Data Definition Language
Database 2 - Lecture 2

1. Introduction

DDL commands are used to define a database, including creating, altering, and dropping tables and establishing constraints.

2. Creating a Database

The first step in databases is creating the database. There are two main ways to create a database. First, many RDBMSs come with a nice, user-friendly front-end interface, which makes the task of creating a new database very easy. The second, RDBMSs allow you to use SQL to create a database.

CREATE DATABASE *DatabaseName**;***

To delete database:

DROP DATABASE *DatabaseName**;***

Note: In the case of MS Access, the only way to create a new database is by using Access GUI.

3. Data types in SQL

When we create a table we must specify a data type for each of its columns. These data types define the domain of values that each column can take.

ANSI SQL	MS Access	SQL Server 2000	IBM DB2	MySQL	Oracle 10
Character	char	char	char	char	char
Character varying	varchar	varchar	varchar	varchar	varchar
National character	char	nchar	graphic	char	nchar
National character varying	varchar	nvarchar	vargraphic	varchar	nvarchar
Integer	number (long integer)	int	int	int	int
Smallint	number (integer)	smallint	smallint	smallint	smallint
Real	number (double)	real	real	real	real
Decimal	number (decimal)	decimal	decimal	decimal	decimal
Date	date	datetime	date	date	date
Time	time	datetime	time	time	date

STRING

In string we have CHAR, VARCHAR and TEXT datatypes. Character datatype store data which are words and free-form text, in the database character set.

CHAR Datatype

The CHAR datatype specifies a fixed-length character string. The syntax of CHAR datatype declaration is:

CHAR (n) – Fixed length character data, "n" characters long. Here "n" specifies the character length.

VARCHAR Datatype

The VARCHAR datatype specifies a variable-length character string. The syntax of VARCHAR datatype declaration is:

VARCHAR (n) – Variable length character of "n" length.

NUMBER Datatype

The NUMBER datatype stores zero, positive, and negative fixed and floating point numbers.

DATE Datatype

The DATE datatype is used to store the date and time information.

COUNTER or AUTOINCREMENT Datatype

Used as a integer counter.

4. Creating, Altering, and Deleting Tables

4.1 Creating Tables

The SQL command for creating an empty table has the following form:

```
create table <table>    (
<column 1> <data type> [NOT NULL] [<column constraint>],
. . . . . .
<column n> <data type> [NOT NULL] [<column constraint>],
[<table constraint(s)>]   );
```

- [<column constraint>] = [PRIMARY KEY] | [UNIQUE] | [CHECK()]| … etc
- [<table constraint(s)>]= [CONSTRAINT [constraint_name]] [<column constraint>]

Ex:

```
CREATE TABLE Train_Times
    (
    start_location varchar(75),
    destination varchar(75),
    departs time,
    arrives time
    );
```

ALTER Tables

To add a new column, use the basic syntax shown below:

```
ALTER TABLE name_of_table
ADD name_of_field data_type
```

To delete an existing column, the syntax is identical except you now tell the database system that you want to delete a column and supply that column's name:

```
ALTER TABLE name_of_table
DROP COLUMN name_of_field
```

What if you decide later on, after you create a table, to modify data type or constraint, we can use:

```
ALTER TABLE name_of_table
ALTER COLUMN name_of_field  new_ data_type
```

Note: In Oracle and MySQL, you must use MODIFY instead of ALTER COLUMN.

4.2 Deleting an Existing Table

The basic syntax is:

```
DROP TABLE name_of_table
```

5. Ensuring Data Validity with Constraints

When creating database tables and fields, you can specify constraints that limit what data can go in a field.

Using the DBMS to protect data integrity is not the only way, but it's probably the best. An alternative is having an external program ensure that the data is valid, as most databases are not accessed directly by the user but via some third-party front-end interface.

This section covers the following constraints:

- NOT NULL
- UNIQUE
- CHECK
- PRIMARY KEY
- FOREIGN KEY

5.1 NOT NULL Constraint

Ex:

```
CREATE TABLE MyTable
(
        Column1 int NOT NULL,
        Column2 varchar(20),
```

```
    Column3 varchar(12) NOT NULL
)
```

5.2 UNIQUE Constraint

The UNIQUE constraint prevents two records from having identical values in a particular column. For example, you might want to prevent two or more people from having identical email addresses.

You can apply the UNIQUE constraint in two ways: add it when creating the table, or add it after creating the table.

We can add the UNIQUE constraint to the table definition immediately after the column's type definition:

Ex:

```
CREATE TABLE MyUniqueTable
(
    Column1 int,
    Column2 varchar(20) UNIQUE,
    Column3 varchar(12) UNIQUE
);
```

Setting the UNIQUE constraint by simply adding UNIQUE after a column is nice and easy, but there's another way that has two advantages. In this alternative, you add the constraint at the end of the column listing in the table definition, which allows you to specify that two or more columns must in combination be unique. The other advantage is that you can give the constraint a name and you can delete the constraint using SQL.

The following code creates a new table, AnotherTable, and adds a unique constraint called MyUniqueConstraint.

```
CREATE TABLE AnotherTable
(
    Column1 int,
    Column2 varchar(20),
    Column3 varchar(12),
    CONSTRAINT MyUniqueConstraint UNIQUE(Column2, Column3)
);
```

You can add more than one constraint to a table, so long as each constraint has a different name and is separated by a comma:

```
CREATE TABLE AnotherTable
(
    Column1 int,
    Column2 varchar(20),
    Column3 varchar(12),
```

```
    CONSTRAINT MyUniqueConstraint UNIQUE(Column2, Column3),
    CONSTRAINT AnotherConstraint UNIQUE(Column1, Column3)
);
```

So far we know how to add a UNIQUE constraint at the time of a table's creation. However, using the ALTER TABLE statement, we can add and remove a UNIQUE constraint after the table's creation.

Ex:

```
ALTER TABLE YetAnotherTable
ADD CONSTRAINT MyUniqueConstraint UNIQUE(Column2, Column3);
```

To delete constraint we use:

```
ALTER TABLE YetAnotherTable
DROP CONSTRAINT MyUniqueConstraint;
```

Note: in MySQL we use the following:

```
ALTER TABLE YetAnotherTable
DROP INDEX MyUniqueConstraint;
```

5.3 CHECK Constraint

The CHECK constraint enables a condition to check the value being entered into a record. If the condition evaluates to false, the record violates the constraint and isn't entered into the table. For example, allowing a column storing an age to contain a negative value doesn't make sense.

```
CREATE TABLE NamesAges
(
    Name varchar(50),
    Age int CHECK (Age >= 0)
);
```

Note: by using the previous example, we can insert NULL in Age filed because (NULL >= 0) evaluates to unknown.

A problem with adding a CHECK clause to an individual column definition is that the condition can check only that column.

If you want your CHECK condition clause (the clause defined following the CHECK statement) to include multiple columns from the table, you need to define it at the end of the column definitions.

Ex:

```
CREATE TABLE Employee
(
    Name varchar(50),
    a decimal(12,2),
    b decimal(12,2),
```

```
    CONSTRAINT MyConstraint CHECK (a > b)
);
```

5.4 Primary Key and PRIMARY KEY Constraint

Of all the constraints, PRIMARY KEY is the most important and most commonly used. In fact, every table should have a primary key because first normal form requires it.

To specify a single column, simply insert PRIMARY KEY after its definition, like this:

```
CREATE TABLE Books
(
    BookId int PRIMARY KEY,
    Year int,
    BookName varchar(50)
);
```

In the preceding code, BookId is the primary key. Alternatively, more than one column can act as the primary key, in which case you need to add the constraint at the end of the table, after the column definitions.

```
CREATE TABLE T1
(
    Pk1 int,
    Pk2 int,
    name varchar(50),
    CONSTRAINT pks PRIMARY KEY (Pk1, Pk2)
);
```

Note: Always you can use ADD & DROP CONSTRAINT in ALTER TABLE statement but in MySQL to delete primary key use:

```
ALTER TABLE MoreHolidayBookings
DROP PRIMARY KEY;
```

5.5 Foreign Key

Foreign keys are columns that refer to primary keys in another table. Primary and foreign keys create relations between data in different tables.

The basic syntax to create a foreign key is shown in the following code:

```
ALTER TABLE name_of_table_to_add_foreign_key
ADD CONSTRAINT name_of_foreign_key_constraint
FOREIGN KEY (name_of_column_that_is_foreign_key_column)
REFERENCES name_of_table_that_is_referenced(name_of_column_being_referenced)
```

Ex:

```
ALTER TABLE Attendance
```

```
ADD CONSTRAINT locationid_fk
FOREIGN KEY (LocationId)
REFERENCES Location(LocationId);
```

Note: You can also use FOREIGN KEY constraint in CREATE TABLE statement.

6. Speeding Up Results with Indexes

Indexes are special lookup tables that the database search engine can use to speed up data retrieval. They also organize the way in which the database stores data. A good real-world example is a book, much like this one, that has an index at the back that helps you find where the important information is.

Creating an index involves the CREATE INDEX statement, which allows you to name the index, to specify the table and which column or columns to index, and to indicate whether the index is in ascending or descending order. Indexes can also be unique, similar to the UNIQUE constraint, in that the index prevents duplicate entries in the column or combination of columns on which there's an index. The basic format of the statement is as follows:

```
CREATE INDEX <index_name>
ON <table_name> (<column_names>)
```

<column_names>= column_name1 [DESC], column_name2 [DESC],….., column_name n [DESC]

Ex:

```
CREATE INDEX member_name_index
ON MemberDetails (FirstName, LastName);
```

Note: by default, the table will be sorted in an ascending order according to indexes filed(s).

The SQL to drop the index just created in MS SQL Server is as follows:

```
DROP INDEX MemberDetails.member_name_indx;
```

In IBM DB2 and Oracle, the DROP INDEX statement simply requires the index name without the table name prefixed:

```
DROP INDEX member_name_indx;
```

In MySQL, the code to drop the index is as follows:

```
ALTER TABLE MemberDetails
DROP INDEX member_name_indx;
```

MS Access has yet another way of dropping the index:

```
DROP INDEX member_name_indx ON MemberDetails;
```