Create table

NGUYEN Hong Phuong

Email: phuongnh@soict.hust.edu.vn

Site: https://users.soict.hust.edu.vn/phuongnh

Contents

CREATE/DROP DATABASE

- CREATE DATABASE *databasename*;
  - `CREATE DATABASE testDB;`

- DROP DATABASE *databasename*;
  - `DROP DATABASE testDB;`

BACKUP DATABASE for SQL Server

- The SQL BACKUP DATABASE Statement

  -
    ```
    BACKUP DATABASE databasename
    TO DISK = 'filepath';
    ```

  -
    ```
    BACKUP DATABASE testDB
    TO DISK = 'D:\backups\testDB.bak';
    ```
- The SQL BACKUP WITH DIFFERENTIAL Statement

  -
    ```
    BACKUP DATABASE databasename
    TO DISK = 'filepath'
    WITH DIFFERENTIAL;
    ```

  -
    ```
    BACKUP DATABASE testDB
    TO DISK = 'D:\backups\testDB.bak'
    WITH DIFFERENTIAL;
    ```

CREATE TABLE

- The SQL CREATE TABLE Statement

  -
    ```
    CREATE TABLE table_name (
        column1 datatype,
        column2 datatype,
        column3 datatype,
       ....
    );
    ```
  - Example
    ```
    CREATE TABLE Person(
          PersonID int,
          LastName varchar(255),
          FirstName varchar(255),
          Age int,
          Gender char(1),
          City varchar(255)
    );
    ```

CREATE TABLE (cont'd)

```
INSERT INTO Person
VALUES(1,'Hiddleston','Tom',23,'F','Florida'),
(2,'Watson','Angela',18,'F','Texas'),
(3,'Clooney','Pandora',34,'U','Arizona'),
(4,'Crane','Amory',52,'M','California'),
(5,'Clooney','Bush',67,'M','Arizona'),
(6,'Schwimmer','Geoffrey',19,'U','Hawaii')
```

CREATE TABLE (cont'd)

- Create Table Using Another Table
  - CREATE TABLE *new_table_name* AS
    ```
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
    ```
  - Example
    ```
    CREATE TABLE TestTable AS
    SELECT FirstName, LastName
    FROM Persons;
    ```
  - Syntax for SQL Server
    - `Select * into new_table  from  old_table`
    - Example
      ```
      SELECT * INTO Person2 FROM Person;
      ```

DROP TABLE

- The DROP TABLE statement is used to drop an existing table in a database.
  - DROP TABLE *table_name*;
  - Example

```
DROP TABLE Person2;
```

TRUNCATE TABLE

- The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.
  - `TRUNCATE TABLE` *`table_name`*`;`
  - Example

```
TRUNCATE TABLE Person
```

ALTER TABLE

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

ALTER TABLE (cont'd)

- ALTER TABLE - ADD Column
  - To add a column in a table, use the following syntax:
    - `ALTER TABLE table_name ADD column_name datatype;`
  - Example

```
ALTER TABLE Person ADD Email varchar(255);
```

- ALTER TABLE - DROP COLUMN
  - To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):
    - `ALTER TABLE table_name DROP COLUMN column_name;`
  - Example

```
ALTER TABLE Person DROP COLUMN Email;
```

SQL ALTER TABLE Statement (cont'd)

- ALTER TABLE - ALTER/MODIFY COLUMN
  - To change the data type of a column in a table, use the following syntax:
  - Syntax for SQL Server:

    -
      ```
      ALTER TABLE table_name
      ALTER COLUMN column_name datatype;
      ```
  - Example

```
ALTER TABLE Person ALTER COLUMN LastName varchar(50)
ALTER TABLE Person ALTER COLUMN FirstName varchar(50)
```

SQL Constraints

- SQL constraints are used to specify rules for the data in a table.
- Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.
- Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

SQL Constraints (cont'd)

- Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.
- Syntax with CREATE TABLE

  -
    ```
    CREATE TABLE table_name (
        column1 datatype constraint,
        column2 datatype constraint,
        column3 datatype constraint,
        ....
    );
    ```
- The following constraints are commonly used in SQL:

SQL Constraints (cont'd)

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

SQL NOT NULL Constraint

- By default, a column can hold NULL values.
- The NOT NULL constraint enforces a column to NOT accept NULL values.
- This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

SQL NOT NULL Constraint (cont'd)

- SQL NOT NULL on CREATE TABLE
  - The following SQL ensures that the "PersonID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Person" table is created:

```
DROP TABLE Person
CREATE TABLE Person(
       PersonID int NOT NULL,
       LastName varchar(50) NOT NULL,
       FirstName varchar(50) NOT NULL,
       Age int,
       Gender char(1),
       City varchar(255)
);
```

SQL NOT NULL Constraint (cont'd)

- SQL NOT NULL on ALTER TABLE

```sql
ALTER TABLE Person ALTER COLUMN Age int NOT NULL;
```

SQL UNIQUE Constraint

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

SQL UNIQUE Constraint (cont'd)

- SQL UNIQUE Constraint on CREATE TABLE
  - The following SQL creates a UNIQUE constraint on the "PersonID" column when the "Person" table is created:

```
DROP TABLE Person
CREATE TABLE Person(
        PersonID int NOT NULL UNIQUE,
        LastName varchar(50) NOT NULL,
        FirstName varchar(50) NOT NULL,
        Age int,
        Gender char(1),
        City varchar(255)
);
```

SQL UNIQUE Constraint (cont'd)

- To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

```
DROP TABLE Person
CREATE TABLE Person(
       PersonID int NOT NULL,
       LastName varchar(50) NOT NULL,
       FirstName varchar(50) NOT NULL,
       Age int,
       Gender char(1),
       City varchar(255),
       CONSTRAINT UC_Person UNIQUE (PersonID,LastName)
);
```

SQL UNIQUE Constraint (cont'd)

- SQL UNIQUE Constraint on ALTER TABLE
  - To create a UNIQUE constraint on the "PersonID" column when the table is already created, use the following SQL:

```
ALTER TABLE Person ADD CONSTRAINT UC_PersonID UNIQUE (PersonID);
```

SQL UNIQUE Constraint (cont'd)

- DROP a UNIQUE Constraint
  - To drop a UNIQUE constraint, use the following SQL:

```
ALTER TABLE Person DROP CONSTRAINT UC_PersonID;
```

SQL PRIMARY KEY Constraint

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

SQL PRIMARY KEY Constraint (cont'd)

- SQL PRIMARY KEY on CREATE TABLE
  - The following SQL creates a PRIMARY KEY on the "PersonID" column when the "Person" table is created:

```sql
DROP TABLE Person
CREATE TABLE Person(
        PersonID int NOT NULL PRIMARY KEY,
        LastName varchar(50) NOT NULL,
        FirstName varchar(50) NOT NULL,
        Age int,
        Gender char(1),
        City varchar(255),
);
```

SQL PRIMARY KEY Constraint (cont'd)

- To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
DROP TABLE Person
CREATE TABLE Person(
       PersonID int NOT NULL,
       LastName varchar(50) NOT NULL,
       FirstName varchar(50) NOT NULL,
       Age int,
       Gender char(1),
       City varchar(255),
       CONSTRAINT PK_Person PRIMARY KEY(PersonID,LastName)
);
```

SQL PRIMARY KEY Constraint (cont'd)

- SQL PRIMARY KEY on ALTER TABLE
  - To create a PRIMARY KEY constraint on the "PersonID" column when the table is already created, use the following SQL:

```
ALTER TABLE PersonADD PRIMARY KEY(PersonID);
```

  - To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Person
ADD CONSTRAINT PK_Person PRIMARY KEY(PersonID,LastName);
```

SQL PRIMARY KEY Constraint (cont'd)

- DROP a PRIMARY KEY Constraint

```
ALTER TABLE Person DROP CONSTRAINT PK_Person;
```

SQL FOREIGN KEY Constraint

- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

SQL FOREIGN KEY Constraint (cont'd)

- Look at the following two tables:
  - "Person" table

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

  - "Orders" table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

SQL FOREIGN KEY Constraint (cont'd)

- Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Person" table.
- The "PersonID" column in the "Person" table is the PRIMARY KEY in the "Person" table.
- The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.
- The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

SQL FOREIGN KEY Constraint (cont'd)

- SQL FOREIGN KEY on CREATE TABLE
  - The following SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

```
DROP TABLE Person
CREATE TABLE Person(
        PersonID int NOT NULL PRIMARY KEY,
        LastName varchar(50) NOT NULL,
        FirstName varchar(50) NOT NULL,
        Age int,
        Gender char(1),
        City varchar(255)
);
```

```
CREATE TABLE Orders (
        OrderID int NOT NULL PRIMARY KEY,
        OrderNumber int NOT NULL,
        PersonID int FOREIGN KEY REFERENCES Person (PersonID)
);
```

SQL FOREIGN KEY Constraint (cont'd)

- To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

```
DROP TABLE Orders
CREATE TABLE Orders(
       OrderID int NOT NULL,
       OrderNumber int NOT NULL,
       PersonID int,
       PRIMARY KEY(OrderID),
       CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
       REFERENCES Person(PersonID)
);
```

SQL FOREIGN KEY Constraint (cont'd)

- SQL FOREIGN KEY on ALTER TABLE
  - To create a FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created, use the following SQL:

```
ALTER TABLE Orders ADD FOREIGN KEY (PersonID)
REFERENCES Person(PersonID);
```

  - To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Orders ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES Person(PersonID);
```

SQL FOREIGN KEY Constraint (cont'd)

- DROP a FOREIGN KEY Constraint

```sql
ALTER TABLE Orders DROP CONSTRAINT FK_PersonOrder;
```

SQL CHECK Constraint

- The CHECK constraint is used to limit the value range that can be placed in a column.
- If you define a CHECK constraint on a column, it will allow only certain values for this column.
- If you define a CHECK constraint on a table, it can limit the values in certain columns based on values in other columns in the row.

SQL CHECK Constraint (cont'd)

- SQL CHECK on CREATE TABLE
  - The following SQL creates a CHECK constraint on the "Age" column when the "Person" table is created. The CHECK constraint ensures that the age of a person must be equal or greater than 1, and equal or less than 130:

```
DROP TABLE Person
CREATE TABLE Person(
       PersonID int NOT NULL PRIMARY KEY,
       LastName varchar(50) NOT NULL,
       FirstName varchar(50) NOT NULL,
       Age int CHECK (Age>=1 AND Age<=130),
       Gender char(1),
       City varchar(255)
);
```

SQL CHECK Constraint (cont'd)

- To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

```sql
DROP TABLE Person
CREATE TABLE Person(
       PersonID int NOT NULL PRIMARY KEY,
       LastName varchar(50) NOT NULL,
       FirstName varchar(50) NOT NULL,
       Age int,
       Gender char(1),
       City varchar(255),
       CONSTRAINT CHK_Person CHECK (Age>=1 AND City='Florida')
);
```

SQL CHECK Constraint (cont'd)

- SQL CHECK on ALTER TABLE
  - To create a CHECK constraint on the "Age" column when the table is already created, use the following SQL:

```
ALTER TABLE Person ADD CHECK (Age>=1);
```

  - To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Person
ADD CONSTRAINT CHK_PersonAge
CHECK (Age>=1 AND City='Florida');
```

SQL CHECK Constraint (cont'd)

- DROP a CHECK Constraint
  - To drop a CHECK constraint, use the following SQL:

```
ALTER TABLE Person DROP CONSTRAINT CHK_PersonAge;
```

SQL DEFAULT Constraint

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

SQL DEFAULT Constraint (cont'd)

- SQL DEFAULT on CREATE TABLE
  - The following SQL sets a DEFAULT value for the "City" column when the "Person" table is created:

```
DROP TABLE Person
CREATE TABLE Person(
        PersonID int NOT NULL PRIMARY KEY,
        LastName varchar(50) NOT NULL,
        FirstName varchar(50) NOT NULL,
        Age int,
        Gender char(1),
        City varchar(255) DEFAULT 'California'
);
```

SQL DEFAULT Constraint (cont'd)

- SQL DEFAULT on CREATE TABLE
  - The DEFAULT constraint can also be used to insert system values, by using functions like GETDATE():

```
DROP TABLE Orders
CREATE TABLE Orders(
       ID int NOT NULL,
       OrderNumber int NOT NULL,
       OrderDate date DEFAULT GETDATE()
);
```

SQL DEFAULT Constraint (cont'd)

- SQL DEFAULT on ALTER TABLE
  - To create a DEFAULT constraint on the "City" column when the table is already created, use the following SQL:

```
ALTER TABLE Person ADD CONSTRAINT df_City
DEFAULT 'California' FOR City;
```

  - To drop a DEFAULT constraint, use the following SQL:

```
ALTER TABLE Person DROP CONSTRAINT df_City
```

AUTO INCREMENT Field

- Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.
- Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

AUTO INCREMENT Field (cont'd)

- The following SQL statement defines the "PersonID" column to be an auto-increment primary key field in the "Person" table:

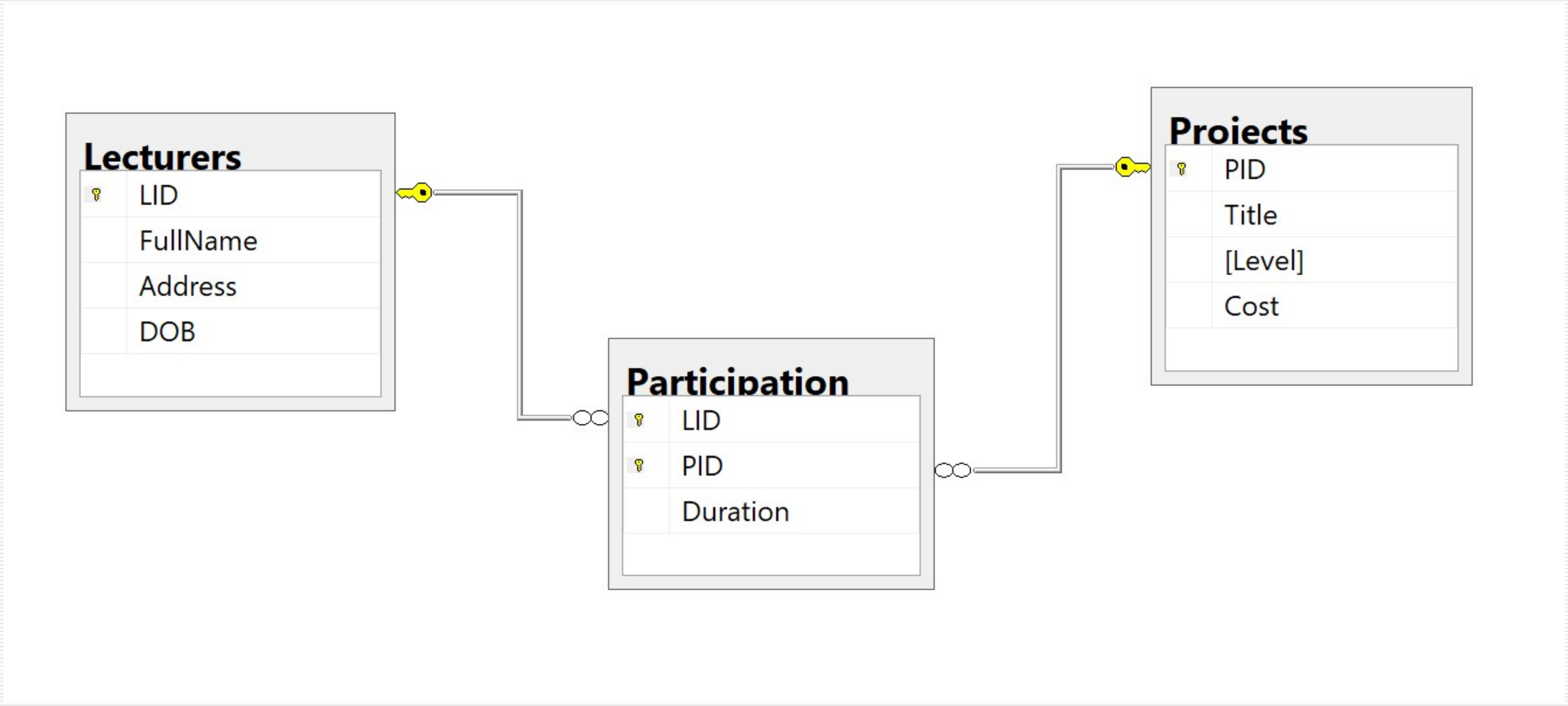
```
DROP TABLE Person
CREATE TABLE Person(
        PersonID int identity(1,1) PRIMARY KEY,
        LastName varchar(50) NOT NULL,
        FirstName varchar(50) NOT NULL,
        Age int,
        Gender char(1),
        City varchar(255)
);
```

Some database

- Lecturers-Projects-Participation
- Company-Supply-Product
- BikeStores
- SinhVien-DangKy-MonHoc

Lecturers-Projects-Participation

- Database Diagram



Lecturers-Projects-Participation (cont'd)

```
CREATE TABLE Lecturers(
  LID char(4) NOT NULL,
  FullName nchar(30) NOT NULL,
  Address nvarchar(50) NOT NULL,
  DOB date NOT NULL,
  CONSTRAINT pkLecturers PRIMARY KEY (LID)
)
```

```
CREATE TABLE Projects(
  PID char(4) NOT NULL,
  Title nvarchar(50) NOT NULL,
  Level nchar(12) NOT NULL,
  Cost integer,
  CONSTRAINT pkProjects PRIMARY KEY (PID)
)
```

Lecturers-Projects-Participation (cont'd)

```
CREATE TABLE Participation(
  LID char(4) NOT NULL,
  PID char(4) NOT NULL,
  Duration smallint,
  CONSTRAINT pkParticipation PRIMARY KEY (LID, PID),
  CONSTRAINT fk1 FOREIGN KEY (LID) REFERENCES Lecturers (LID),
  CONSTRAINT fk2 FOREIGN KEY (PID) REFERENCES Projects (PID)
)
```

Lecturers-Projects-Participation (cont'd)

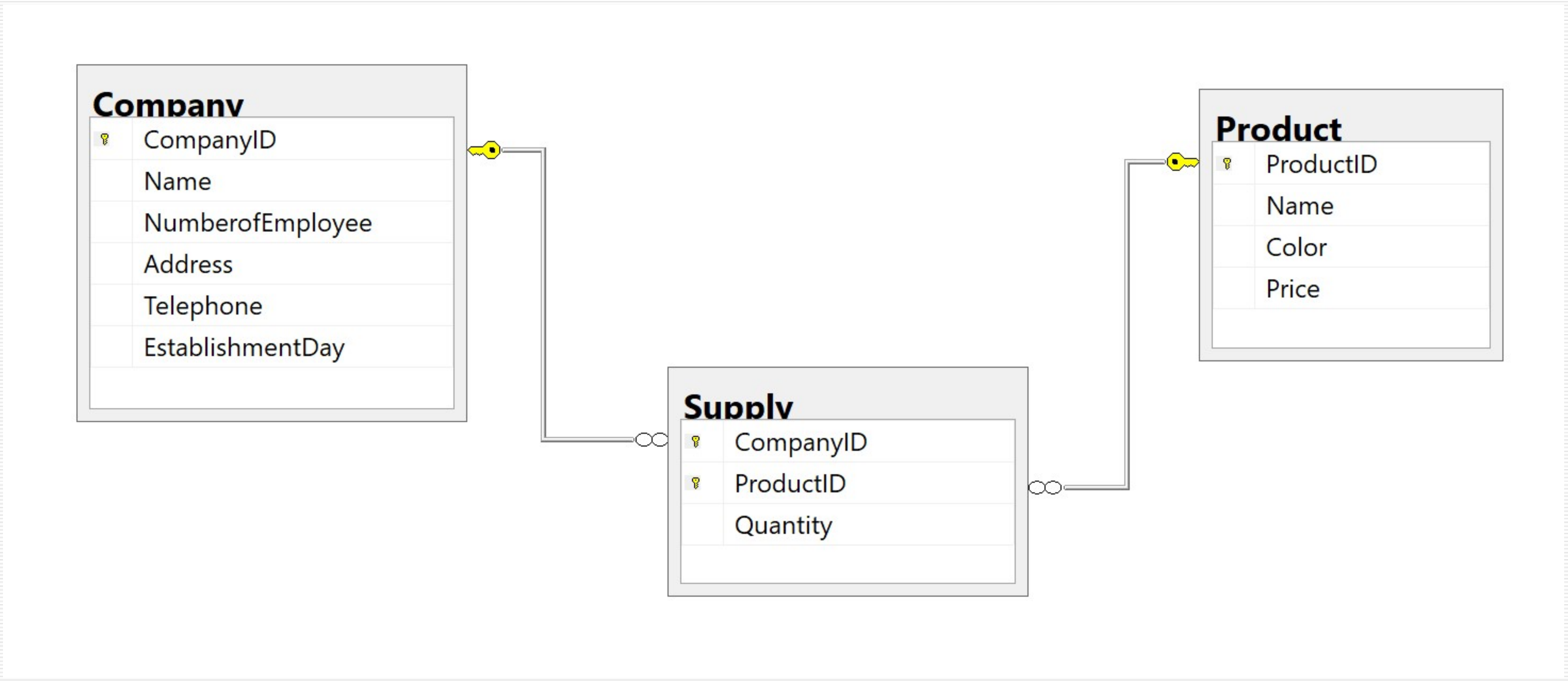
```
INSERT INTO Lecturers VALUES('GV01',N'Vũ Tuyết Trinh',N'Hoàng Mai, Hà Nội',
'1975/10/10'),
('GV02',N'Nguyễn Nhật Quang',N'Hai Bà Trưng, Hà Nội','1976/11/03'),
('GV03',N'Trần Đức Khánh',N'Đống Đa, Hà Nội','1977/06/04'),
('GV04',N'Nguyễn Hồng Phương',N'Tây Hồ, Hà Nội','1983/12/10'),
('GV05',N'Lê Thanh Hương',N'Hai Bà Trưng, Hà Nội','1976/10/10')

INSERT INTO Projects VALUES ('DT01',N'Tính toán lưới',N'Nhà nước','700'),
('DT02',N'Phát hiện tri thức',N'Bộ','300'),
('DT03',N'Phân loại văn bản',N'Bộ','270'),
('DT04',N'Dịch tự động Anh Việt',N'Trường','30')

INSERT INTO Participation VALUES ('GV01','DT01','100'),
('GV01','DT02','80'),
('GV01','DT03','80'),
('GV02','DT01','120'),
('GV02','DT03','140'),
('GV03','DT03','150'),
('GV04','DT04','180')
```

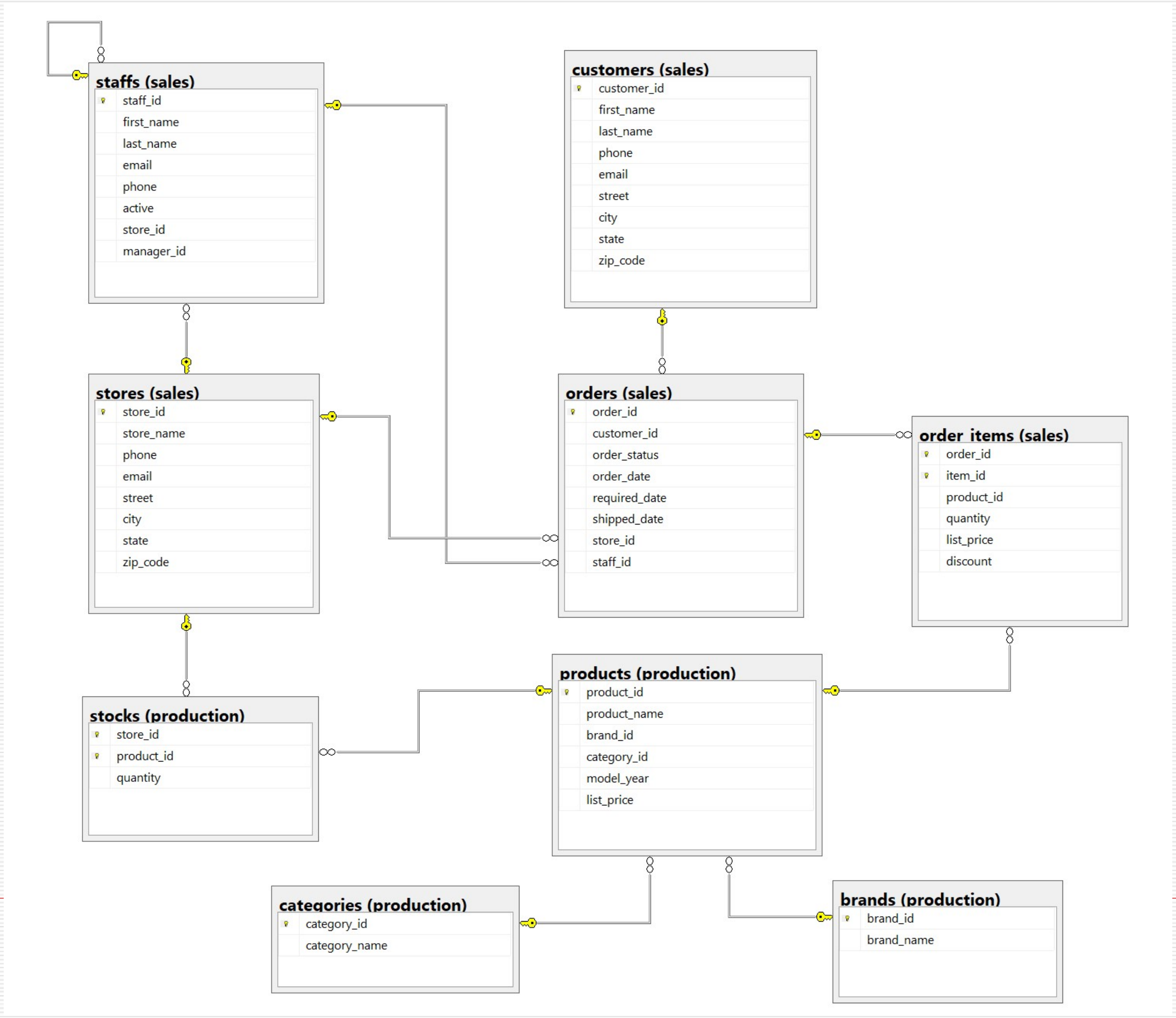
Company-Supply-Product

- Database diagram



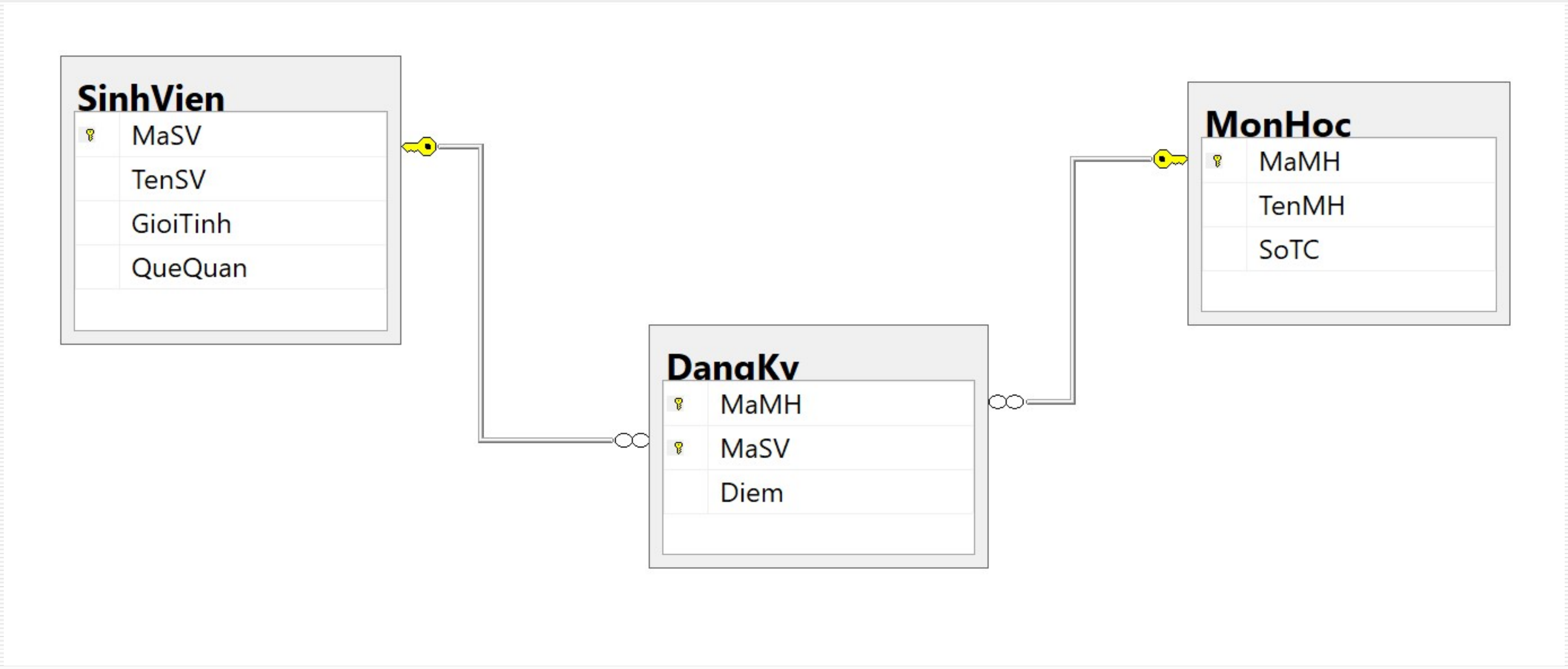
BikeStores

- Database diagram



SinhVien-DangKy-MonHoc

- Database diagram


SinhVien
MaSV
TenSV
GioiTinh
QueQuan
DangKy
MaMH
MaSV
Diem
MonHoc
MaMH
TenMH
SoTC