CS3CRT06 : Database Management Systems (Core)

Syllabus: Unit III: SQL
Data Types - Data Definition commands : CREATE, ALTER DROP - Adding constraints in SQL - Basic SQL Queries : INSERT SELECT DELETE UPDATE - Substring comparison using LIKE operator ,BETWEEN operator - Ordering of rows - SQL set operations :UNION, EXCEPT, INTERSECT - Complex Queries : Comparison involving NULL and Three-valued logic .Nested queries . EXISTS and UNIQUE functions, Renaming of attributes and Joining of tables, Aggregate functions Grouping - Managing Views.

Unit III : Basics of SQL

Query language (QL) refers to any computer programming language that requests and retrieves data from database and information systems by sending queries.
The name SQL is expanded as Structured Query Language. Originally, SQL was called SEQUEL (Structured English QUEry Language) and was designed and implemented at IBM Research as the interface for an experimental relational database system called SYSTEM R. SQL is now the standard language for commercial relational DBMSs. A joint effort by the American National Standards Institute (ANSI) and the International Standards Organization (ISO) has led to a standard version of SQL.
SQL is a comprehensive database language: It has statements for data definitions, queries, and updates. Hence, it is both a DDL and a DML. In addition, it has facilities for defining views on the database, for specifying security and authorization, for defining integrity constraints, and for specifying transaction controls.

Data types in SQL

The basic data types available in SQL are

1. numeric
2. character string
3. bit string
4. Boolean
5. date and
6. time

1. Numeric data types include integer numbers of various sizes and and floating-point (real) numbers of various precision

● INTEGER or INT	- used for integer numbers
● SMALLINT	- used for small integer numbers
● FLOAT or REAL	- used for real numbers
● DOUBLE PRECISION	- used for real numbers
● DECIMAL(m,n) or DEC(m,n)	- used for real numbers
● NUMBER(m,n)	- used for real numbers
● NUMERIC(m,n)	- used for real numbers

—where m is the total number of decimal digits and n is the number of digits after the decimal point.

Eg: 1) Roll_Number INTEGER

2) RN NUMBER(2)

3) Internal_Mark NUMBER(5,2)

2. **Character-string data types** are

- CHAR(n) or CHARACTER(n) - used for Fixed length character strings.
- VARCHAR(n) - used for variable length character strings.
- CLOB - CHARACTER LARGE OBJECT. It is used for columns that have large text values, such as documents. The CLOB maximum length can be specified in kilobytes (K), megabytes (M), or gigabytes (G).

Eg: 1) Name CHAR(10)

- if the value 'Smith' is given for an attribute of type CHAR(10), it is padded with five blank characters to become 10 character length.

2) Name VARCHAR(10)

- if the value 'Smith' is given for an attribute of type VARCHAR(10), it is not padded with blank characters. Its length will be 5 only.

3) CLOB(20M) specifies a maximum length of 20 megabytes.

3. **Bit-string data types** are

- BIT(n) - used for fixed length n bits string
- BIT VARYING(n) - used for varying length bit string, where n is the maximum number of bits. The default for n, the length of a character string or bit string, is 1. Literal bit strings are placed between single quotes but preceded by a B to distinguish them from character strings.Eg: B'10101'.
- BLOB - BINARY LARGE OBJECT is used to
- Another variable-length specify columns that have large binary values, such as images. Like CLOB, the maximum length of a BLOB can be specified in kilobits (K), megabits (M), or gigabits (G). Eg: BLOB(30G)

4. **Boolean data type** has the values of TRUE or FALSE.

5. **DATE data type** has ten positions, and its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD.

6. **TIME data type** has at least eight positions, with the components HOUR, MINUTE, and SECOND in the form HH:MM:SS. Only valid dates and times should be allowed by the SQL implementation.

Some additional data types are

- TIMESTAMP - It includes the DATE and TIME fields, plus a minimum of six positions for decimal fractions of seconds and an optional WITH TIME ZONE qualifier. Eg: TIMESTAMP '2008-09-27 09:12:47.648302'.
- INTERVAL - It is an interval used to increment or decrement the value of a date, time, or timestamp. Intervals are qualified to be either YEAR/MONTH intervals or DAY/TIME intervals.

Note: The remaining portion was dictated in your Offline classes. Refer to your notes.

Integrity Constraints in SQL: Integrity constraints guard against accidental damage to the database. It ensures data consistency. The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value.
- UNIQUE - Ensures that all values in a column are different.
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. It Uniquely identifies each row in a table
- FOREIGN KEY - Uniquely identifies a row/record in another table
- CHECK - Ensures that all values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column when no value is specified

Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

Syntax: CREATE TABLE table_name (column1 datatype constraint, ...);

```
Eg: CREATE TABLE Persons ( ID number(6)  PRIMARY KEY, Adar number(12) UNIQUE,
                           Name varchar2(15) NOT NULL,
                           Age number(2) CHECK (Age>=18),
                           City varchar(25) DEFAULT 'Kottayam');
```

FOREIGN KEY is a key used to link two tables together. A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table. The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

```
Eg: CREATE TABLE Orders ( OrderID number(6), PersonID int,
                          PRIMARY KEY (OrderID),
                          FOREIGN KEY (PersonID) REFERENCES Persons(PersonID));
```

The PersonId attribute in Orders table is a foreign key referencing Persons table. Only values occurring in the primary key attribute of the referenced relation may occur in the foreign key attribute of the referencing relation. When a foreign key constraint is violated, the normal procedure is to reject the action that caused the violation. The two additional specifications are:

- on delete cascade : This would delete the records that depended on the key.
- on update cascade: This would update all the records that depended on the key.

Primay key and Unique are entity integrity constraints. Foreign key is referential integrity constraint. NOT NULL, CHECK and DEFAULT are domain integrity constraints.

Set Operations

The set operations union, intersect, and except operate on relations and correspond to the relational algebra operations ∪, ∩, −. Ex:

1. Find all customers who have a loan, an account, or both:

(select customer_name from depositor) union (select customer_name from borrower)

2. Find all customers who have both a loan and an account:

(select customer_name from depositor) intersect (select customer_name from borrower)

3. Find all customers who have an account but no loan.

(select customer_name from depositor) except (select customer_name from borrower)

Aggregate Functions:

These functions operate on the values of a column of a relation, and return a value
The MIN() function returns the smallest value of the selected column.
The MAX() function returns the largest value of the selected column.
The COUNT() function returns the number of rows that matches a condition.
The AVG() function returns the average value of a numeric column.
The SUM() function returns the total of a numeric column.

Subqueries: A subquery is a select-from-where expression that is nested within another query.

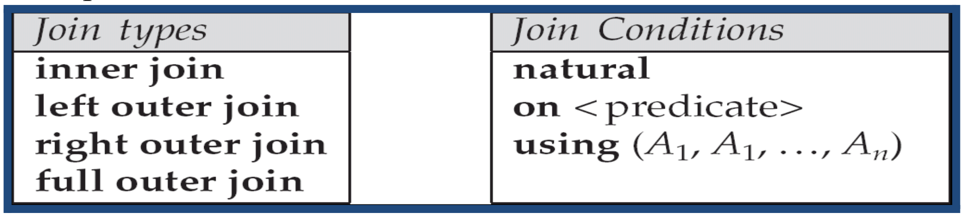
Ex: Find all customers who have both an account and a loan at the bank:
select distinct customer_name from borrower where customer_name in (select customer_name from depositor)

Views:
A view provides a mechanism to hide certain data from the view of certain users. A view is defined using the create view statement which has the form

create view v as query expression

The view name is represented by v.
Ex: Create a view of all loan data in the loan relation, hiding the amount attribute
create view branch_loan as select branch_name, loan_number from loan
SQL implementations allow updates on simple views (without aggregates) defined on a single relation.

Join:
Join operations take two relations and return as a result another relation.



Join types	*Join Conditions*
inner join	**natural**
left outer join	**on** < predicate>
right outer join	**using** $(A_1, A_1, \ldots, A_n)$
full outer join	

Ex: loan inner join borrower on loan . loan_number = borrower . loan_number