SQL- DATA DEFINITION LANGUAGE

RESMI P D
BCA

DDL

INTRODUCTION

- To understand the SQL Data Definition Language
  - Create
  - Insert
  - Delete
  - Drop
  - Truncate
  - Alter

DDL

Creating a Database

- To initialize a new database:
- **Syntax:**

  CREATE DATABASE database_name
- There are numerous arguments that go along with this command but are database specific
- Only some databases require database to be created and space to be allocated prior to creation of tables.
- Some databases provide graphical user interfaces to create databases and allocate space.
  - Access only allows database to be created using User Interface

DDL

Creating a Table

- **Syntax**

```
CREATE TABLE table_name
(Column_name      datatype[(size)],
 Column_name      datatype[(size)],
)
```

- **Example**

```
CREATE TABLE books
(ISBN             char(20),
Title                    char(50),
AuthorID          Integer,
Price             float)
```

- Creates a table with four columns

DDL

Data Types

- Following broad categories of data types exist in most databases:
  - String Data
  - Numeric Data
  - Temporal Data
  - Large Objects

DDL

String Data

- **Fixed Length:**
- Occupies the same length of space in memory no matter how much data is stored in them.
- **Syntax:**

  char(n) where n is the length of the String

  e.g. name char(50)
- If the variable stored for name is ‘Sanjay’ the extra 43 fields are padded with blanks

DDL

String Data

- **Variable Length** string is specified with maximum length of characters possible in the string, however, the allocation is sized to the size of the data stored in memory.
- **Syntax:**

  Varchar(n) – n is the maximum length of data possible for the type
- There may be a restriction in the maximum length of the data that you can specify in the declaration which will vary according to the database.
- All character data has to be enclosed in single quotes during specification.

DDL

Numeric Data Types

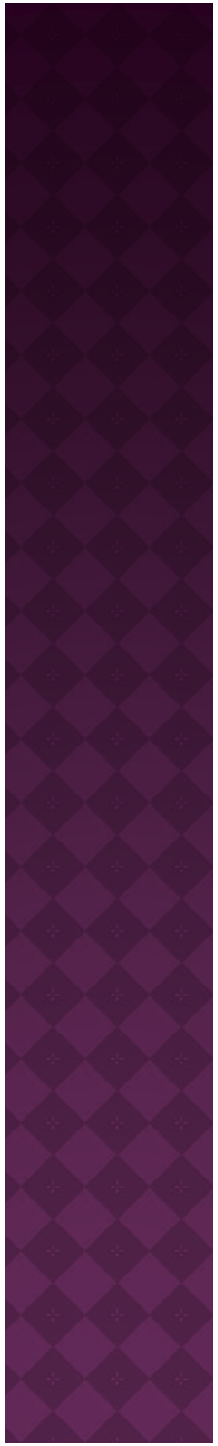
- Store all the data related to purely numeric data.
- Some numeric data may also be stored as a character field e.g. zip codes
- **Common Numeric Types:**
  - Decimal — Floating point number
  - Float — Floating point number
  - Integer(size) — Integer of specified length
  - Money — A number which contains exactly two digits after the decimal point
  - Number — A standard number field that can hold a floating point data

Note: Different databases name their numeric fields differently and may not support all numeric types. They may also support additional numeric types.

DDL

Temporal Data Types

- **These represent the dates and time:**
- Three basic types are supported:
  - Dates
  - Times
  - Date-Time Combinations



DDL

Large Data Objects

- These are used for storing data objects like files and images:
- **There are two types:**
  - Character Large Objects (clobs)
  - Binary Large Objects (blobs)

DDL

Specifying Keys- Introduction

- Unique keyword is used to specify keys.
  - This ensures that duplicate rows are not created in the database.
- Both Primary keys and Candidate Keys can be specified in the database.
- Once a set of columns has been declared unique any data entered that duplicates the data in these columns is rejected.
- **Specifying a single column as unique:**
- Example

```
CREATE TABLE Studios
(studio_id          Number,
name                    char(20),
city                varchar(50),
state               char(2),
UNIQUE (name))
```

- Here the name column has been declared as a candidate key

DDL

Specifying Keys- Multiple Columns

- Specifying multiple columns as unique:
- **Example:**

```
CREATE TABLE Studios
(studio_id          Number,
name                char(20),
city                varchar(50),
state                       char(2),
UNIQUE (name),
UNIQUE(city, state))
```

- Here both name & city/state combination are declared as candidate keys

DDL

Specifying Keys- Primary Key

- Specifying multiple columns as unique:
- To specify the Primary Key the **Primary Key** clause is used

- **Example:**

```
CREATE TABLE Studios
(studio_id        Number,
name                    char(20),
city              varchar(50),
state             char(2),
PRIMARY KEY (studio_id),
UNIQUE (name),
UNIQUE(city, state)
)
```

DDL

Specifying Keys- Single and MultiColumn Keys

- Single column keys can be defined at the column level instead of at the table level at the end of the field descriptions.
- MultiColumn keys still need to be defined separately at the table level

```
CREATE TABLE Studios
(studio_id     Number          PRIMARY KEY,
name           char(20)        UNIQUE,
city           varchar(50),
state                  char(2),
Unique(city, state))
```

- Note: Some databases require the use of Unique Index for specification of keys.

DDL

Specifying Keys- Foreign Keys

- References clause is used to create a relationship between a set of columns in one table and a candidate key in the table that is being referenced.

- **Example:**

```
CREATE TABLE Movies
(movie_title      varchar(40),
studio_id         Number REFERENCES Studios(studio_id))
```

- Creates a relationship from the Movies table to the Studios table

DDL

Constraints- Disallowing Null Values

Disallowing Null Values:

- Null values entered into a column means that the data in not known.
- These can cause problems in Querying the database.
- Specifying Primary Key automatically prevents null being entered in columns which specify the primary key

- Not Null clause is used in preventing null values from being entered in a column.

- **Example:**

```
CREATE TABLE Studios
( studio_id      number          PRIMARY KEY,
  name           char(20)                NOT NULL,
  city           varchar(50)     NOT NULL,
  state                  char(2)         NOT NULL)
```

- Null clause can be used to explicitly allow null values in a column also

DDL

Constraints- Value Constraints

Value Constraints:

- Allows value inserted in the column to be checked condition in the column constraint.

- Check clause is used to create a constraint in SQL

- **Example:**

```
CREATE TABLE Movies
(movie_title        varchar(40)        PRIMARY KEY,
 studio_id          Number,
 budget                                Number check (budget > 50000)
)
```

- Table level constraints can also be defined using the Constraint keyword

- **Example:**

```
CREATE TABLE Movies
(movie_title        varchar(40)        PRIMARY KEY,
 studio_id          Number,
 budget                                Number check (budget > 50000),
release_date        Date,
 CONSTRAINT release_date_constraint Check (release_date between '01-Jan-1980' and '31-
        dec-1989))
```

- Such constraints can be activated and deactivated as required.

DDL

Constraints- Default Value

Default Value:

- A default value can be inserted in any column by using the Default keyword.

- **Example:**

```
CREATE TABLE Movies (
movie_title       varchar(40)       NOT NULL,
release_date      date              DEFAULT sysdate          NULL,
genre                               varchar(20)       DEFAULT 'Comedy'
          Check genere In ('Comedy', 'Horror', 'Drama')
)
```

- Table level constraints can also be defined using the Constraint keyword
- release_date defaults to the current date, however Null value is enabled in the column which will need to be added explicitly when data is added.
- **Note:** Any valid expression can be used while specifying constraints

Modifying Records

Insert Statement

- **Insert:**
  - Allows you to add new records to the Table
- **Syntax:**
  - Insert into table_name[(column_list)] values (value_list)
- **Example:**

```
INSERT INTO studios
VALUES (1, 'Giant', 'Los Angeles', 'CA')

INSERT INTO studios
(studio_city, studio_state, studio_name, studio_id)
VALUES ('Burbank', 'CA', 'MPM', 2)
```

- **Notes1:** If the columns are not specified as in the first example the data goes in the order specified in the table
- **Notes2:** There are two ways of inserting Null values
  1. If the field has a default value of Null, you can use an Insert statement that ignores the column where the value is to be Null.
  2. You can specify the column in the column list specification and assign a value of Null to the corresponding value field.

Modifying Records

Select & Insert

- **Select & Insert:**
  - A select query can be used in the insert statement to get the values for the insert statement

- **Example:**

```
INSERT INTO city_state
SELECT studio_city, studio_state FROM studios
```

- This selects the corresponding fields from the studios table and inserts them into the city_state table.

- **Example:**

```
INSERT INTO city_state
SELECT Distinct studio_city, studio_state FROM studios
```

- This selects the corresponding fields from the studios table, deletes the duplicate fields and inserts them into the city_state table. Thus the final table has distinct rows

Modifying Records

Delete Statement

- **Delete Statement:**
  - is used to remove records from a table of the database. The where clause in the syntax is used to restrict the rows deleted from the table otherwise all the rows from the table are deleted.

- **Syntax:** DELETE FROM table_name [WHERE Condition]

- **Example:**

```
DELETE FROM City_State
WHERE state = 'TX'
```

- Deletes all the rows where the state is Texas keeps all the other rows.

Modifying Records

Update Statement

- **Update Statement:**
  - used to make changes to existing rows of the table. It has three parts. First, you ,must specify which table is going to be updated. The second part of the statement is the set clause, in which you should specify the columns that will be updated as well as the values that will be inserted. Finally, the where clause is used to specify which rows will be updated.

- **Syntax:**

```
UPDATE table_name
SET column_name1 = value1,  column_name2 = value2, …..
[WHERE Condition]
```

- **Example:**

```
UPDATE studios
SET studio_city = 'New York', studio_state = 'NY'
WHERE studio_id = 1
```

- **Notes1**: If the condition is dropped then all the rows are updated.

Modifying Records

Truncate Statement

- **Truncate Statement:**
  - used to delete all the rows of a table. Delete can also be used to delete all the rows from the table. The difference is that delete performs a delete operation on each row in the table and the database performs all attendant tasks on the way. On the other had the Truncate statement simply throws away all the rows at once and is much quicker. The note of caution is that truncate does not do integrity checks on the way which can lead to inconsistencies on the way. If there are dependencies requiring integrity checks we should use delete.

- **Syntax:** TRUNCATE TABLE table_name

- **Example:**

```
TRUNCATE TABLE studios
```

- This deletes all the rows of the table studios

Modifying Records

Drop Statement

- **Drop Statement:**
  - used to remove elements from a database, such as tables, indexes or even users and databases. Drop command is used with a variety of keywords based on the need.

- **Drop Table Syntax:** DROP TABLE table_name
- **Drop Table Example:** DROP TABLE studios

- **Drop Index Syntax:** DROP INDEX table_name
- **Drop Index Example:** DROP INDEX movie_index

Modifying Records

Alter Statement

- **Alter Statement:**
  - used to make changes to the schema of the table. Columns can be added and the data type of the columns changed as long as the data in those columns conforms to the data type specified.

- **Syntax:**

ALTER TABLE table_name
ADD (column datatype [Default Expression])
[REFERENCES table_name (column_name)'
[CHECK condition]

- **Example:**

ALTER TABLE studios
ADD (revenue Number DEFAULT 0)

Modifying Records

Alter Statement

Add table level constraints:

- **Syntax:**

ALTER TABLE table_name
ADD ([CONSTRAINT constraint_name CHECK comparison]
[columns REFERENCES table_name (columns)]

- **Example:**

ALTER TABLE studios
ADD (CONSTRAINT check_state CHECK (studio_state in ('TX', 'CA', 'WA'))

Modify Columns:

- **Syntax:**

ALTER TABLE table_name
MODIFY column [data type]
[Default Expression]
[REFERENCES table_name (column_name)'
[CHECK condition]

- **Example:**

ALTER TABLE People
MODIFY person_union varchar(10)

- **Notes1:** Columns can not be removed from the table using alter. If you want to remove columns you have to drop the table and then recreate it without the column that you want to discard

Modifying Records

Alter Statement

- **Alter Statement:**
  - used to make changes to the schema of the table. Columns can be added and the data type of the columns changed as long as the data in those columns conforms to the data type specified.

- **Syntax:**

ALTER TABLE table_name
ADD (column datatype [Default Expression])
[REFERENCES table_name (column_name)'
[CHECK condition]

- **Example:**

ALTER TABLE studios
ADD (revenue Number DEFAULT 0)


THANK
YOU