Birla Institute of Technology and Science, Pilani

CS F212 Database Systems

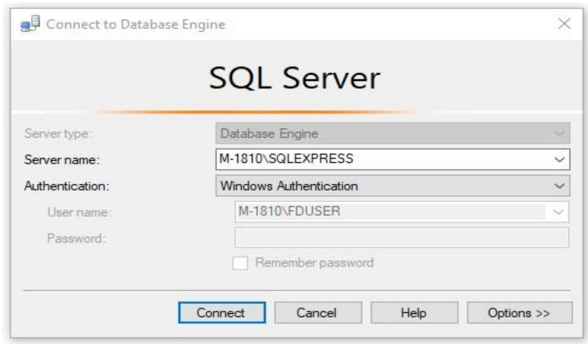
Lab No #1 - SQL Data Description Language (DDL)

Lab sheets are based on MS SQL Server 2016. Each DBMS does things differently, and no major DBMS follows the specification exactly.

SQL is divided into three major parts. Data description language (DDL) is used to define the structure of the data. Data manipulation language (DML) is used to store and retrieve data from the database. Data control language (DCL) is used to restrict access to data by certain users.

Setting up MS SQL:

"Start" → "Microsoft SQL Server Management Studio".

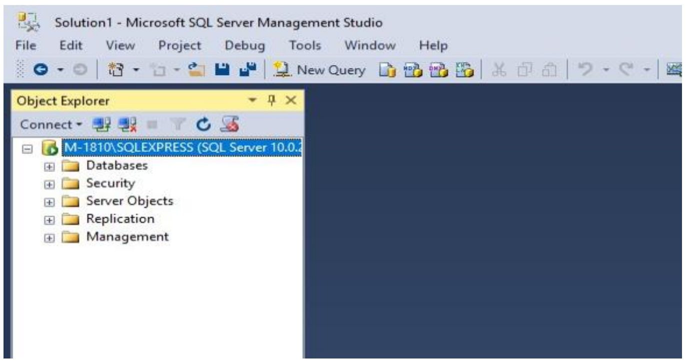


Connect to Database Engine

SQL Server

Server type: Database Engine
Server name: M-1810\SQLEXPRESS
Authentication: Windows Authentication
User name: M-1810\FDUSER
Password:
Remember password

Click on New Query and start using the worksheet on the right-hand side.



Solution1 - Microsoft SQL Server Management Studio

Right Click on the query tab and save it in a file. For executing an SQL query saved in a file open the saved file using File menu and execute.

Before we start

❖ To delete (drop) all the tables that currently exist in the database, use the following query:

```
exec sp_MSforeachtable "declare @name nvarchar(max); set @name =
parsename('?', 1);
exec sp_MSdropconstraints @name";
exec sp_MSforeachtable "drop table ?";
```

Note:

❖ SQL is NOT case sensitive.

❖ "--" is used for single line comments

❖ "/*" and "*/" are used in pair for multi-line comments, just like C.

Basic data types:

Although there are lots of data types available, we shall be covering only the following:

Datatype	Format	Explanation
Number	Numeric(m[,n])	m is the total number of digits in the number and n signifies number of decimal digits. A square bracket means the contents inside it are optional (in which case the number is an integer).
Character	Char(n)	Strings with a fixed length n.
Varchar	Varchar(n)	Variable length strings with a maximum length n.
Date	Date	Storing date values

Creating a table

❖ To create a table, we use the "Create table …" DDL.

```
 CREATE TABLE <TABLENAME> (
<COLUMNNAME><TYPE> [<DEFAULTVALUE>] [<COLUMNCONSTRAINTS>],  :::
<COLUMNNAME><TYPE> [<DEFAULTVALUE>] [<COLUMNCONSTRAINTS>],
<TABLECONSTRAINT>,  :::   <TABLECONSTRAINT>
);
```

❖ This creates a table named <tablename>. The columns of the table are specified in a comma delimited list of name/data type pairs. Optionally, you may specify constraints and default values.

❖ Execute the following DDL in the query editor.

```
CREATE TABLE STUDENTS (
IDNO CHAR(11),
NAME VARCHAR(30),
DOB DATE,
CGPA NUMERIC(4,2),
AGE NUMERIC(2)
);
```

Inserting records in a table

- To insert records in a table, we shall be using the "Insert into …" DML. There are two forms.

```
    INSERT INTO <TABLENAME>
    [(<ATTRIBUTE>,:::, <ATTRIBUTE>)]
    VALUES (<EXPRESSION>,:::, <EXPRESSION>);
```

- First form: Specify values for all the columns in the same order as during table creation. For columns that take null values, NULL can also be specified.

```
    INSERT INTO STUDENTS
    VALUES (‘2017A7PS1196P’,‘K RAMESH’,‘21-JUN-75’, 8.82, 27);
```

- The above Insert statement will create a new record in the students table and insert these values in the corresponding column. Note that all the char, varchar and date literals are specified in single quotes. The default format of date value is "yyyy-mm-dd". Various other formats are also supported.
- Second Form: We directly reference column names in the SQL statement. This gives us a lot of flexibility: we can omit any column and don't need to maintain any order.

```
    INSERT INTO STUDENTS (NAME, IDNO, AGE)
    VALUES (‘R SURESH’, ’2016A7PS003P’, 25);
```

- The above Insert statement will create a new record and insert the specified values into the corresponding columns. It will insert NULL values in columns that are not specified here. Hence only those columns which can take NULL values should be omitted.

- Note: Specifying a column as type char doesn't guarantee that it will accept exactly that many number of characters. If you specify lesser number of characters, it will simply pad blank spaces.

Constraints

- Your DBMS can do much more than just store and access data. It can also enforce rules (called constraints) on what data is allowed in the database. Such constraints are important to maintain data integrity. For example, you may want to ensure that each CGPA is not less than 2.0.
- When you specify the data type of a column, you constrain the possible values that column may hold. This is called a domain constraint. For example, a column of type INTEGER may only hold whole numbers within a certain range.

- SQL allows the specification of many more constraint types. SQL enforces constraints by prohibiting any entries in the database that violate any constraint. Any insert, update, or delete that would result in a constraint violation is rejected without changing the database.
- There are two forms of constraint specification:
  - Column Level: Apply to individual columns (are specified along with the column definition)
  - Table Level: Apply to one or more columns (are specified at the end)
- Constraints can be added to the table at the time of creation or after the creation of the table using '`alter table`' command.

NOT NULL

- By default, most DBMSs allow NULL as a value for any column of any data type. Database can prohibit NULL values for particular columns by using the NOT NULL column constraint.

- Many DBMSs also include a NULL column constraint, which specifies that NULL values are allowed. However, this is the default behaviour, so this constraint usually is unnecessary. Note that the NULL column constraint is not part of the SQL specification.

```
CREATE TABLE STAFF (
SID NUMERIC (10) NOT NULL,
NAME VARCHAR (20),
DEPT VARCHAR (15)
);
```

- Executing the following will give error: "Cannot insert the value NULL into column …"

```
INSERT INTO STAFF (NAME, DEPT)
VALUES ('KRISHNA', 'PSD');
```

UNIQUE

- The UNIQUE constraint forces distinct column values. To avoid duplicate course numbers, just specify the UNIQUE column constraint on the `courseno` column as follows:

```
CREATE TABLE COURSE (
COMPCODE NUMERIC (4) UNIQUE,
COURSENO VARCHAR (9) NOT NULL UNIQUE,
COURSE_NAME VARCHAR (20),
UNITS NUMERIC (2) NOT NULL
);
```

- Note that UNIQUE only applies to non-NULL values. A UNIQUE column may have many rows containing a NULL value. Of course, we can exclude all NULL values for the column using the NOT NULL constraint with the UNIQUE constraint.
- UNIQUE also has a table constraint form that applies to the entire table instead of just a single column. Table constraints are specified as another item in the comma-delimited list of table elements. Such table constraints apply to groups of one or more columns.
- Consider the following CREATE TABLE statement where the combination of compcode, courseno is always unique:

```
DROP TABLE COURSE; --TO DELETE THE TABLE

CREATE TABLE COURSE (
COMPCODE NUMERIC (4),
COURSENO VARCHAR (9) NOT NULL,
COURSE_NAME VARCHAR (20),
UNITS NUMERIC (2) NOT NULL,
UNIQUE (COMPCODE, COURSENO)   -- TABLE LEVEL CONSTRAINT
);
```

- A table level unique constraint can also be enforced for a single column.

PRIMARY KEY

- We can declare a primary key using the PRIMARY KEY constraint. The primary key of a table is a column or set of columns that uniquely identifies a row in the table. It is similar to unique but with implicit NOT NULL constraint.
- For example, idno is the primary key from the students table. Here we show PRIMARY KEY used as a column constraint.

```
CREATE TABLE EMPLOYEE (
EMPID NUMERIC (4) PRIMARY KEY,
NAME VARCHAR (30) NOT NULL
);
```

- Creating primary key as a table constraint is shown below.

```
CREATE TABLE REGISTERED (
COURSENO VARCHAR (9),
IDNO CHAR (13),
GRADE VARCHAR (10),
PRIMARY KEY (COURSENO, IDNO)
);
```

- You can name your constraints by using an optional prefix.

```
[CONSTRAINT <NAME>] <CONSTRAINT>
```

```
CREATE TABLE REGISTERED (
COURSENO VARCHAR(9),
IDNO CHAR(13),
GRADE VARCHAR(10),
CONSTRAINT PK_REGISTERED PRIMARY KEY(COURSENO, IDNO));
```

- ❖ Naming can also be applied to unique constraints. Naming constraint helps in altering or dropping that constraint at a later time.

FOREIGN KEY

- ❖ A foreign key restricts the values of a column (or a set of columns) to the values appearing in another column (or set of columns) or to NULL.
- ❖ In table registered (child table), courseno is a foreign key that refers to courseno in table course (parent table). All values of courseno in the registered table should either refer a courseno in course or be NULL. Any other courseno in the registered table would create problems because you couldn't look up information about the courseno which isn't offered.
- ❖ In SQL, we specify a foreign key with the REFERENCES column constraint.

```
REFERENCES <REFERENCED TABLE> [(<REFERENCED COLUMN>)]
```

- ❖ A column with a REFERENCES constraint may only have a value of either NULL or a value found in column <referenced column> of table <referenced table>. If the <referenced column> is omitted, the primary key of table <referenced table> is used.
- ❖ Before we go ahead, let us re-create course, students tables with proper constraints.

```
DROP TABLE STUDENTS;
CREATE TABLE STUDENTS(
IDNO CHAR(13),
NAME VARCHAR(30),
DOB DATE,
CGPA NUMERIC(4,2),
AGE NUMERIC(2),
CONSTRAINT PK_STUDENTS PRIMARY KEY(IDNO)
);

DROP TABLE COURSE;
CREATE TABLE COURSE (
COMPCODE NUMERIC(4),
COURSENO VARCHAR(9) NOT NULL,
COURSE_NAME VARCHAR(20),
UNITS NUMERIC(2) NOT NULL,
CONSTRAINT UN_COURSE UNIQUE (COMPCODE, COURSENO),   -- TABLE LEVEL
CONSTRAINT
CONSTRAINT PK_COURSE PRIMARY KEY (COURSENO)
);

CREATE TABLE REGISTERED1 (
COURSENO VARCHAR (9) REFERENCES COURSE, --COLUMN LEVEL FOREIGN KEY
IDNO CHAR(13) REFERENCES STUDENTS,
GRADE VARCHAR(10),
PRIMARY KEY(COURSENO, IDNO)
);
```

- The same can be declared as table level constraint with proper naming.

```
CREATE TABLE REGISTERED2(
COURSENO VARCHAR(9),
IDNO CHAR(13),   GRADE VARCHAR(10),
CONSTRAINT PK_REGISTERED2 PRIMARY KEY(COURSENO, IDNO),
CONSTRAINT FK_CNO FOREIGN KEY(COURSENO) REFERENCES COURSE,
CONSTRAINT FK_IDNO FOREIGN KEY (IDNO) REFERENCES STUDENTS
);
```

- To create a foreign key reference, SQL requires that the referenced table/column already exist.
- To update or delete a referenced value in the parent table, we must make sure that we first handle all foreign keys referencing that value in the child table. For example, to update or delete 2017A7PS1195P from the students table, we must first update or delete all registered idno.
- SQL allows us to specify the default actions for maintaining foreign key constraints for UPDATE and DELETE on the parent table by adding a referential action clause to the end of a column or table foreign key constraint:

```
ON UPDATE <ACTION>
ON DELETE <ACTION>
```

- Any UPDATE or DELETE on the parent table triggers the specified <action> on the referencing rows in the child table.

Action	**Definition**
SET NULL	Sets any referencing foreign key values to NULL.
SET DEFAULT	Sets any referencing foreign key values to the default value (which may be NULL).
CASCADE	On delete, this deletes any rows with referencing foreign key values. On update, this updates any row with referencing foreign key values to the new value of the referenced column.
NO ACTION	Rejects any update or delete that violates the foreign key constraint. This is the default action.

RESTRICT	Same as NO ACTION except in database systems which allow deferred checks. In that case, the additional restriction is that the action cannot be deferred.

- Execute the following:

```
DROP TABLE REGISTERED2;

CREATE TABLE REGISTERED2(
COURSENO VARCHAR(9),
IDNO CHAR(13),
GRADE VARCHAR(10),
CONSTRAINT PK_REGISTERED2  PRIMARY KEY(COURSENO, IDNO),
CONSTRAINT FK_CNO FOREIGN KEY (COURSENO) REFERENCES COURSE ON DELETE
CASCADE,
CONSTRAINT FK_IDNO FOREIGN KEY (IDNO) REFERENCES STUDENTS ON DELETE
CASCADE
);
```

List all the tables created using the following query.

```
SELECT * FROM INFORMATION_SCHEMA.TABLES;
```

List all the constraints using the following query

```
SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS;
```

Creating a table from another table

SQL allows you to create and populate a new table from existing tables with one statement.

Execute the following query.

```
SELECT * INTO STU_COPY FROM STUDENTS;

SELECT IDNO, NAME
INTO STU_COPY2
FROM STUDENT;
```

Dropping a table

```
DROP TABLE <TABLENAME> [CASCADE | RESTRICT];
SQL> DROP TABLE STUDENT;
```

Exercise 1:

❖ Create the following tables (Don’t specify any constraints):

Category_details (category_id numeric (2), category_name varchar (30))

Sub_category_details (sub_category_id numeric(2), category_id numeric(2), sub_category_name varchar(30))

Product_details (Product_id numeric (6), category_id numeric(2), sub_category_id numeric(2), product_name varchar(30))

Now perform the following operations:

1. Add a primary key constraint (without any constraint name) on column category_id of category_details table.
2. Add a primary key constraint with a constraint name on column sub_category_id of sub_category_details table.
3. Add a foreign key constraint with constraint name on column category_id of sub_category_details table referencing category_id of category_details table.
4. For product_details table add primary key constraint on product_id. Also add foreign key constraint on category_id and sub_category_id columns referencing category_details (category_id) and sub_category_details (sub_category_id). Give appropriate names for all constraints.
5. Insert four tuples in the table (with valid data).

Exercise 2 (refer supplementary material):

1. Add a new column (price numeric(2)) to product_details table
2. Modify the data type of price to numeric(6,2)
3. Drop the price column
4. Using the rules defined in the beginning, add a new column BRANDNAME varchar(20) NOT NULL
5. Rename Category_details table to Cat_dt

---------&----------