Module 12 – Create Tables and Constraints

Module Objectives

Create a database table in each SQL dialect

Describe the different data types in each SQL dialect

ALTER, RENAME, and DROP tables in each SQL dialect

Describe the five constraints and their functionality

Create and maintain constraints in each SQL dialect

Create a database table in each SQL dialect

Tables can be created at any time, even while users are using the database.

You do not need to specify the size of a table when it is created, because the size is ultimately defined by the amount of space allotted to the database as a whole.

Table structures can be modified while the database is online and functioning.

Table names and column must begin with a letter and can be 1 to 30 characters in length.

Names must not duplicate the name of another database object owned by the same database user.

Names must not be a reserved word.

Always use descriptive names for tables and other database objects.

To create a new table, use the following syntax:

```
CREATE TABLE table_name
(column_name    data_type,
column_name     data_type);
```

ORACLE Example:

```
CREATE TABLE schema.dept
(Deptno       NUMBER(2),
Dname         VARCHAR2(15),
Location      VARCHAR2(15));
```

MySQL Example:

```
CREATE TABLE schema.dept
(Deptno       INT,
Dname         VARCHAR(15),
Location      VARCHAR(15));
```

MS SQL Example:

```
CREATE TABLE schema.dept
(Deptno       INT,
Dname         VARCHAR(15),
Location      VARCHAR(15));
```

Describe the different data types in each SQL dialect

NUMBERS		
	INT	Numeric data from -2,147,483,648 to 2,147,483,647
	FLOAT	Numeric data supporting up to 23 positions in size (4-bytes)
	DOUBLE	Numeric data supporting up to 53 positions in size (8-bytes)
DATE		
	DATE	A date (YYYY-MM-DD)
	TIME	A time (HH:MI:SS)
	DATETIME	A date and time combination (YYYY-MM-DD HH:MI:SS)
CHARACTERS		
	CHAR	Used for fixed size text up to 255 characters
	VARCHAR	Used for text up to 255 characters
	TEXT	Used for text up to 65,535 characters
	LONGTEXT	Used for text up to 4,294,967,295 characters
	LONGBLOB	Used to hold up to 4,294,967,295 of data

ORACLE®

NUMBERS		
	NUMBER	Numeric data that can be represented to full 38-digit precision
DATE		
	DATE	A date (DD-MON-YY)
CHARACTERS		
	CHAR	Used for fixed size text up to 2000 characters
	VARCHAR2	Used for text up to 4000 characters
	BLOB	Used to hold up to 4,294,967,295 of binary data
	CLOB	Used to hold up to 4,294,967,295 of character data

Microsoft® SQL Server®

NUMBERS		
	INT	Numeric data from -2,147,483,648 to 2,147,483,647
	DECIMAL	Numeric data that can be represented to full 38-digit precision
	NUMERIC	Numeric data that can be represented to full 38-digit precision
	MONEY	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807
DATE		
	DATE	A date only (YYYY-MM-DD)
	TIME	A time only (HH:MM:SS)
	DATETIME	A date and time combination (YYYY-MM-DD HH:MM:SS)
CHARACTERS		
	CHAR	Used for fixed size text up to 8000 characters
	VARCHAR	Used for text up to 1,073,741,824 characters
	TEXT	Used to hold up to 2,147,483,648 of character data
	VARBINARY	Used to hold up to 2,147,483,648 of binary data

ALTER, RENAME, and DROP tables in each SQL dialect

After you create a table, you may need to change the table structure for any number of reasons. This task can be accomplished by using the ALTER TABLE statement.

ADDING A COLUMN

To alter a table and add a new column to that table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name       data_type;
```

ORACLE Example:

```
ALTER TABLE schema.dept
ADD TerritoryID VARCHAR2(20);
```

MySQL Example:

```
ALTER TABLE schema.dept
ADD TerritoryID VARCHAR(20);
```

MS SQL Example:

```
ALTER TABLE schema.dept
ADD TerritoryID VARCHAR(20);
```

MODIFING A COLUMN

To alter a table and modify an existing column in a table, use the following syntax:

```
ALTER TABLE table_name
MODIFY | ALTER COLUMN  column_name        data_type;
```

ORACLE Example:

```
ALTER TABLE schema.dept
MODIFY TerritoryID VARCHAR2(40);
```

MySQL Example:

```
ALTER TABLE schema.dept
MODIFY TerritoryID VARCHAR(40);
```

MS SQL Example:

```
ALTER TABLE schema.dept
ALTER COLUMN TerritoryID VARCHAR(40);
```

RENAMING A TABLE

To rename an existing table with a new table name, use the following syntax:

ORACLE Example:

```
RENAME dept to department;
```

MySQL Example:

```
RENAME TABLE dept to department;
```

MS SQL Example:

EXEC SP_RENAME '*schema*.dept', 'department';

DROPPING A TABLE

To drop an existing table, use the following syntax:

DROP TABLE *table_name*;

ORACLE Example:

```
DROP TABLE department;
```

MySQL Example:

DROP TABLE *schema*.department;

MS SQL Example:

DROP TABLE *schema*.department;

Describe the five constraints and their functionality

Constraints enforce rules at the table level whenever a row is inserted, updated, or deleted from that table. The constraint must be satisfied for the operation to succeed.
Constraints prevent the deletion of a table if there any dependencies.

Constraint	**Description**
NOT NULL	Specifies that this column may not contain a null value.
UNIQUE	Specifies a column values must be unique for all rows in the table.
PRIMARY KEY	Uniquely identifies each row of the table.
FOREIGN KEY	Establishes and enforces a relationship between the column and a column of the referenced table.
CHECK	Specifies a condition that must be true.

Constraints can be created when the table is created or added to an existing table.
When creating constraints during table creation, you can create the constraints at the column level or at the table level.

Column Constraint Level

```
CREATE TABLE table_name
(column_name          data_type       CONSTRAINT constraint_name constraint_type,
column_name           data_type);
```

Table Constraint Level

```
CREATE TABLE table_name
(column_name          data_type,
column_name           data_type,
CONSTRAINT constraint_name constraint_type (column_name));
```

ORACLE Example:

Column Constraint Level

```
CREATE TABLE dept
(deptno         NUMBER(2) CONSTRAINT dept_deptno_pk PRIMARY KEY,
dname           VARCHAR2(15),
location        VARCHAR2(15));
```

Table Constraint Level CREATE

```
TABLE dept (deptno
                NUMBER(2),
dname           VARCHAR2(15),
location        VARCHAR2(15),
CONSTRAINT dept_deptno_pk PRIMARY KEY (deptno));
```

MySQL Example:

Column Constraint Level

```
CREATE TABLE dept
(deptno     INT PRIMARY KEY,
dname       VARCHAR(15),
location    VARCHAR(15));
```

Table Constraint Level

```
CREATE TABLE dept
(deptno     INT,
dname       VARCHAR(15),
location    VARCHAR(15),
CONSTRAINT dept_deptno_pk PRIMARY KEY (deptno));
```

MS SQL Example:

Column Constraint Level

```
CREATE TABLE dept
(deptno     INT CONSTRAINT dept_deptno_pk PRIMARY KEY,
dname       VARCHAR(15),
location    VARCHAR(15));
```

Table Constraint Level

```
CREATE TABLE dept
(deptno     INT,
dname       VARCHAR(15),
location    VARCHAR(15),
CONSTRAINT dept_deptno_pk PRIMARY KEY (deptno));
```

Create a FOREIGN KEY Constraint

The syntax to create a foreign key constraint is as follows:

Column Level
CREATE TABLE *table_name*
(*column_name data_type* CONSTRAINT *constraint_name* REFERENCES
    table_name(*column_name*));

Table Level
CREATE TABLE *table_name*
(*column_name data_type*,
CONSTRAINT *constraint_name* FOREIGN KEY (*column_name*)
    REFERENCES *table_name*(*column_name*));

Create a CHECK Constraint

The syntax to create a check constraint is as follows:

Column Level
CREATE TABLE *table_name*
(*column_name data_type* CONSTRAINT *constraint_name*
    CHECK (*test_condition*));

Table Level
CREATE TABLE *table_name*
(*column_name data_type*,
CONSTRAINT *constraint_name* CHECK (*test_condition*));