SQL – Database manipulation

- SQL provides operations for changing the data in the database:
  - insert – for adding new rows
  - delete – for removing existing rows
  - update – for changing existing rows

SQL – Database manipulation

- Insert command has two forms:
  - `insert into table [(column_list)] values (constants)`
    - This form is used for inserting a single row the content of which is given as constant values
  - `insert into table [(column_list)] query`
    - This form is used for adding the result rows of a query into the target table

SQL – Database manipulation

```sql
CREATE TABLE course (
    ccode  numeric(8) NOT NULL ,
    name  varchar(40) NOT NULL ,
    credit_units  numeric(5,1) NOT NULL ,
    lecturer  varchar(12),
    PRIMARY KEY (ccode ),
    FOREIGN KEY (lecturer) REFERENCES teacher)
```

```sql
insert into course values
    (1234,'Introduction to Databases',2,'HLAINE');
```
- A complete row is inserted

SQL – Database manipulation

```sql
insert into course values
    (1234,'Introduction to Databases',2,NULL); OR
insert into course (ccode, name, credit_units)
    values (1234, 'Introduction to Databases',2);
```
- Column list is used when all the values are not provided
- When insert statements are used within a program it is good programming practice to use column lists always – new columns may be added and the inserts still work

SQL – Database manipulation

```sql
CREATE TABLE registration (    ccode numeric(8) not null,    groupNo numeric(2) not null,    studentNo  numeric(5) NOT NULL ,    dateRegistered  date NOT NULL ,    PRIMARY KEY (studentNo, ccode) ,    FOREIGN KEY (ccode, groupNo)  REFERENCES exerciseGroup on delete cascade,    FOREIGN KEY (studentNo, ccode) REFERENCES student )
```

SQL – Database manipulation

```sql
If we do not know the lecturer’s identifier but only his name, we may use the following query

```sql
insert into course
    select 1234, 'Introduction to Databases', 2, teacher_id from teacher
    where name='Laine Harri';
```
- This works as expected, if there is only one teacher with the given name. Otherwise, it fails to insert the row either because there is no teacher or there are many and the operation violates the key constraint.
SQL – Database manipulation

In Oracle SQL:

- **Insert into registration**
  
  ```sql
  select java.ccode, groupNo, studentNo, sysdate
  from course java, course intro, registration
  where java.name='Java-programming' and
  intro.name='Introduction to Programming' and
  intro.ccode=registration.ccode;
  ```

  java course is not connected, but there is only one record that satisfies the selection condition.

- **Changing the rows (update)**

  ```sql
  update table
  set column1=expression1 [, column2=expression2, …]
  [where constraints on the target ]
  ```

  - Many rows within the same table may be changed at the same time,
  - All the rows that meet with the constraints in the where part may are changed – sub queries may be used in the where part
  - If where part is missing, all the rows are changed

- **Add one credit unit for the course Java-programming**

  ```sql
  update course
  set credit_units= 2*credit_units
  where name='Java-programming';
  ```

  Operation fails, if it violates the integrity constraints.

- **Removing rows (delete)**

  ```sql
  delete from table
  [where conditions to select the rows ];
  ```

  - All rows that satisfy the conditions are deleted
  - If where part is missing all rows in the table are removed
  - Operation fails if integrity constraints are violated

- **Remove all exercise groups that have no students.**

  ```sql
  delete from exerciseGroup
  where (ccode, groupNo) not in
  (select ccode, groupNo
  from registration);
  ```

- **Moving rows from one table to another is used, for example, to keep the active tables small, in which case the passive records are moved to history tables. Here is an example**

  ```sql
  insert into registration_history
  select * from registration where dateRegistered<'1.1.2004';
  delete from registration where dateRegistered<'1.1.2004';
  ```
A collection of operations that must all be executed as a single unit is called a database transaction. A transaction must be done as a whole not only partially. Money transfer for example consists of two operations:

- `update account set balance=balance-500 where accountNo=123456;`
- `update account set balance=balance+500 where accountNo=654321;`

DBMS guarantees:
- Transactions are not carried out only partially.
- Outsiders cannot see the intermediate results (for example that money is taken from one account but not yet added to the other).
- Changes made on the database may be cancelled if the transaction has not been committed.
- When the transaction is committed the changes become permanent and visible also to the outsiders.

To commit a transaction issue the command:

```
commit [work]
```

This ends the previous transaction and starts a new one. Instead of committing a transaction, it may be cancelled with the command:

```
rollback [work]
```

Thus the money transfer would be:

```
commit;
update account set balance=balance-500 where accountNo=123456;
update account set balance=balance+500 where accountNo=654321;
commit;
```

We try to remove groups that do not have students. There is an `on delete cascade` defined for the foreign key in registration:

```
commit;
select count(*) from registrations;
>> 3500 <<
delete from exerciseGroup where groupNo is not null;
select count(*) from registrations;
>>> 0 <<<
rollback;
select count(*) from registrations;
>> 3500 <<
```

Not correct