COMP 3311
Database Management Systems

Lab 5

Simple SQL DDL, DML and Constraint Statements

Lab Topics

- ☐ 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

Data Definition Language (DDL)

- ☐ The Data Definition language (DDL) is used to specify/alter the database schema (i.e., the table definitions).

- ☐ The basic SQL DDL statements are:

  - ■ create table - create a new table

  - ■ rename - rename an existing table

  - ■ drop table - drop an existing table

  - ■ alter table - add/drop an attribute or change an attribute's data type;
    add/drop/change table constraints.

Example SQL DDL (1)

☐ Create a new table.

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

Example:　create table Facility (
　　　　　departmentId　　　　char(4) not null,
　　　　　numberProjectors　　int default 0,
　　　　　numberComputers　　int default 0);

☐ Rename an existing table.

rename *old_table* to *new_table*;

Example:　rename Facility to RenameTest;

☐ Drop an existing table.

drop table *table_name*;

Example:　drop table RenameTest;

Example SQL DDL (2)

- ☐ 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));

- ☐ 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 (DML)

- [ ] The Data Manipulation language (DML) is used to manipulate data in a database.

- [ ] Besides the select DML statement for retrieving data, SQL also provides the following DML statements for modifying data:

  - insert   - inserts tuples into an existing table

  - update   - updates tuples of an existing table

  - delete   - removes tuples from an existing table

Example SQL DML (1)

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

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

The attribute names can be omitted, if tuples are inserted 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', 8, 12);

By stating explicitly the attributes, partial tuples can be inserted with some of the attributes being absent, if these attributes do not have the **not null** constraint (covered later in these lab notes).

Example: insert into Facility (departmentId) values ('test');

Example SQL DML (2)

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

    Example: update Facility set numberComputers=200
                       where departmentId='COMP';

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

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

    The delete statement without a where clause removes all tuples from a table.

    Example: delete from Facility;

Integrity Constraints

- Integrity constraints are used to ensure data consistency and can be declared at the attribute level or at the table level.

- Attribute-level constraints apply to the attributes only and only involve one attribute.

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

- Table-level constraints apply to the whole table and usually involve multiple attributes.

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

Specifying Table Integrity Constraints

☐ The basic integrity 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.

☐ To list all constraints of a table, use the query:

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

Important Note
<table_name> must be all uppercase.

Example Integrity Constraints (1)

```
create table Staff (
    staffId    int primary key,
    age        int,
    email      varchar2(20) unique,
    salary     number(10,2) check (salary>0));


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

Attribute-Level Constraints

Table-Level Constraint

Note: not null can only be an attribute-level constraint.

Example Integrity Constraints (2)

```
create table WorksAt (
    staffId         int references Staff (staffId) on delete cascade,
    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) on delete
                cascade,
    constraint primary_key primary key(staffId, firmName));
```

> These two create statements are identical. The constraints in the second create statement were given names (in *italic font*).

Note: A constraint that refers to a table can only be defined <u>after</u> that table is created. Thus, the order in which tables and constraints are define is important (e.g., the Staff table must be defined before the WorksAt table since the WorksAt table references the Staff table).

Modifying Table Integrity Constraints

□ The alter table statement is used to add or modify constraints in a table.

- **Add a primary key**

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

- **Drop a primary key**

  alter table WorksAt drop primary key;

- **Add a constraint**

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

- **Drop a constraint**

  alter table WorksAt drop constraint not_null_firmName;

- **Modify a constraint**

  alter table Staff modify (age not null);