.

Example –

SELECT COST FROM PRODUCT WHERE **EXISTS** (SELECT COST FROM PRODUCT WHERE COST > 100);

Conjunctive Operators

Conjunctive operators provide a means to make multiple comparisons with different operators in the same SQL statement. These operators are:

- AND
- OR

AND

The **AND** operator allows the existence of multiple conditions in an SQL statement's WHERE clause. For an action to be taken by the SQL statement, whether it be a transaction or query, all conditions separated by the AND must be TRUE.

Example –

- SELECT * FROM PRODUCT WHERE COST > 10 AND COST < 30;
- SELECT * FROM PRODUCT WHERE ID = '7725' or ID = '2345';

Conjunctive OR Operator

The **OR** operator combines multiple conditions in an SQL statement's WHERE clause. For an action to be taken by the SQL statement, whether it is a transaction or query, at least one of the conditions that are separated by OR must be TRUE. **Example** –

- SELECT * FROM PRODUCT WHERE ID = '90' OR ID = '2345';
- SELECT * FROM PRODUCT WHERE COST > 10 AND (ID = '222' OR ID = '90' OR ID = '11235');

Negative Operators

The NOT operator reverses the meaning of the logical operator with which it is used. The NOT can be used with other operators to form the following methods:

- <>, != (NOT EQUAL)
- NOT BETWEEN
- NOT IN
- NOT LIKE
- IS NOT NULL
- NOT EXISTS

Arithmetic Operators

Arithmetic operators perform mathematical functions in SQL—the same as in most other languages. The four conventional operators for mathematical functions are :

- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)

Addition

Addition is performed through the use of the plus (+) symbol.

Example - SELECT SALARY + BONUS FROM JOB;

SELECT SALARY FROM JOB WHERE SALARY + BONUS > 40000;

Subtraction

Subtraction is performed through the use of the minus (-) symbol.

E.g. SELECT SALARY - BONUS FROM JOB;

SELECT SALARY FROM JOB WHERE SALARY - BONUS > 40000;

Arithmetic Operators cont...

Multiplication

Addition is performed through the use of the asterisk (*) symbol.

Example -

SELECT SALARY * 10 FROM JOB;

SELECT ID, SALARY * 1.1 FROM JOB WHERE BONUS IS NOT NULL;

Division

Division is performed through the use of the slash (/) symbol.

Example -

SELECT SALARY / 10 FROM JOB;

SELECT SALARY FROM JOB WHERE (SALARY / 10) > 40000;

ORDER BY Clause

Data can be **sorted** by using the **ORDER BY** clause. The **ORDER BY** clause arranges the results of a query in a **specified sorted format**. The default ordering of the ORDER BY clause is an **ascending order**; the sort displays in the order A–Z if it's sorting output names alphabetically. A descending order for alphabetical output would be displayed in the order Z–A. Ascending order for output for numeric values between 1 and 9 would be displayed 1–9; descending order would be displayed as 9–1. Synt

[represents optional

*SELECT [* | DISTINCT COLUMN1, COLUMN2] FROM TABLE1 [, TABLE2] WHERE [CONDITION1 | EXPRESSION1] [AND | OR CONDITION2 | EXPRESSION2] ORDER BY COLUMN1 | INTEGER [ASC | DESC]*

Example:

SELECT ID, DESC, COST FROM PRODUCT WHERE COST < 20 **ORDER BY ID ASC;**

SELECT ID, DESC, COST FROM PRODUCT WHERE COST < 20 **ORDER BY 3 DESC;**

DQL

Sorting

ORDER BY clause using column name

ORDER BY clause is used to sort records in a table

SELECT columns FROM tablename [WHERE *cond^n***]**
ORDER BY column [ASC/DESC];

*SELECT * FROM Student ORDER BY Age;*
*SELECT * FROM Student ORDER By CGPA, Age DESC;*
NULL values come at the end of the table in case of ORDER BY clause

ORDER BY clause using column number

Records can be sorted by using the column number
SELECT columns FROM tablename [WHERE *cond^n***]**
ORDER BY columnno [ASC/DESC];
*SELECT * FROM Student ORDER BY 3;*

DUAL Table

DUAL table is a small worktable, which consists of only one column **DUMMY** and a single row with value **X** of VARCHAR2 type. This table is owned by user **SYS** and is available to all users. It is used for arithmetic calculations and date retrieval. Example –

- SELECT 2*5 FROM DUAL;
- SELECT SYSDATE FROM DUAL;

DML – Data Manipulation

DML is the part of SQL that enables a database user to actually propagate changes among data in a relational database. With DML, the user can populate tables with new data, update existing data in tables, and delete data from tables. Simple database queries can also be performed within a DML command. Three basic DML commands in SQL are **INSERT**, **UPDATE** AND **DELETE**

Inserting Data into Limited Columns of a Table

There is an way to insert data into specified columns and the syntax is:

INSERT INTO TABLE_NAME (‘COLUMN1’, ‘COLUMN2’) VALUES (‘VALUE1’, ‘VALUE2’);

Example - suppose we want to insert all values for an employee except a pager number. In this case, specify a column list as well as a VALUES list in INSERT statement.

INSERT INTO EMPLOYEE (EMP_ID, LAST_NAME, FIRST_NAME, MIDDLE_NAME, ADDRESS, CITY, STATE, ZIP, PHONE) VALUES (‘123456789’, ‘SMITH’, ‘JOHN’, ‘JAY’, ‘12 BEACON CT’, ‘INDIANAPOLIS’, ‘IN’, ‘46222’, ‘3172996868’);

Updating Existing Data

Modification of pre-existing data in a table is done with the UPDATE command. This command does not add new records to a table, nor does it remove records—UPDATE simply updates existing data. The update is generally used to update one table at a time in a database, but you can use it to update multiple columns of a table at the same time. An individual row of data in a table can be updated, or numerous rows of data can be updated in a single statement, depending on what's needed.

Updating the Value of a Single Column

The most simple form of the UPDATE statement is its use to update a single column in a table. Either a single row of data or numerous records can be updated when updating a single column in a table. The syntax for updating a single column follows:

WHERE statement is optional

update table_name set column_name = value [where condition];

Example - UPDATE ORDER SET QTY = 1 WHERE ORD_NUM = '23A16';

Updating Existing Data

Updating Multiple Columns in One or More Records

To update multiple columns with a single UPDATE statement, the syntax is

update table_name

set column1 = ‘value’, [column2 = ‘value’,] [column3 = ‘value’]

[where condition]; ***WHERE statement is optional***

Example - UPDATE ORDER SET QTY = 1, CUST_ID = ‘221’ WHERE ORD_NUM = ‘23A16’;

Update . . .

Roll	Name	Age	Branch
101	Vikas	19	
102	Soheb	20	
103	Gita	18	
104	Monalisa	19	
105	Ganesh	20	

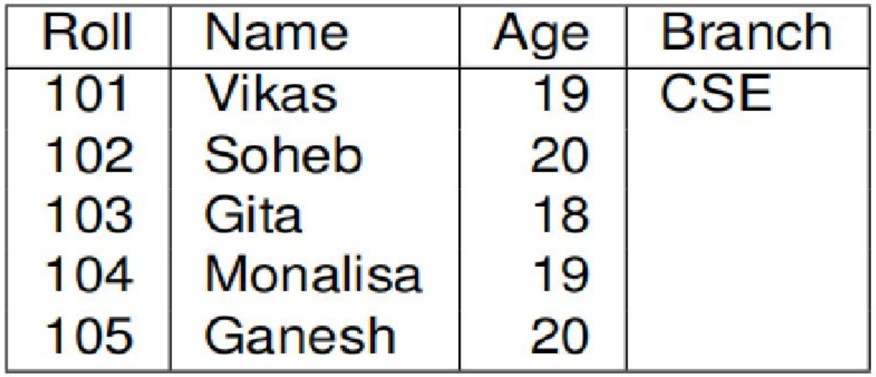
UPDATE Statement

UPDATE tablename SET columnname=value [WHERE *cond^n***];**

UPDATE Stud SET Branch='CSE' WHERE Roll=101;

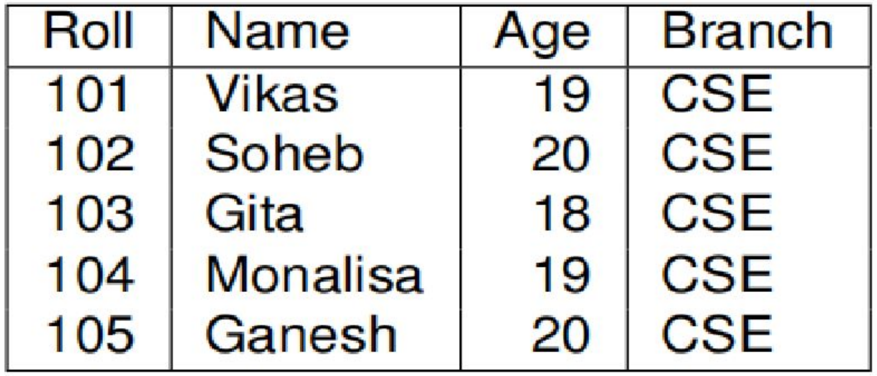
Roll	Name	Age	Branch
101	Vikas	19	CSE
102	Soheb	20	
103	Gita	18	
104	Monalisa	19	
105	Ganesh	20	

Update...



Roll	Name	Age	Branch
101	Vikas	19	CSE
102	Soheb	20	
103	Gita	18	
104	Monalisa	19	
105	Ganesh	20	

UPDATE Stud SET Branch='CSE';



Roll	Name	Age	Branch
101	Vikas	19	CSE
102	Soheb	20	CSE
103	Gita	18	CSE
104	Monalisa	19	CSE
105	Ganesh	20	CSE

Deleting Data from Table

The DELETE command removes entire rows of data from a table. It does not remove values from specific columns; a full record, including all columns, is removed. To delete a single record or selected records from a table, the syntax is:

delete from table_name

[where condition]; ***WHERE statement is optional***

Example - DELETE FROM ORDERS WHERE ORD_NUM = '23A16';

NOTE: WHERE clause is an essential part of the DELETE statement if you are attempting to remove selected rows of data from a table. DELETE statement without the use of the WHERE clause is rarely used & will delete all the records.

DELETE vs TRUNCATE

- The **DELETE** statement is used to delete rows from a table. **Example** -
  - **DELETE** FROM EMPLOYEE WHERE ID = 100;
  - **DELETE** FROM EMPLOYEE;
- The **TRUNCATE** statement is used to delete all the rows from the table and **free the space containing the table**. It is **DDL** in the sense that it commits and don't roll back. When we are using truncate, we are de-allocating the whole space allocated by the data without saving into the undo-table-space. But, in case of delete, we are putting all the data into undo table-space and then we are deleting all the data.
- Example – **TRUNCATE** TABLE EMPLOYEE;

DQL

Creating Table from another Table

CREATE TABLE tablename(column1,column2) AS SELECT column1,column2 FROM tablename;

CREATE TABLE Person(Roll, Name, Age) AS SELECT Roll, Name, Age FROM Stud;

The SQL statement populates the target table with data from the source table

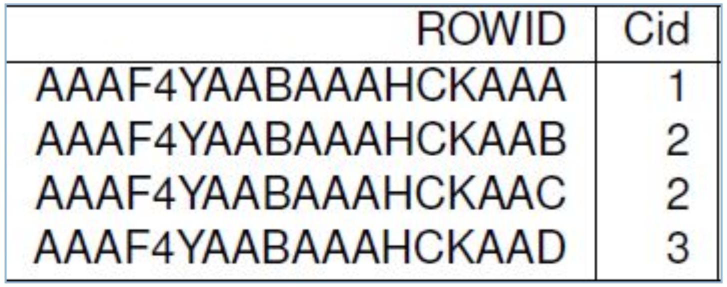
Inserting data into a Table from another Table

INSERT INTO tablename SELECT column1, column2 FROM tablename[WHERE *cond*^{n}**];**

INSERT INTO Person SELECT Roll, Name, Age FROM Stud WHERE Roll=101;
INSERT INTO Person SELECT Roll, Name, Age FROM Stud;

ROW ID

For each row in the database, the ROWID pseudo column (Oracle assigned field and not part of the table) returns the address of the row. Oracle rowid values contain information necessary to locate a row. It contains the physical address of a row in a database. If the user wants to delete the duplicate copies of the same record, then ROWID is used. **Example** - SELECT ROWID, Cid FROM CUSTOMER;



ROWID	Cid
AAAF4YAABAAAHCKAAA	1
AAAF4YAABAAAHCKAAB	2
AAAF4YAABAAAHCKAAC	2
AAAF4YAABAAAHCKAAD	3

ROW ID

ROWID

- Each row has a unique ROWID
- It is an 18-bit number and represented as a base-64 number
- It contains the physical address of a row in a database

In case user has inputted same records more than one time, ROWID is used to distinguish each record

For example, consider Customer table

Cid	CName	Address
1	Akash	BBS
2	Amir	BBS
2	Amir	BBS
3	Ashok	CTC

ROW ID

If the user wants to delete the duplicate copies of the same record, then ROWID is used

```
SELECT ROWID, Cid FROM Customer;
```

ROWID	Cid
AAAF4YAABAAAHCKAAA	1
AAAF4YAABAAAHCKAAB	2
AAAF4YAABAAAHCKAAC	2
AAAF4YAABAAAHCKAAD	3

```
DELETE FROM Customer WHERE ROWID=
'AAAF4YAABAAAHCKAAC';
```

Thank You
End of Lab 2

Assignment 2

DBMS Lab - Assignment 2 Q1: Create a new user making “your_name” as user-name and “your_surname” as the password. Q2: Grant all privileges to the newly created user. Q3: connect to the new user. Q4: Create a table employee with attributes emp_id, f_name , l_name , job_type, salary, commision, dept, and manager_id. Q5: Describe the table employee Q6: Add a new column doj to the employee table. Q7: Create a new table department with attributes d_name, d_loc, and hod_id. Q8: Create another table named location with attributes loc_id, city and contact_no. Q9. Enhance the size of city attribute in location table by 5. Q10. Delete the contact_no attribute in the location table. Q11. Rename the city attribute in the location table to address. Q12. Change the name of the table from location to loc.

Assignment 2..

Q13. Insert the following values into the loc table.

LOC_ID	ADDRESS
1	kolkata
2	mumbai

Q14. Show the values of location table. Q15. Delete all values and spaces consumed by loc table. Q16. Delete the loc table. Q17. Insert the following values into the department table.

D_Name	D_LOC	HOD_ID
sales	Kol	4
accounts	delhi	6
production	kol	1
marketing	kol	2
r&d	delhi	8

Assignment 2..

Q18. Insert the following values into the employee table.

EMP_ID	F_NAME	L_NAME	JOB_TYPE	SALARY	COMMISION	DEPT	MANAGER_ID	DOJ
1	arun	khan	manager	90000		production		04-JAN-1998
2	barun	kumar	manager	80000		marketing		09-FEB-1998
3	chitra	kapoor	engineer	60000		production	1	08-JAN-1998
4	dheeraj	mishra	manager	75000		sales	2	27-DEC-2001
5	emma	dutt	engineer	55000		production	1	20-MAR-2002
6	floki	dutt	accountant	70000		accounts		16-JUL-2000
7	dheeraj	kumar	clerk	40000		accounts	6	01-JUL-2016
8	saul	good	engineer	60000		r&d		06-SEP-2014
9	mou	bhat	clerk	30000		sales	4	08-MAR-2018
10	sunny	deol	salesman	20000	10000	marketing	2	31-MAR-01
11	bobby	deol	engineer	35000		r&d	8	17-OCT-17
12	amir	khan	salesman	15000	5000	marketing	2	11-JAN-13

Q19. Save the database.

Assignment 2..

Q20: Show all the attribute values of the department table. Q21: Display the department names and their locations. Q22: Show the employee's first name, last name, current salary and the salary with a 1000 rupees bonus. Q23: Show the employee's annual salary with a 1000 rupees yearly bonus and the annual salary with a 100 rupees monthly bonus. Q24: Show f_name as Name and annual salary as ANNSAL from the employee table. Q25: Show the L_name as SurName and 100 rupees incremented salary as NewSal from the employee table. Q26: Display the employees f_name and l_name joined together using the concatenation operator. Q27: Show the f_name, l_name and job_type as Employees. Q28: Show the employee details in the following fassion: Employees Details
-- arun khan is a manager
barun kumar is a manager ....

Assignment 2..

Q29: Show the monthly salary details in the following fassion: Monthly Salary Details
-- arun's monthly salary is 90000
……… Q30: Show the department names from the employee table.
Q31: Show the distinct department names from the employee table. Q32: Show the employees earning more than 50000. Q33. Show the employee's id's who are not working under manager id-1. Q34: Show the employee's names and salaries whose salary ranges between 40000 to 70000. Q35: Show the employees who work for manager id 1 or 6 or 8. Q36: Select the first names and salaries of those employee whose last name is khan. Q37: Select the first names and salaries of those employee whose last name starts with k. Q38: Select the first name, last name and salary of those employee whose last name starts with k and ends with r. Q39: Select the employees whose 3rd letter of their last name is o.

Assignment 2..

Q40: Select the employees who are not working under any manager. Q41: Select the employees who work as engineers with salary greater than 50000. Q42: Select the employees who work in the production department or earns more than 60000. Q43: Select those employees who are not managers or engineers or clerks. Q44: Select the employees who earns more than 49000 or less than 29000. Q45. Select the employees who don’t have an ‘o’ as the 2nd last letter of their last name. Q46. Select the employees who get commission. Q47. WAQ to display the current date. Q48. Show the total experience in weeks for all the employees. Q49. Find the employees working under employee_id 2. Q50. Delete the employees from sales department if they are not working as managers.

Assignment 2..

Q51. Insert the following two rows in the employee table without inserting any value in the department field.

EMP_ID	F_NAME	L_NAME	JOB_TYPE	SALARY	COMMISION	D_NAME	MANAGER_ID	DOJ
13	anand	patil	engineer	28000	2000		1	31-JAN-17
14	anandi	patel	clerk	12000	500		1	01-APR-17

Q52. . Insert the following two rows in the department table.

D_NAME	D_LOC	HOD_ID
----------	----------	----------
Admin	Mumbai	5
Transport	Mumbai	3

Q53. Update the employee table. Assign Anand to the admin department. Q54. Update the manager_id from 2 to 1 in the employee table. Q55. Display the employee details in descending order on their salary. Q56. Display the employee details in ascending order on their l_name. Q57. Delete the employees who are working as salesmen and having less experience than 15 years. Q58. Commit the database.