ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY

Kanyakumari Main Road, near Anjugramam, Palkulam, Anjugramam, Tamil Nadu 629401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATION COURSE

ON

STRUCTURED QUERY LANGUAGE

COURSE MATERIAL

STRUCTURED QUERY LANGUAGE

What is SQL?

- SQL stands for Structured Query language.
- SQL is the language to operate databases. Now, what is Database?
- Okay, let's go back and get from scratch….

Introduction to databases:

- A database is an organized collection of data.
- Data is basically information that exists in a variety of forms. Data is everywhere. People upload videos, take pictures, use several apps on their phones, search the web, and more. Machines, too, are generating and keeping more and more data.
- A database is a collection of information organized for easy access, management, and maintenance.

Examples:

- Telephone directory
- Customer data
- Product inventory
- Visitors' register
- Weather records

How does it work?

- Data is represented in terms of rows in RDBMS.
- It contains a number of tables, each with its primary key.

What is the table?

A database uses a table to store data in a structured way. A table is basically a collection of information in the form of data entries and contains rows and columns to store data.

Let us see a simple example of data stored in RDBMS.

ID	Name	Age	Department
101	Ross	23	CSE
201	Rio	21	Mechanical
301	Tina	23	Civil
302	Sergio	22	Mechanical

Remember, the table structure is a base to know in SQL.

What is the field?

A field will tell you about every record in a table.

For example, in the above student table, the field consists of id, name, age and department.

What is row or record?

- A row of a table is also called a record. It specifies some information about each individual entry in the table.
- It represents horizontally in the table.

For example, the row/record in the table is,

ID	Name	Age	Department

What is the column?

A column represents vertically in the table which contains information about the header which is the column name.

For example, from the above student table:

Age
23
21
23
22

NULL values:

The NULL value in the table means the field with “space”. It is different if it is filled as ‘0’ or if it contains space.

Data Integrity:

There are few categories of data integrity with each RDBMS:

· Entity integrity: This specifies that there should be no duplicate rows in a table.

· Domain Integrity: This specifies that there should be valid entries for a given column according to the type, the range of values and format etc.

· Referential Integrity: This specifies that rows cannot be deleted which are used by other rows/records in the table.

· Use defined Integrity: This specifies some business rules that are defined by users.

DBMS Operations:

- Adding new files
- Inserting data
- Retrieving data
- Modifying data
- Removing data
- Removing files

Introduction to RDBMS:

• A relational database refers to a database that stores data in a structured format, using rows and columns.

• It is “relational” because the values within each table are related to each other. Tables may also be related to other tables.

• The relational structure makes it possible to run queries across multiple tables at once.

Features of RDBMS:

- Every piece of information is stored in the form of tables
- Has primary keys for unique identification of rows
- Has foreign keys to ensure data integrity
- Provides SQL for data access
- Uses indexes for faster data retrieval
- Gives access privileges to ensure data security

Difference between DBMS and RDBMS:

DBMS	RDBMS
DBMS stands for Database management system	Relational database management system
DBMS stores data as file	RDBMS stores data in a tabular form.
DBMS is basically meant to deal with small data.	RDBMS is meant to handle large amounts of data.
It supports single user	It supports multiple users
Examples are XML, file systems etc	Examples are mysql, sql server, postgreSQL, oracle etc.

RDBMS vs Traditional Approach:

• The key difference is that RDBMS (relational database management system) applications store data in a tabular form, whereas in traditional approach, applications store data as files.

• There can be, but there will be no "relation" between the tables, like in a RDBMS. In traditional approach, data is generally stored in either a hierarchical form/navigational form. This means that a single data unit will have 0, 1 or more children nodes and 1 parent node.

SQL statement:

SQL statements are started with any

of the SQL commands/keywords like SELECT, INSERT, DELETE, DROP, UPDATE, ALTER

etc and it end with a semicolon;

Example explanation:

SELECT column_name from table_name;

Now let us know the reason for semicolon being used.

Why is semicolon used after SQL statements/commands?

Semicolon is used to separate SQL statements. It is the standard way of practice to separate SQL statements in a database system in which more than one SQL statements are used in the same call.

SQL Syntax:

There are various SQL syntax to be followed by a unique set of rules.

Keywords include Select, where, update order by, having etc.

Some of them are:

SQL Select:

Select col 1, col 2… col x

SQL Where:

Select co l1, col 2… colx

From table_name

Where condition;

SQL Process:

What happens when we execute a SQL query?

The system determines the best way to convey out our request and SQL engine figures out how to interpret the task.

Isn’t it a smarter idea to know how the internal flow of a process is?

I am curious to make you think that way and let us see the various components included in the process.

SQL Commands:

There are few important SQL commands to know about:

Select: It extracts data from a database.

Update: It updates data in a database.

Delete: It deletes data from a database.

Create table/database: It creates a new table/database.

Drop table/database: It deletes a table/database.

Alter table/database: It alters or modifies the table/database.

Insert into: It inserts new data into a database.

******Quite straight forward huh!***

Normalization:

- Decompose larger, complex table into simpler and smaller ones
- Moves from lower normal forms to higher normal forms.

Need for Normalization:

- In order to produce good database design
- To ensure all database operations to be efficiently performed
- Avoid any expensive DBMS operations
- Avoid unnecessary replication of information

Functional Dependency:

- Consider the relation

- Result (Student#, Course#, CourseName#, Marks#, Grade#)

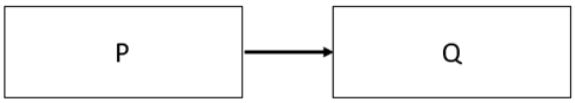
- Student# and course# together defines exactly one value of marks. Student#, course# ,Marks

- Student# and course# determines Marks or Marks is functionally dependent on student# and course#

- Other functional dependencies in the relation:

• Course# – CourseName

• Marks# – Grade• In a given relation R, P and Q are attributes. Attribute Q is functionally dependent on attribute P if each value of P determines exactly one value of Q.



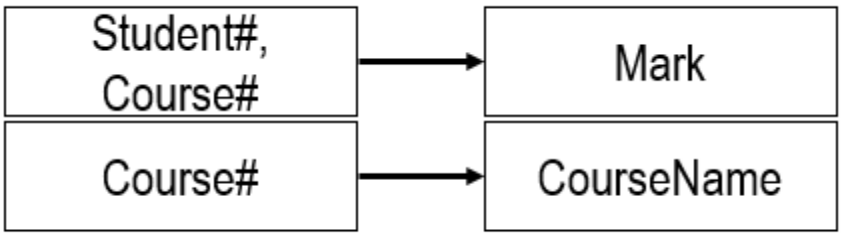
Functional Dependency Types:

- Partial Functional dependency
- Transitive Dependency

Partial Functional Dependency:

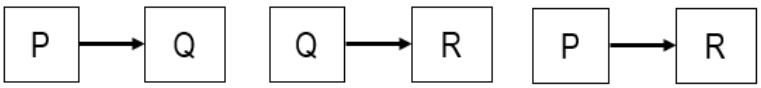
- Attribute Q is partially dependent on attribute P, if and only if it is dependent on the subset of attribute P.

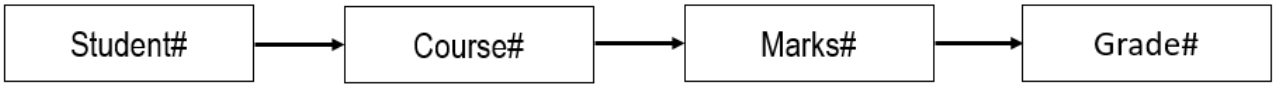
- REPORT (Student#, Course#, StudentName, CourseName, Marks, Grade)



Transitive Dependency:

P,Q, R are three attributes





Normalization: Types

- First Normal Form(1NF)
- Second Normal Form(2NF)
- Third Normal Form(3NF)

First Normal Form:

A relation schema is in 1NF, if and only if:

– All attributes in the relation are atomic(indivisible value)

-And there are no repeating elements or group of elements.

Example:

Student Marks Table

Student_Details	Course_details		Pre-requisite	Result_details		
101 Jack 11/4/1975	M1 Advance Math	17	Basic Math	03/11/2015	82	A
102 Rock 10/04/1976	P4 Advance Physics	18	Basic Physics	21/11/2015	83	A
103 Mary 11/07/1975	B3 Advance Biology	10	Basic Biology	12/11/2015	68	B
104 Roby 10/04/1976	H6 Advance History	19	Basic History	21/11/2015	83	A
105 Jim 03/08/1978	C3 Advance Chemistry	12	Basic Biology	12/11/2015	50	C

Student Marks Table in 1NF

Student #	Student_ Name	Date Of Birth	Course#	CourseName	Pre-Requisite	Duration in days	Date Of Exam	Marks	Grade
101	Jack	11/4/1975	M1	Advance Math	Basic Math	17	02/11/ 2015	82	A
102	Roby	10/04/1976	P4	Advance Physics	Basic Physics	18	21/11/ 2015	83	A
103	Mary	11/07/1975	B3	Advance Biology	Basic Biology	10	12/11/ 2015	68	B
...	...	...	...	...	...	...	...	...	...

Second Normal Form – (2NF)

A relation is said to be in 2NF, if and only if:

It is in 1st Normal Form.

No partial dependency exists between non-key attributes and key attributes.

Student Marks Table in 1NF

Student#	Student_Name	DOB	Course#	CourseName	Pre Requisite	Duration in days	Date of Exam	Marks	Grade
101	Jack	11/4/1975	M1	Advance Math	Basic Math	17	02/11/2015	82	A
102	Rob	10/04/1976	P4	Advance Physics	Basic Physics	18	21/11/2015	83	A
103	Mary	11/07/1975	B3	Advance Biology	Basic Biology	10	12/11/2015	68	B

Second Normal Form:

Student table:

Student#	Student_Name	Date Of Birth
101	Jack	11/4/1975
102	Roby	10/04/1976
103	Mary	11/07/1975

Result table:

Student#	Course#	Marks	Grade
101	M1	82	A
102	P4	83	A
103	B3	68	B

Course table:

Course#	CourseName	Prerequisite	Durationindays	Date Of Exam

M1	Advance Maths	Basic Math	17	02/11/2015
P4	Advance Physics	Basic Physics	18	21/11/2015
B3	Advance Biology	Basic Biology	10	12/11/2015

Third Normal form(3NF):

3 Normal Form:

- A relation R is said to be in 3NF if and only if:
- It is in 2NF.

No transitive dependency exists between non-key attributes and key attributes through another non-key attribute.

Result_table:

Student#	Course#	Marks	Grade
101	M1	82	A
102	P4	83	A
103	B3	68	B

Student# ,Course# → Marks

Student#, Course# → Grade

Marks → Grade

Result table:

Student#	Course#	Marks
101	M1	82
102	P4	83
103	B3	68

Marks Grade table:

Marks	Grade
82	A
83	A
68	B

Advantages and Disadvantages of Normalization:

Advantages:

- Based on the mathematical foundation
- Removes the redundancy to a large extent
- After 3NF, data redundancy is minimized to the extent of foreign keys
- Removes the anomalies present in INSERTs, UPDATEs and DELETEs

Disadvantages:

- Data retrieval or SELECT operation performance will be severely affected

- Normalization might not always represent real-world scenarios

Now we know what is the brief scenario and working of databases:

Lets dig more on SQL..

Why SQL?

SQL is widely popular because it offers the following advantages:

- Allows users to communicate i.e., access and manipulate the database.
- Allows users to retrieve data from a database.
- Allows users to create, update, modify and delete the database

SQL is a language for defining the structure of a database.

SQL Data Types:

1. Numeric – bit, tinyint, smallint, int, bigint, decimal, numeric, float, real
2. Character/String – Char, Varchar, Varchar(max), Text
3. Date/Time – Date, Time, Datetime, Timestamp, Year
4. Unicode character/String -Nchar, Nvarchar, Nvarchar(max), NText
5. Binary- Binary, Varbinary, Varbinary(max), image
6. Miscellaneous- Clob, Blob, Json, XML

SQL Constraints:

Constraint	**Description**
Not Null	Ensures that a column has a NULL value.
Default	Provides a default value for a column when none is specified.
Unique	Ensures that all the values in a column are different.

Primary	Identifies each row/record in a database table uniquely.
Foreign	Identifies each row/record in any database table uniquely.
Check	Ensures that all values in a column satisfy certain conditions.
Index	Creates and retrieves data from the database very quickly.

SQL Database: #thebase

SQL CREATE database:

This SQL statement is used to create a database.

Syntax:

Create database database_name;

Ex: Create database testdb;

SQL DROP database:

This command is used to delete/remove indexes from a table in the database.

Syntax:

Drop database database_name;

Ex: Drop database testdb;

NOTE: If you delete or drop the database, all the tables and views will be deleted. So be careful while using this command. Also there is a slight difference between drop and delete command. You will find it in further lessons.

SQL RENAME database:

This statement/command is used to change the name of your database

Syntax:

Rename database old_dbname to new_dbname;

Ex: Rename testdb to newdb;

SQL Select Database:

Here, we need to select a database first before executing any query on table/database.

Syntax:

Use database database_name;

Ex: Use database newdb;

SQL Keys:

SQL PRIMARY key:

A column/columns is called the primary *key* that uniquely identifies each row in the table.

When in case of create/alter table we can define primary key constraint if we want to create the primary key. If there are multiple columns used as a primary key, then it is called a composite primary key.

Primary key is by default NOT NULL, INDEXED an UNIQUE!

Syntax:

Create table students(ID int NOT NULL,

Name varchar(255) NOT NULL,

Age int NOT NULL,

Department varchar(255)

Primary key(ID));

SQL Primary key on Alter table:

Alter table students

Add Primary key(ID)

NOTE: When we use ALTER TABLE statement to add a primary key, the primary key column must NOT contain NULL values(when the table was initially created)

SQL FOREIGN key

In Relational databases, a foreign key is a field or a column that is used to develop a link between two tables.

To simply say, a foreign key in one table used to point to the primary key in another table.

Let's take an example:

First table:

Student_id	Name	Age	Department
101	Ross	23	CSE
201	Rio	21	Mechanical
301	Tina	23	Civil
302	Sergio	22	Mechanical

Second table:

Faculty_ID	Faculty_Name	Student_id
401	Jack	201
402	Leo	201
403	Harry	301
404	Oliver	101

Here you see that “Student_id” in the second table points to the “Student_id” in the first table.

- The “Student_id” column in the first table is the PRIMARY key in the first table.
- The “Student_id” column in the second table is a FOREIGN key in the second table.

SQL FOREIGN key constraint on create table:

Create table Faculty(faculty_ID int NOT NULL,

Faculty_name varchar(255) NOT NULL,

Student_id int NOT NULL

Primary key(Student_id),

Foreign key(Student_id) References Students(Student_id)

Differences between PRIMARY KEY and FOREIGN KEY:

Below are some important difference between primary key and foreign key in SQL-

- Primary key cannot be null vs foreign key can be null.
- Primary key is always unique vs foreign key can be duplicated.
- Primary key uniquely identifies a record in a table vs foreign key is a field in a table that is primary key in another table.

- There is only one primary key in the table vs we can have more than one foreign key in the table.
- By default primary key adds a clustered index vs foreign key does not automatically create an index, clustered or non-clustered.

SQL COMPOSITE Key:

A composite key is a combination of two or more fields or columns of a given table.

Syntax:

Create table table_name(col 1 int,

col2 int,

col3 varchar(50),

Primary key (col1, col2));

SQL UNIQUE Key:

A Unique key is a set of one or more than one fields/columns of a table that uniquely identifies a record in a database/table.

The Unique key and primary key both provide uniqueness for a column or set of columns.

SQL ALTERNATE Key:

Alternate key is the secondary key.

If a table has more than one candidate key, one of them will be primary and others will be alternate keys.

Basically, an alternate key is just a candidate key that has not been selected as the primary key.

SQL Table:

CREATE table

SQL CREATE table is used to create tables in a database.

Syntax:

Create table table_name

(column 1 “data type”,

column 2 “data type”,

.

.

column x “data type”);

DROP table:

Used to delete table information from the table.

Syntax:

DROP table “table_name”;

Ex:

DROP table student;

SQL DELETE table:

Used to delete rows from a table. If specific row is to be deleted, then we use WHERE condition to filter the rows in the table

Syntax:

-DELETE from table_name;

-DELETE from table_name [WHERE condition];

Ex: Delete from student;

Delete from student where student_id =2;

SQL ALTER table:

Used to add, modify or delete columns in an existing table

Syntax:

ADD –

ALTER table table_name

ADD(col 1 col definition,

col 2 col definition,

.

.

col x col definition);

MODIFY:

ALTER table table_name

MODIFY(col 1 col definition,

col 2 col definition,

.

.

col x col definition);

DROP:

ALTER table table_name DROP column column_name;

SQL RENAME table:

Used to change/modify the name of the table.

Syntax:

ALTER table table_name;

REAME to new_table_name;

Subsets of SQL:

SQL Command Groups:

- DDL (Data Definition Language) : creation of objects
- DML (Data Manipulation Language) : manipulation of data
- DCL (Data Control Language) : assignment and removal of permissions• TCL (Transaction Control Language) : saving and restoring changes to a database

DDL – Data definition Language:

Command	**Description**
Create	Creates objects in the database/database objects
Alter	Alters the structures of the database/ database objects

Drop	Deletes objects from the database
Truncate	Removes all records from a table permanently
Rename	Renames an object

DDL – Create command:

CREATE TABLE employees (

emp_id INT (10) NOT NULL,

first_name VARCHAR(10),

last_name VARCHAR(10) NOT NULL,

salary int(10) NOT NULL,

PRIMARY KEY (emp_id));

ALTER command:

ALTER TABLE employees ADD COLUMN contact INT(10);

Rename command:

ALTER TABLE employees RENAME COLUMN contact TO job_code;

Truncate Command:

TRUNCATE TABLE employees;

Drop Command:

DROP TABLE table_name;

DROP TABLE employees;

DML – Data Manipulation command:

Command	**Description**
Insert	Insert data into a table
Update	Updates existing data within a table
Delete	Deletes specified/all records from a table

DML – **Insert command:**

INSERT INTO employees (emp_id,first_name,last_name,salary) VALUES (101, 'Steven', 'King', 10000);

INSERT INTO employees (emp_id,first_name,last_name,salary) VALUES (102, 'Edwin', 'Thomas', 15000);

INSERT INTO employees (emp_id,first_name,last_name,salary) VALUES (103, 'Harry', 'Potter', 20000);

Update command:

UPDATE employees

SET last_name='Cohen'

WHERE emp_id=101;

Delete command:

DELETE FROM employees WHERE emp_id IN (101,103);

DCL – Data control language:

Command	Description
Grant	Gives access privileges to database
Revoke	Withdraws access privileges given with the grant command

TCL – Transaction control language: Used to manage transactions in the database**.**

Command	Description
Commit	Saves the work done
Rollback	Restores database to origin state since the last commit
Savepoint	Identify a point in a transaction to which you can roll back later

SQL Operators:

SQL statements generally contain some reserved word or a character in a 'where' clause to perform operations such as comparison and logical operations.

Time to revise school lessons :p

SQL Arithmetic Operators:

Operator	Description
Addition(+)	It is used to add containing values of both operands
Subtraction(-)	Its subtracts right hand operand from left hand operand
Multiplication(*)	It multiply both operands values
Division(/)	It divides left hand operand by right hand operand

Percentage(%)	It divides left hand operand by right hand operand and returns reminder

If you are wondering right hand operand and left hand operand, the see below illustration:

X+Y

*X*Y*

Here, X is left operand and Y is right operand.

SQL Logical Operators:

Operator	**Description**
AND	In SQL's where condition, it compares a value to all values with other values.TRUE if all the conditions separated by AND is TRUE
OR	In SQL's where condition, it compares a value to all values with other values.TRUE if all the conditions separated by OR is TRUE
LIKE	The LIKE operator is used to compare a value to similar values using wildcard operators.Checks an attribute value matches a given string pattern
BETWEEN	Checks an attribute value within range. Given the maximum value and minimum value.
NOT	Displays a record if the condition(s) is NOT TRUE
IN	TRUE if the operand is equal to one of a list of expressions.Checks an attribute value matches any value within a value list
ISNULL	The NULL operator is used to compare a value with a NULL value.

Operator	IllustrativeExample	Result
AND	(5<2) AND (5>3)	FALSE
OR	(5<2) OR (5>3)	TRUE
NOT	NOT(5<2)	TRUE

Sample Queries:

SELECT * FROM employees WHERE salary between 10000 and 20000;

SELECT * FROM employees WHERE first_name like ‘Steven’;

SELECT * FROM employees WHERE salary is null;

SELECT * FROM employees where salary in (10000,12000,20000);

SELECT DISTINCT(first_name) from employees;

SQL Comparison operators:

Operator	Description
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

<> or !=	Not equal to

Sample Queries:

SELECT * FROM employees WHERE first_name = ‘Steven’ AND salary <=10000;

SELECT * FROM employees WHERE first_name = ‘Steven’ OR salary >=10000;

SELECT * FROM employees WHERE first_name = ‘Steven’ and salary <>10000;

SQL Aggregate Functions:

Function	**Description**
Avg()	Returns the average value from specified columns
Count()	Returns number of table rows
Max()	Returns largest value among the records
Min()	Returns smallest value among the records
Sum()	Returns the sum of specified column values

Sample Queries:

SELECT avg(salary) FROM employees;

SELECT count(*) FROM employees;

SELECT min(salary) FROM employees;

SELECT max(salary) FROM employees;

SELECT sum(salary) FROM employees;

SQL Clauses:

SQL WHERE Clause:

WHERE Clause :

- Used to specify a condition while fetching the data from a single table or by joining with multiple tables.

- Not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc.,

e.g.

SELECT * FROM employees WHERE emp_id=101;

The example mentioned above extracts all the columns from the table ‘employees’ whose emp_id=101

GROUP BY clause:

- Arrange identical data into groups.

- This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause if used.

e.g.,

SELECT SUM(salary),dept_id

FROM employees

WHERE salary >=15000

GROUP BY dept_id

HAVING CLAUSE

- Used with aggregate functions due to its non-performance in the WHERE clause.
- Must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used.

e.g.,

```
SELECT AVG(salary),dept_id
FROM employees
WHERE salary >=10000
GROUP BY dept_id
HAVING count(dept_id)>=2
```

ORDER BY CLAUSE

- Used to sort output of SELECT statement
- Default is to sort in ASC (Ascending)
- Can Sort in Reverse (Descending) Order with “DESC” after the column name

e.g.,

```
SELECT * FROM employees
ORDER BY salary DESC;
```

SQL Set operators:

- Used to combine the results of two SELECT statements including duplicate rows.

- The same rules that apply to the UNION clause will apply to the UNION ALL operator.

SYNTAX:

SELECT a.col1,b.col2,…,a.coln FROM table1 a,table1 b WHERE a.commonfield = b.commonfield

UNION ALL

SELECT a.col1, b.col2,…, a.coln FROM table1 a, table1 b

WHERE a.commonfield = b.commonfield

UNION ALL

- Used to combine the results of two SELECT statements including duplicate rows.

- The same rules that apply to the UNION clause will apply to the UNION ALL operator.

SYNTAX:

SELECT a.col1,b.col2,…,a.coln FROM table1 a,table1 b WHERE a.commonfield = b.commonfield

UNION ALL

SELECT a.col1, b.col2,…, a.coln FROM table1 a, table1 b

WHERE a.commonfield = b.commonfield

SQL UNION

- Used to combine the result-set of two or more SELECT statements removing duplicates

- Each SELECT statement within the UNION must have the same number of columns

- The selected columns must be of similar data types and must be in the same order in each SELECT statement

- More than two queries can be clubbed using more than one UNION statement

SQL JOINS

Combine rows/columns from one or more tables, based on a related column between them in a database

- INNER JOIN − Returns rows when there is a match in both tables.

- LEFT JOIN − Returns all rows from the left table, even if there are no matches in the right table.

- RIGHT JOIN − Returns all rows from the right table, even if there are no matches in the left table.

- FULL OUTER JOIN − Returns rows when there is a match in one of the tables.

- SELF JOIN − Used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

- CARTESIAN JOIN (CROSS JOIN) − Returns the Cartesian product of the sets of records from the two or more joined tables.

SQL INNER JOIN

The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate.

SYNTAX :

SELECT table1.col1, table2.col2,…, table1.coln

FROM table1

INNER JOIN table2

ON table1.commonfield = table2.commonfield;

- Display details of employee and department
- dept_id is a common column between employees & departments tables
- ‘e’ and ‘d’ are alias for the table names

SQL LEFT JOIN

The LEFT JOIN returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.

SYNTAX :

SELECT table1.col1, table2.col2,…, table1.coln

FROM table1

LEFT JOIN table2

ON table1.commonfield = table2.commonfield;

- List all employees those with and without departments
- Department names will be NULL for those employees in which there are no departments

SQL RIGHT JOIN

- The RIGHT JOIN returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

SYNTAX :

SELECT table1.col1, table2.col2,…, table1.coln

FROM table1

RIGHT JOIN table2

ON table1.commonfield = table2.commonfield;

- List all departments those with and without managers
- For those departments where there are no managers, their names will be shown as NULL

SQL FULL OUTER JOIN

The FULL OUTER JOIN combines the results of both left and right outer joins. The joined table will contain all records from both the tables and fill in NULLs for missing matches on either side.

SYNTAX :

SELECT table1.col1, table2.col2,…, table1.coln

FROM table1

Left JOIN table2

ON table1.commonfield = table2.commonfield;

Union

SELECT table1.col1, table2.col2,…, table1.coln

FROM table1

Right JOIN table2

ON table1.commonfield = table2.commonfield;

SQL SELF JOIN

• The SELF JOIN joins a table to itself; temporarily renaming at least one table in the SQL statement.

SYNTAX:

SELECT a.col1, b.col2,…, a.coln

FROM table1 a, table1 b

WHERE a.commonfield = b.commonfield;

SQL CROSS JOIN

• The CROSS JOIN produces a result set with the number of rows in the first table multiplied by the number of rows in the second.

SYNTAX:

SELECT table1.col1, table2.col2,…, table1.coln

FROM table1

CROSS JOIN table2;

To help you master SQL perfectly, GL academy has launched free upskilling courses.