Birla Institute of Technology and Science, Pilani

CS F212 Database Systems

Lab No #1

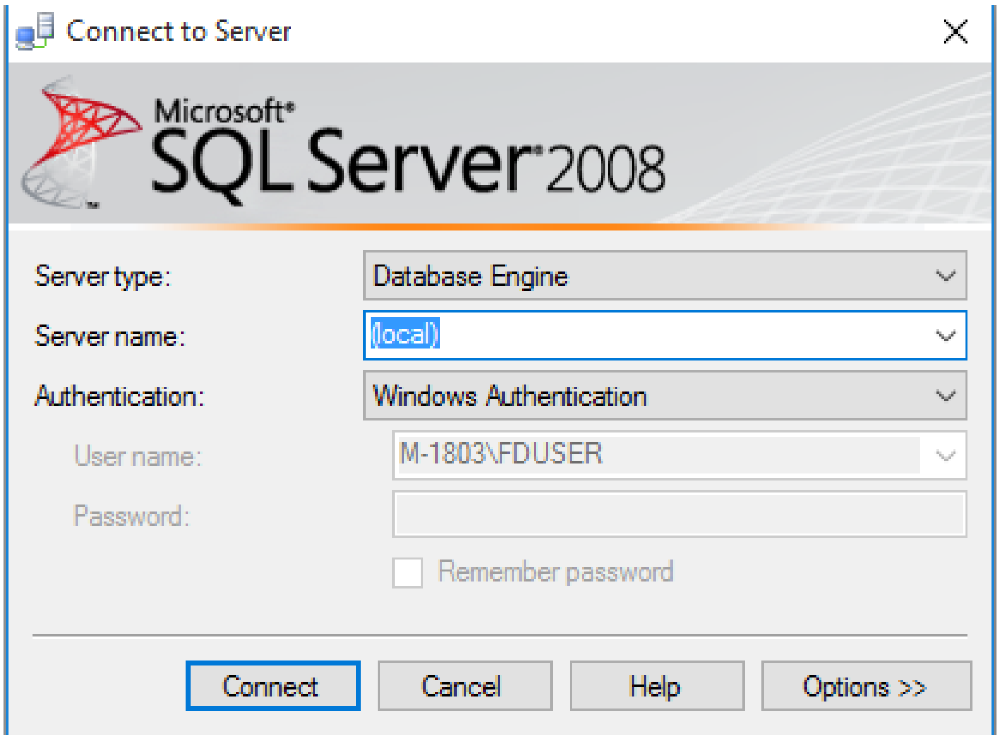
Lab sheets are based on MS SQL Server. Each DBMS does things differently, and no major DBMS follows the specification exactly.

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.

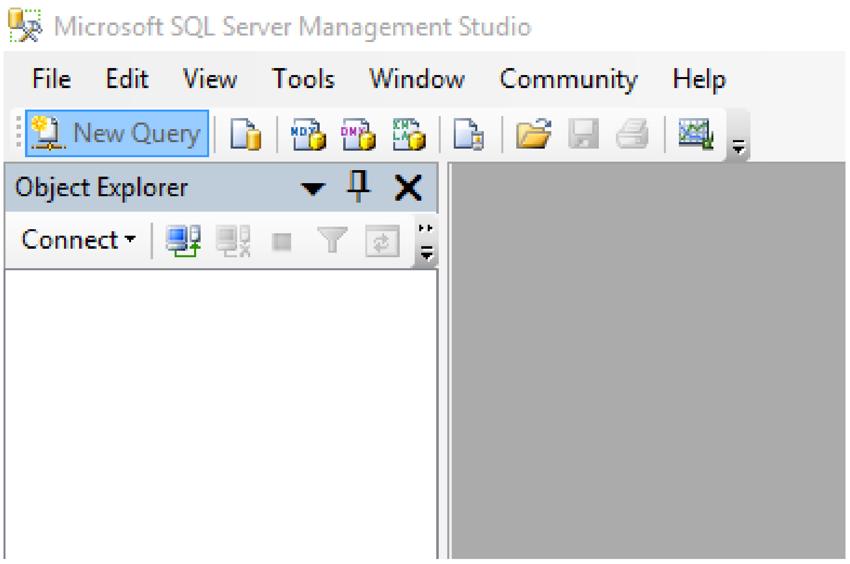
<u>SQL Data Description Language (DDL):</u>

How to log on the SQL Server ?

- “Start” → “Programs” → “Microsoft SQL Server 2008” → “SQL Server Management Studio”
- Click on New Query and start using the worksheet on the right hand side.



Server type: Database Engine
Server name: (local)
Authentication: Windows Authentication
User name: M-1803\FDUSER
Password:
Remember password



the last executed query in a file for future use

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.

Basic data types in Oracle server:

Although there are lots of data types available but for our purpose we shall be covering only the following:

Datatype	Format	Explanation
Number	Numeric(m[,n])	Here, 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 and in that case the number become an integer.
Character	Char(n)	This is for fixed length strings and n signifies the maximum number of characters.
Varchar	Varchar(n)	For variable length strings and n signifies the maximum number of characters.
Date	Date	For storing date values

Other data types shall be introduced as and when required.

Creating your first object (a table without any integrity constraints)

- 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 <table name>. 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)
);
```

- Now list the tables created using the above mentioned query.

Inserting records in the table

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

```
 INSERTINTO <TABLENAME>

[(<ATTRIBUTE>,:::, <ATTRIBUTE>)]

VALUES (<EXPRESSION>,:::, <EXPRESSION>);
```

- First form: Here we need to specify values for all the columns and in the same order as they were specified at the time of table creation. For a particular column that can take null values we can specify its value as NULL also.

```
INSERT INTO STUDENTS

VALUES ('1997B2A5563','K RAMESH','21-JUN75', 7.86, 27);
```

- The above Insert statement will create a new record in the students table and insert these values in the corresponding columns. Note that all the char, varchar and date literals are specified in single quotes and also the format of date value is “dd-mm-yy” which is the default format.
- Using the above SQL statement you can insert as many records as you want.
- Second Form: In the second form we can use column names in the SQL statement and this gives us a lot of flexibility. Firstly, we can omit any column and secondly, we don’t need to maintain any order.

```
INSERT INTO STUDENTS (NAME, IDNO, AGE)

VALUES (‘R SURESH’, ’1998A7PS003’, 25);
```

- The above Insert statement will create a new record and inserts the specified values into the corresponding columns. It will insert NULL values in those columns that are not specified here except… Hence only those columns which can take NULL values should only be omitted. Can you write the first stmt in the second form?
- Note that specifying a column as of type char doesn’t guarantee that it will accept exactly that much number of characters only. In those cases where you will specify lesser number 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 are allowed in the database. Such constraints are important because they help 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. Any attempt to insert an invalid value will be rejected by SQL. SQL allows the specification of many more constraint types. SQL enforces constraints by prohibiting any data 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 Constraints:- Apply to individual columns only (are specified along with the column definition only)
    - Table Level constraints:- 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. You may not be so keen on allowing NULL values for some columns. You can require the database to 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, because this is the default behavior, 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)
);
```

- Now if you try inserting,

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

- You will get error "Cannot insert the value NULL into column …"

UNIQUE

- The UNIQUE constraint forces distinct column values. Suppose you want 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:

```
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
);
```

- The combination of compcode, courseno is always unique. Note that table level unique constraint can also be for single column. Unique is nothing but the candidate key constraint but a unique column can take null values.

PRIMARY KEY

- It is similar to unique but with implicit NOT NULL constraint. The primary key of a table is a column or set of columns that uniquely identifies a row in the table. For example, idno is the primary key from the students table. We can declare a primary key using the PRIMARY KEY constraint. 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 (11),
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(11),
GRADE VARCHAR(10),
CONSTRAINT PK_REGISTERED PRIMARY KEY(COURSENO, IDNO)
);
```

- This naming can be applied to unique constraints also. 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). We want all of the values of courseno in the registered table either to reference a courseno from course or to be NULL. Any other courseno in the registered table would create problems because you couldn't look up information about the courseno which is not 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 creating a foreign keys in registered table, let us re-create course, students tables with proper constraints.

```
DROP TABLE STUDENTS;
CREATE TABLE STUDENTS(
IDNO CHAR(11),
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(11) 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(11),
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.

❖ Maintaining foreign key constraints can be painful. 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 2007A7PS001 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 with the additional restriction that the action cannot be deferred

❖ Try the following.

```
DROP TABLE REGISTERED2; CREATE
TABLE REGISTERED2( COURSENO
VARCHAR(9) ,  IDNO CHAR(11),
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

);
```

❖ Modify the above query with other actions ON DELETE SET NULL, ON DELETE NO ACTION, ON UPDATE NO ACTION.

CHECK

- We can specify a much more general type of constraint using the CHECK constraint. A CHECK constraint specifies a boolean value expression to be evaluated for each row before allowing any data change. Any INSERT, UPDATE, or DELETE that would cause the condition for any row to evaluate to false is rejected by the DBMS.

```
CHECK (<condition>)
```

- A CHECK constraint may be specified as either a column or table constraint. In the following example, we specify CHECK constraints on the students table:

```
CREATE TABLE STUDENT1(
IDNO CHAR(11) PRIMARY KEY,
NAME VARCHAR(20) NOT NULL,
CGPA NUMERIC(4,2) CHECK(CGPA >= 2 AND CGPA <= 10),  -- CGPA CONSTRAINT
ROOMNO NUMERIC(3) CHECK(ROOMNO >99),
HOSTEL_CODE VARCHAR(2) CHECK(HOSTEL_CODE IN ('VK','RP','MB'))
);
```

- Check constraints can also be named.
- Does a roomno with a NULL value violate the CHECK constraint? No. In this case, the CHECK condition evaluates to unknown. The CHECK constraint only rejects a change when the condition evaluates to false. In the SQL standard, a CHECK constraint condition may even include subqueries referencing other tables; however, many DBMSs do not implement this feature.

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;
```

More on Creating, Deleting, and Altering Tables: (DDL)

In the previous section, creation tables with defaults values, NOT NULL constraint, UNIQUE constraint, Primary key, foreign key and check constraints have been discussed.

- **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;
```

❖ **Alter Table**

You can modify the columns and constraints of a table using the following:

```
ALTER TABLE <TABLE NAME> ACTION>;
```

There are several types of actions:

- **Rename table:**

  ```
  EXEC SP_RENAME 'PHONE','PHONE1';
  ```

- **ADD [COLUMN] <column definition>**

  Adds a new column to <table name>. The <column definition> may contain any of the elements of a column definition in CREATE TABLE, such as types and constraints.

  Rules for adding a Column

  1. You may add a column at any time if NOT NULL isn't specified.
  2. You may add a NOT NULL column in three steps
     2.1. Add the column without NOT NULL specified.
     2.2. Fill every row in that column with data.
     2.3. Modify the column to be NOT NULL.

  ```
  SQL> ALTER TABLE STAFF ADD EMAIL VARCHAR (20);
  SQL> ALTER TABLE STUDENTS ADD HOSTEL VARCHAR (20);
  ```

- **MODIFY <column name> [SET DEFAULT <value expression> | DROP DEFAULT]**

  Change or drop the specified default value from <column name>. Your DBMS will likely let you alter much more than just the default values.

  Rules for modifying a Column

  1. You can increase a character column's width at any time.
  2. You can increase the number of digits in a NUMBER Column at any time.
  3. You can increase or decrease the number of decimal places in a NUMBER Column at any time.

In addition, if a column is NULL for every row of the table, you can make any of these changes.

1. You can change its datatype.
2. You can decrease a character Column's width.
3. You can decrease the number of digits in a NUMBER column.

```
SQL> ALTER TABLE STAFF ALTER COLUMN EMAIL VARCHAR (50);
```

- **DROP <column name> [CASCADE | RESTRICT]**

  Delete <column name> from <table name>. If there are any external dependencies on this column, the drop will be rejected. If CASCADE is specified, the DBMS will attempt to eliminate all external dependencies before deleting the column. If RESTRICT is specified, the DBMS will not allow the DROP if any external dependencies would be violated. This is the default behavior.

```
SQL> ALTER TABLE STAFF DROP COLUMN EMAIL;
```

- **ADD <table constraint>**

  Add a new constraint to <table name>. <table constraint> uses the same syntax as table constraints in CREATE TABLE.

```
SQL> ALTER TABLE STUDENTS ADD CONSTRAINT AGECHECK CHECK (AGE>15);
```

- **DROP CONSTRAINT <constraint name> [CASCADE | RESTRICT]**

  Delete <constraint name> from <table name>. If there are any external entities that depend on this constraint, the drop could invalidate the entity. If RESTRICT is specified and an entity could be invalidated, the DBMS will not drop the constraint. This is the default behavior. If CASCADE is specified, the DBMS will attempt to eliminate all potentially violated entities before deleting the constraint.

```
SQL> ALTER TABLE COURSE DROP CONSTRAINT UN_COURSE;
```

❖ **Generated Values**

SQL provides several mechanisms for auto generation of data.

```
CREATE TABLE PURCHASES (
ORDERNO INTEGER IDENTITY,
VENDORID CHAR (5), ORDERTIME DATETIME
);
```

- **Sequences (**SUPPORTED BY SQL SEVER 2012)**

Identity is one way of automatically generating values in SQL, but a more general approach is a sequence. A sequence is a database object that generates values. The values are numeric, but the sequence can be ascending or descending, can skip values, and can start at any valid value.

The 2003 standard defines a sequence as follows:

```
CREATE SEQUENCE <SEQUENCE NAME> [AS <DATA TYPE>] [START WITH
<SIGNED NUMERIC LITERAL>]
[INCREMENT BY <SIGNED NUMERIC LITERAL>]
[MAXVALUE <SIGNED NUMERIC LITERAL> | NO MAXVALUE]
[MINVALUE <SIGNED NUMERIC LITERAL> | NO MINVALUE]
[CYCLE | NO CYCLE]
```

Effectively, each time a new value is generated by a sequence, the increment value is added to the current value. If the increment value is negative, then the current value decreases and the sequence is descending. If the increment value is positive, then the current value increases and the sequence is ascending. It is illegal for the increment value to be 0.

The following statement creates the sequence customers_seq. This sequence could be used to provide customer ID numbers when rows are added to the customers table.

```
CREATE SEQUENCE CUSTOMERS_SEQ
START WITH     1000
INCREMENT BY   1
NOCACHE
NOCYCLE;
```

The first reference to customers_seq.nextval returns 1000. The second returns 1001. Each subsequent reference will return a value 1 greater than the previous reference.

```
SELECT CUSTOMERS_SEQ.CURRVAL FROM DUAL;
SELECT CUSTOMERS_SEQ.NEXTVAL FROM DUAL;
CREATE TABLE CUSTOMERS (
CUSTOMER_ID NUMERIC (9) PRIMARY KEY,
CUSTOMER_NAME CHAR (50)
);

INSERT INTO CUSTOMERS (CUSTOMER_ID, CUSTOMER_NAME)
VALUES (CUSTOMERS_SEQ.NEXTVAL,'JIVA');
```

- ❖ **DROPPING A TABLE**

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

Exercise:

- ❖ 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. Add a new column (price numeric(2)) to product_details table
6. Modify the data type of price to numeric(6,2)
7. Insert four tuples in the table. (With valid data)
8. Drop the price column
9. Using the rules defined in the beginning, add a new column BRANDNAME varchar(20) NOT NULL
10. Rename Category_details table to Cat_dt

- Case study:

The Employees Database stores information about a business, including employees, departments, and projects. Each employee works for one department and is assigned many projects. Each department has many employees, has many projects, and may be a subdepartment of one department (e.g., Accounting is a subdepartment of Administration). Each project is assigned to one department and is worked on by many employees.

Employees

Employeeid	Numeric(9)	Unique employee identifier (Primary Key)
Firstname	Varchar(10)	Employee first name
Lastname	Varchar(20)	Employee last name
Deptcode	Char(5)	Identifier of department the employee works for. Foreign key referencing departments.code
Salary	Numeric(9,2)	Employee salary

Departments

Code	Char(5)	Unique Department Identifier
Name	Varchar(30)	Department name
Managerid	Numeric(9)	Identifier of employee who manages the department. Foreign key referencing employees.employeeid
Subdeptof	Char(5)	Code of department that includes this department as one of its immediate subdepartments. Foreign key referencing departments.code

Projects

Projectid	Char(8)	Unique project identifier
Deptcode	Char(5)	Identifier of department managing this project. Foreign key referencing departments.code
Description	Varchar(200)	Project description
Startdate	Date	Project start date
Stopdate	Date	Project stop date. NULL value indicates that the project is ongoing
Revenue	Numeric(12,2)	Total project revenue

Workson

Employeeid	Numeric(9)	Identifier of employee working on a project. Foreign key referencing employees.employeeid
Projectid	Char(8)	Identifier of project that employee is working on. Foreign key referencing projects.projectid
Assignedtime	Numeric(3,2)	Percentage of time employee is assigned to project

- Let us create the database for the above schema

```
CREATE TABLE DEPARTMENTS(
CODE CHAR(5) PRIMARY KEY,
NAME VARCHAR(30),
MANAGERID NUMERIC (9),
SUBDEPTOF CHAR (5)
);
```

- Create employees table

```
ALTER TABLE DEPARTMENTS ADD CONSTRAINT FK_DEPARTMENTS_EMPLOYEES
FOREIGN KEY (MANAGERID) REFERENCES EMPLOYEES;
ALTER TABLE DEPARTMENTS ADD CONSTRAINT FK_DEPARTMENTS_SUBDEPT
FOREIGN KEY (SUBDEPTOF) REFERENCES DEPARTMENTS;
```

- Create project table with primary key
- Create workson table with primary key
- Create foreign keys between employees and department
- Create foreign keys between workson and employees
- Create foreign keys between projects and departments

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