19CSE202 DATABASE MANAGEMENT SYSTEMS

LAB 3

Ms. Vidhya. S
Assistant Professor(Sr. Gr)
Department of Computer Science and Engineering
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
Coimbatore

Table Constraints

- A CONSTRAINT clause is an optional part of a CREATE TABLE statement or ALTER TABLE statement.
- A constraint is a rule to which data must conform. Constraint names are optional.
- A CONSTRAINT can be one of the following:
  - a column-level constraint: Column-level constraints refer to a single column in the table and do not specify a column name (except check constraints). They refer to the column that they follow.
  - a table-level constraint: Table-level constraints refer to one or more columns in the table. Table-level constraints specify the names of the columns to which they apply. Table-level CHECK constraints can refer to 0 or more columns in the table.
- By default, SQL tables have no constraints
  - Can insert multiple copies of a given row.
  - Can insert rows with NULL values in any column.

CONSTRAINTS

- **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. 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
- **INDEX** - Used to create and retrieve data from the database very quickly

Column Constraints

- NOT NULL: Specifies that this column cannot hold NULL values (constraints of this type are not nameable).
- PRIMARY KEY: Specifies the column that uniquely identifies a row in the table. The identified columns must be defined as NOT NULL.

  Note: If you attempt to add a primary key using ALTER TABLE and any of the columns included in the primary key contain null values, an error will be generated and the primary key will not be added.

- UNIQUE: Specifies that values in the column must be unique.
- FOREIGN KEY: Specifies that the values in the column must correspond to values in a referenced primary key or unique key column or that they are NULL.
- CHECK: Specifies rules for values in the column.

Table Constraints

- PRIMARY KEY: Specifies the column or columns that uniquely identify a row in the table. NULL values are not allowed.
- UNIQUE: Specifies that values in the columns must be unique.
- FOREIGN KEY: Specifies that the values in the columns must correspond to values in referenced primary key or unique columns or that they are NULL.

  Note: If the foreign key consists of multiple columns, and *any* column is NULL, the whole key is considered NULL. The insert is permitted no matter what is on the non-null columns.
- CHECK: Specifies a wide range of rules for values in the table.

Column/Table Constraints

- Column constraints and table constraints have the same function; the difference is in where you specify them.
- Table constraints allow you to specify more than one column in a PRIMARY KEY, UNIQUE, CHECK, or FOREIGN KEY constraint definition.
- Column-level constraints (except for check constraints) refer to only one column.

PRIMARY KEY - Remark

ALTER TABLE ADD PRIMARY KEY allows you to include existing columns in a primary key if they were first defined as NOT NULL.

NULL values are not allowed. If the column(s) contain NULL values, the system will not add the primary key constraint.

UNIQUE - Constraints

- A UNIQUE constraint defines a set of columns that uniquely identify rows in a table only if all the key values are not NULL. If one or more key parts are NULL, duplicate keys are allowed.
- For example, if there is a UNIQUE constraint on `col1` and `col2` of a table, the combination of the values held by `col1` and `col2` will be unique as long as these values are not NULL.
- If one of `col1` and `col2` holds a NULL value, there can be another identical row in the table.
- A table can have multiple UNIQUE constraints.

Primary Key vs Unique Constraint

- Can specify columns that comprise primary key
- No two rows can have same values for primary key
- A table can have only one primary key
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- ***Can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.***

Primary Key - Syntax

```
CREATE TABLE account (
account_number CHAR(10),
branch_name VARCHAR(20),
balance NUMERIC(12, 2),
PRIMARY KEY (account_number)
);
```

- For multi-column primary keys, must specify primary key after column specifications

```
CREATE TABLE depositor (
customer_name VARCHAR(30),
account_number CHAR(10),
PRIMARY KEY (customer_name, account_number)
);
```

Primary key with constraint name

```
CREATE TABLE depositor (
customer_name VARCHAR(30),
account_number CHAR(10),
CONSTRAINT PK_depositor PRIMARY KEY (customer_name, account_number)
);
```

SQL PRIMARY KEY on ALTER TABLE

- ALTER TABLE Persons ADD PRIMARY KEY(ID)

  OR

- ALTER TABLE Persons
  ADD CONSTRAINT PK_Person PRIMARY KEY (FirstName,LastName);

DROP Constraint

- ALTER TABLE Persons
  DROP CONSTRAINT PK_Person;

CHECK Constraints

- Often want to specify other constraints on values.
- Can require values in a table to satisfy some predicate, using a CHECK constraint
    - Very effective for constraining columns' domains, and eliminating obviously bad inputs
- CHECK constraints must appear after the column specifications.

CHECK Constraint Example

- Can constrain values in a particular column:

```
CREATE TABLE employee (
emp_id INT PRIMARY KEY,
emp_ssn CHAR(9) NOT NULL UNIQUE,
emp_name VARCHAR(40) NOT NULL,
pay_rate NUMERIC(5,2) NOT NULL,
CHECK (pay_rate > 5.25)
);
```

- Ensures that all employees have a minimum wage

CHECK Constraint Example(2)

```
CREATE TABLE employee (
emp_id INT PRIMARY KEY,
emp_ssn CHAR(9) NOT NULL UNIQUE,
emp_name VARCHAR(40) NOT NULL,
status VARCHAR(10) NOT NULL,
pay_rate NUMERIC(5,2) NOT NULL,
CHECK (pay_rate > 5.25),
CHECK (status IN ('active', 'vacation', 'suspended'))
);
```

- Employee status must be one of the specified values

ALTER Table Command to ADD/DROP Constraints

```
ALTER TABLE employee
ADD CONSTRAINT pays CHECK (pay_rate > 5.25);
```

```
ALTER TABLE employee
DROP CONSTRAINT pays
```

DEFAULT Constraint

```
CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
City varchar(255) DEFAULT 'Mumbai'
);
```

FOREIGN KEY

```
CREATE TABLE Orders(
OrderID int NOT NULL,
OrderNumber int NOT NULL,
PersonID int,
PRIMARY KEY(OrderID),
CONSTRAINT FK_PersonOrder FOREIGN KEY(PersonID) REFERENCES
Persons(PersonID)
);
```

SELECT DISTINCT

- SELECT DISTINCT statement is used to return only distinct (different) values.
- Syntax: SELECT DISTINCT column1, column2 FROM table_name

WHERE CLAUSE

- WHERE clause can be combined with AND, OR, and NOT operators. (With Usual meaning for the operator)

AND

- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE *condition1* AND *condition2* AND *condition3 ...;*

OR

- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE *condition1* OR *condition2* OR *condition3 ...;*

NOT

- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE NOT *condition;*

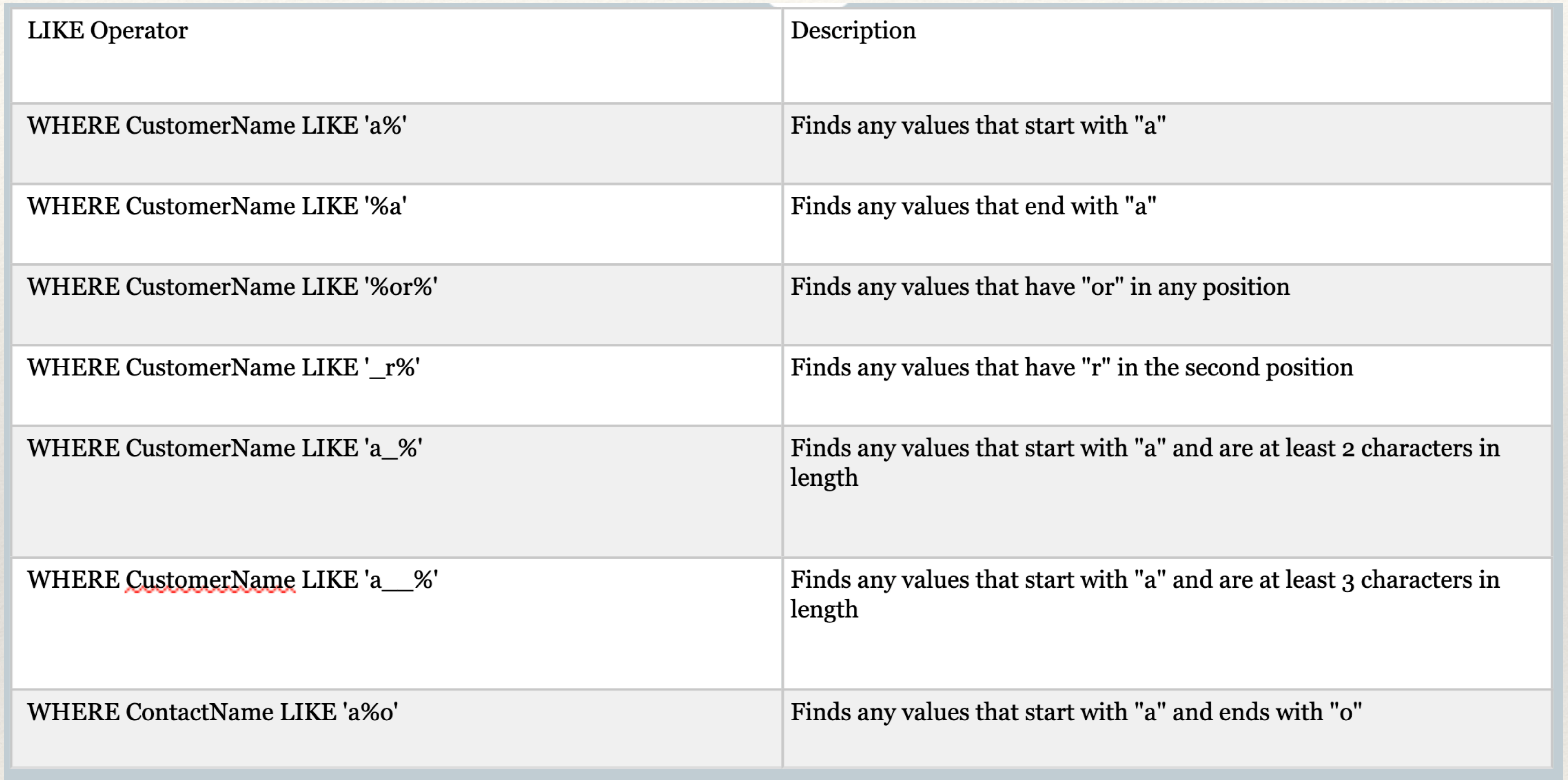
The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

Syntax:

- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *column_name* BETWEEN *value1* AND *value2;*

LIKE operator

- There are two wildcards often used in conjunction with the LIKE operator:
- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character



LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

Syntax:
```
SELECT column1, column2, ...FROM table_name
WHERE columnN LIKE pattern;
```

ORDER BY

- ORDER BY keyword is used to sort the result-set in ascending or descending order.
- SELECT column1, column2, ...
  FROM table_name
  ORDER BY column1, column2, ... ASC | DESC;
  Eg1:
- SELECT * FROM Customers
  ORDER BY Country DESC;
  Eg2:
- SELECT * FROM Customers
  ORDER BY Country ASC, CustomerName DESC;