# COMP 3311
# Database Management Systems

Lab 4

Simple SQL DDL, DML and Constraint Statements

## Lab Objectives

- After this lab you should be able to:
  - issue simple Data Definition Language (DDL) statements.
  - issue simple Data Manipulation Language (DML) statements.
  - specify simple integrity constraint statements.

## Data Definition Language (1)

- The Data Definition language (DDL) is a language for specifying the database schemes (in the form of tables).
- It enables tables to be created and altered.
- The basic SQL DDL statements are:
  - create - create a new table
  - rename - rename an existing table
  - drop - drop an existing table
  - alter - add an attribute to/drop an attribute from a table, change an attribute's data type or add/alter a table's constraints.

## Data Definition Language (2)

- Create a new table.

  create table *table_name* (*attribute1 datatype*, *attribute2 datatype*, …);

  Example:
  ```
  create table Facility (
          departmentId          char(4) not null,
          departmentName        varchar2(40),
          numberProjectors      int,
          numberComputers       int);
  ```

- Rename an existing table.

  rename *old_table* to *new_table;*

  Example: `rename Facility to RenameTest;`

# Data Definition Language (3)

- Drop an existing table.

  drop table *table_name*;

  Example: drop table RenameTest;

- Add new attributes to an existing table.

  alter table *table_name* add (*attribute1 datatype*, *attribute2 datatype*, …);

  Example: alter table Facility add (funding number(10,2));

## Data Definition Language (4)

- Change the data type of table attributes.

  alter table *table_name* modify (*attribute1 datatype*, *attribute2 datatype*, …);

  Example: alter table Facility modify (funding varchar2(10));

- Delete an attribute from an existing table.

  alter table *table_name* drop (*attribute1*, *attribute2*, …);

  Example: alter table Facility drop (funding);

## Data Manipulation Language (1)

- The Data Manipulation language (DML) is a language for manipulating data in a database.
- The basic SQL DML statements for modifying data are:
  - insert - inserts tuples into an existing table
  - update - updates tuples of an existing table
  - delete - removes tuples from an existing table

## Data Manipulation Language (2)

- Insert tuples into an existing table.

  insert into *table_name* (*attribute1*, *attribute2*, …) values (*value1*, *value2*, …)

  Example:
  ```
  insert into Facility (departmentId, departmentName,
  numberProjectors, numberComputers)
  values ('HUMA', 'Humanities', 2, 10);
  ```

- You can omit the attribute names, if you insert tuples with *values for all the attributes* present and *in the order in which the attributes are defined in the table*.

  Example:
  ```
  insert into Facility values ('PHYS', 'Physics', 8, 12);
  ```

## Data Manipulation Language (3)

- By stating explicitly the attributes, you can insert partial tuples with some of the attributes being absent, as long as these attributes do not have the NOT NULL constraint (covered later in these lab notes).

Example:

```
insert into Facility (departmentId, departmentName)
values ('test', 'test name');
```

## Data Manipulation Language (4)

- Update tuples of an existing table.

**update** *table_name* **set** *attribute*=*value* [**where** *conditions*];

Example:

```
update Facility set numberComputers=200
where departmentId= 'COMP';
```

# Data Manipulation Language (5)

- Remove tuples from an existing table.

  delete from *table_name* [where *conditions*];

  Example: delete from Facility where departmentId='test';

- The following statement removes all tuples from the table Facility.

  Example: delete from Facility;

## Specifying Integrity Constraints (1)

- One way to ensure that changes made to the database do not affect data consistency, is to specify integrity constraints on the database.
- Integrity constraints can be declared at the attribute level or at the table level.
- Attribute-level constraints apply to the attributes only. Each constraint involves one attribute.
- Table-level constraints apply to the whole table and usually involve multiple attributes.

## Specifying Integrity Constraints (2)

- The basic constraint keywords are:
  - Primary Key: specifies the attribute(s) that are used to uniquely identify the tuples (records) in a table.
  - Foreign Key: specifies the attribute(s) whose value refers to another table and which value must be present in that table.
  - Unique: indicates the attribute has unique values.
  - Not Null: indicates the attribute must have a value.
  - Check: places conditions (in the form of a predicate) on the attribute.

## Specifying Integrity Constraints (3)

- Oracle allows applying the constraint either at the attribute level *or* at the table level.

  **Exception:** The Not Null constraint can only be applied as an attribute-level constraint.

- Attribute-level constraints are placed right after the attribute definitions.

- Table-level constraints are placed after all the definitions of the attributes.

**Note:** A constraint that refers to a table can only be defined after that table is created. Thus, the order in which you define tables and constraints is important.

## Specifying Integrity Constraints (4)

- Examples:

```
create table Staff (
        staffId   int primary key,
        age       int check (age between 0 and 65),
        salary    number(10,2) check (salary>0));

create table WorksAt (
        staffId        int references Staff(staffId),
        firmName       varchar2(100) not null,
        primary key(staffId, firmName));
```

*Attribute-Level Constraints*

*Table-Level Constraint*

## Specifying Integrity Constraints (5)

- Examples:

The following two statements are identical. Note that all the constraints in the second create statement were given names (in italic font).

```
create table WorksAt (
        staffId        int references Staff (staffId),
        firmName       varchar2(100) not null,
        primary key (staffId, firmName));

create table WorksAt (
        StaffId        int,
        firmName       varchar2(100) constraint not_null_firmName not null,
        constraint foreign_key foreign key (staffId) references Staff (staffId),
        constraint primary_key primary key(staffId, firmName));
```

## Specifying Integrity Constraints (6)

- We can drop a non-primary key constraint.

  Example: `alter table WorksAt drop constraint not_null_firmName;`

- We can also drop a primary key

  Example: `alter table WorksAt drop primary key;`

  or add it back.

  Example:
  ```
  alter table WorksAt
  add new_primary_key primary key (staffId, firmName);
  ```

# Specifying Integrity Constraints (7)

- One can add or modify constraints in an existing table, using the alter table statement.

Example:
```
alter table Staff
add constraint CHK_age check (age between 20 and 40);
```

Example:
```
alter table Staff
modify (age not null);
```

- To list all the constraints for a table, use the query:

```
select *
from user_constraints
where table_name='<table_name>';
```

<u>Important Note</u>
*<table_name>* must be all uppercase.

## Summary

- We covered the following topics in this lab.
  - Specifying simple DDLs
    - create, rename, drop, alter
  - Specifying simple DMLs
    - insert, update, delete
  - Specifying simple integrity constraints.
    - primary key, foreign key, unique, not null, check

## Lab Exercise

- You must complete the lab exercise and upload the result to Canvas by **11:59 p.m. this Friday**.

**Ask for help if you need it!**

**IMPORTANT NOTE**

If you want to save your modified SQL script files, copy them to the M drive or to a USB drive as any personal files on the lab computers will be automatically deleted periodically.