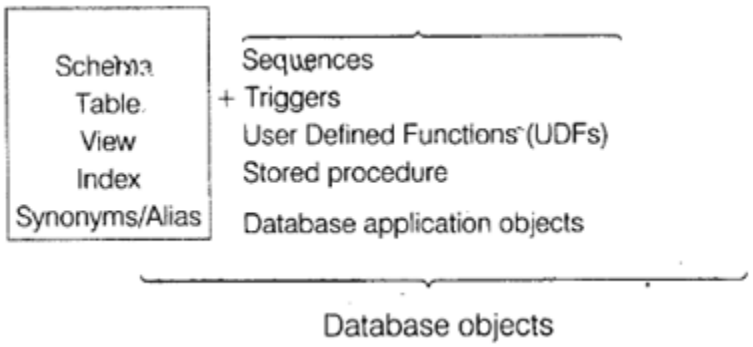
Study Notes on SQL

SQL is a declarative programming language that allows users to define and manipulate relational databases. It is used by database administrators, developers, and analysts to perform various operations on data. SQL follows a simple syntax and can be employed with popular database management systems such as MySQL, Oracle, Microsoft SQL Server, and PostgreSQL.

SQL operates on the principle of using declarative statements to specify what data to retrieve or modify, rather than specifying how to achieve it. The database management system (DBMS) interprets these statements and performs the necessary operations. SQL is an important part of the GATE CSE syllabus. Here we have provided study notes on SQL for the GATE exam to help you in preparation.

SQL

Structured Query language (SQL) is a language that provides an interface to relational database systems. SQL was developed by IBM in the 1970, for use in system R and is a defector standard, as well as an ISO and ANSI standard.



- To deal with the above database objects, we need a programming language and that programming language is known as SOL.

Three subordinate languages of SOL are

Data Definition Language (DDL)

It includes the commands as

- **CREATE** To create tables in the database.

- **ALTER** To modify the existing table structure:
- **DROP** To drop the table with table structure.
- **Data Manipulation Language** (DML) It is' used to insert, delete, update data and perform queries on these tables. Some of the DML commands are given below.
- **INSERT** To insert data into the table.
- **SELECT** To retrieve data from the table.
- **UPDATE** To-update existing data in the table.
- **DELETE** To delete data from the table.

Data Control Language (DCL)

It is used to control user's access to the database objects. Some of the DCL commands are:

- **GRANT** Used to grant select/insert/delete access.
- **REVOKE** Used to revoke the provided access

Transaction Control Language (TCL): It is used to manage changes affecting the data.

- **COMMIT** To save the work done, such as inserting or updating or deleting data to/from the table.
- **ROLLBACK** To restore the database to the original state, since the last commit.
- **SQL Data Types** SQL data types specify the type, size and format of data/information that can be stored in columns and variables.

Constraint Types with Description

Name of constraint	Table level/column level/Row level/External level	Description
NOT NULL	Column level	Restricts a column by making it mandatory to have some value.
Unique	Table level	Checks whether a column value will be unique among all rows in a table
Primary key	Column level and Table level	Checks whether a column value will be unique among all rows in a table and disallows NULL values.
Check Constraint	Column level Row level External level	Restrict a column value to a set of values defined by the constraint.
Foreign key constraint	Column level External level	Restrict the values that are acceptable in a column or group of columns of a table to those values found in a listing of the column/group of columns used to define the primary key in other table.

Default Constraint: It is used to insert a default value into a column, if no other value is specified at the time of insertion.

Syntax

CREATE TABLE Employee

{

Emp_idint NOT NULL,

Last_Name varchar (250),

City varchar (50)OEFAULT *BENGALURU*

}

DDL Commands

1. CREATE TABLE < Tab1e_Name> { Co1umn_name 1< data_type >, Column_name 2 < d'lta_type > }
2. ALTER TABLE < Table_Name> ALTER Column < Column_Name> SET NOT NULL
3. RENAME < object_type >object_name > to <new_name>

4. DROP TABLE <Table_Name>

DML Commands

SELECT A_1, A_2, A_3……,A_n what to return

FROM R_1, R_2, R_3, ….., R_m relations or table

WHERE condition filter condition i.e., on what basis, we want to restrict the outcome/result.

If we want to write the above SQL script in the form of relational calculus, we use the following syntax

$$\left\{\prod_{A_1,\ldots,A_n}\left(\sigma_{\text{condition}}\left(R_1 \times R_2 \times \ldots\ldots \times R_m\right)\right)\right\}$$

Comparison operators which we can use in filter condition are (=, >, <, > = , < =, < >,) '< >' means not equal to.

INSERT Statement: Used to add row (s) to the tables in a database

INSERT INTO Employee (F_Name, L_Name) VALUES ('Atal', 'Bihari')

UPDATE Statement: It is used to modify/update or change existing data in a single row, group of rows or all the rows in a table.

Example: //Updates some rows in a table. UPDATE Employee SET City = 'LUCKNOW' WHERE Emp_Id BETWEEN 9 AND 15; //Update city column for all the rows UPDATE Employee SET City='LUCKNOW';

DELETE Statement

This is used to delete rows from a table,

Example:

//Following query will delete all the rows from Employee table DELETE Employee Emp_Id=7; DELETE Employee

ORDER BY Clause: This clause is used to sort the result of a query in a specific order (ascending or descending), by default sorting order is ascending.

SELECT Emp_Id, Emp_Name, City FROM Employee

WHERE City = ‘LUCKNOW’

ORDER BY Emp_Id DESC;

GROUP BY Clause: It is used to divide the result set into groups. Grouping can be done by a column name or by the results of computed columns when using numeric data types.

- The HAVING clause can be used to set conditions for the GROUPBY clause.
- HAVING clause is similar to the WHERE clause, but having puts conditions on groups.
- WHERE clause places conditions on rows.
- WHERE clause can’t include aggregate: function, while HAVING conditions can do so.

Example:

SELECT Emp_Id, AVG (Salary)

FROM Employee

GROUP BY Emp_Id

HAVING AVG (Salary) > 25000;

Aggregate Functions

SUM ( )	It returns total sum of the values in a column.
AVG ( )	It returns average of the values in a column.
COUNT ( )	Provides number of non-null values in a column.
MIN ( ) and MAX ( )	Provides lowest and highest value respectively in a column.
COUNT (*)	Counts total number of rows in a table.

Joins: Joins are needed to retrieve data from two tables' related rows on the basis of some condition which satisfies both the tables. Mandatory condition to join is that at least one set of column (s) should be taking values from the same domain in each table.

Inner Join: Inner join is the most common join operation used in applications and can be regarded as the default join-type. Inner join creates a new result table by combining column values of two tables (A and B) based upon the join-predicate. These may be further divided into three parts.

1. Equi Join (satisfies equality condition)
2. Non-Equi Join (satisfies non-equality condition)
3. Self Join (one or more columns assumes the same domain of values).

Outer Join: An outer join does not require each record in the two joined tables to have a matching record. The joined table retains each record-even if no other matching record exists.

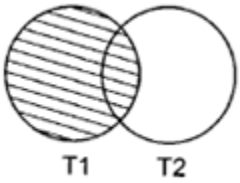
Considers also the rows from table (s) even if they don't satisfy the joining condition

(i) Right outer join (ii) Left outer join (iii) Full outer join

Left Outer Join: The result of a left outer join for table A and B always contains all records of the left table (A), even if the join condition does not find any matching record in the right table (B).

ID	Name
1	Ram
2	Shyam
T1	

ID	Branch
1	IT
3	CS
Left Outer Join	

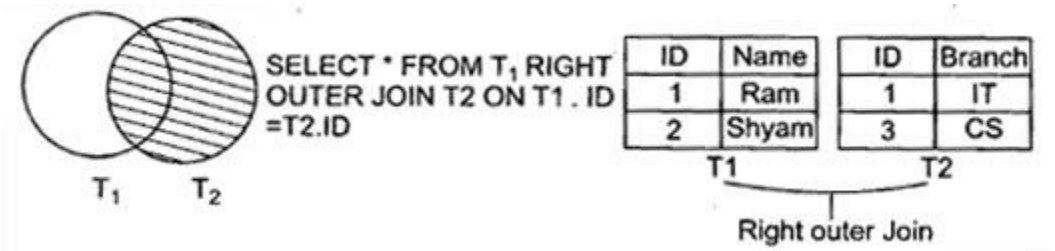



```
SELECT * FROM T1
Left Outer Join T2 ON T1.
ID = T2 . ID
```

Result set of T1 and T2

T1. ID	Name	T2. ID	Branch
1	Ram	1	IT
2	Shyam	NULL	NULL

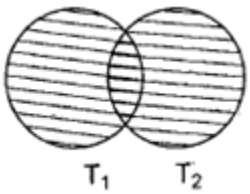
Right Outer Join: A right outer closely resembles a left outer join, except with the treatment of the tables reversed. Every row from the right table will appear in the joined table at least once. If no matching with the left table exists, NULL will appear.



Result set of T1 and T2

T1. ID	Name	T2. ID	Branch
1	Ram	1	IT
NULL	NULL	3	CS

Full Outer Join: A full outer join combines the effect of applying both left and right outer joins where records in the FULL OUTER JOIN table do not match, the result set will have NULL values for every column of the table that lacks a matching row for those records that do match, as single row will be produced in the result set.




```
SELECT * FROM T1 FULL OUTER
JOIN T2 ON T1 . ID=T2.ID
```

Result set of T1 and T2 (Using tables of the previous example)

T1. ID	Name	T2. ID	Branch
1	Ram	1	IT
2	Shyam	NULL	NULL
NULL	NULL	3	CS

Cross Join (Cartesian product): Cross join returns the Cartesian product of rows form tables in the join. It will produce rows which combine each row from the first table with each row from the second table.

Select * FROM T1, T2

Number of rows in result set = (Number of rows in table 1 × Number of rows in table 2)

Result set of T1 and T2 (Using previous tables T1 and T2)

ID	Name	ID	Branch
1	Ram	1	IT
2	Shyam	1	IT
1	Ram	3	CS
2	Shyam	3	CS

Thanks!

BYJU'S Exam Prep - The Most Comprehensive App!