

INTRODUCTION TO DATABASES

TEAM 6

ZhangBank - The place for notes Part 4

Author:

Ian LOGAN
Cameron LOPEZ
Anton MOCZYGEMBA
Isaac NOOJIN

Professor:

Weining ZHANG

November 13, 2012

Contents

1	Description	3
2	Design	3
2.1	Users	3
2.2	Roles	4
2.3	Documents	4
2.4	Tags	4
2.5	Professors	4
2.6	Courses	4
3	Relational Schema	4
3.1	User Table	4
3.1.1	Keys	5
3.1.2	Functional Dependencies	5
3.1.3	Normal Form	6
3.2	Role Table	6
3.2.1	Keys	6
3.2.2	Functional Dependencies	6
3.2.3	Normal Form	6
3.3	UserRoles Table	6
3.3.1	Keys	6
3.3.2	Functional Dependencies	6
3.3.3	Normal Form	6
3.4	Professor Table	7
3.4.1	Keys	7
3.4.2	Functional Dependencies	7
3.4.3	Normal Form	7
3.5	Course Table	7
3.5.1	Keys	7
3.5.2	Functional Dependencies	7
3.5.3	Normal Form	7
3.6	Takes Table	8
3.6.1	Keys	8
3.6.2	Functional Dependencies	8
3.6.3	Normal Form	8
3.7	Teaches Table	8
3.7.1	Keys	8
3.7.2	Functional Dependencies	8
3.7.3	Normal Form	8
3.8	Document Table	9
3.8.1	Keys	9
3.8.2	Functional Dependencies	9
3.8.3	Normal Form	9
3.9	UserDocs Table	9

3.9.1	Keys	9
3.9.2	Functional Dependencies	9
3.9.3	Normal Form	9
3.10	Tag Table	9
3.10.1	Keys	9
3.10.2	Functional Dependencies	10
3.10.3	Normal Form	10
3.11	DocTag Table	10
3.11.1	Keys	10
3.11.2	Functional Dependencies	10
3.11.3	Normal Form	10
4	Database	10
4.1	User	10
4.2	Role	11
4.3	UserRoles	11
4.4	Professor	12
4.5	Course	12
4.6	Takes	12
4.7	Teaches	13
4.8	UserDocs	13
4.9	Tag	14
4.10	DocTag	14
5	Views	15
5.1	Takes	15
5.2	UserDocs	15
5.3	Document Tags	16
5.4	Professor Documents	16
5.5	Course Docuements	16
5.6	User Roles	17
6	Spool	17
7	SQL Queries	23
7.1	Updates	23
8	Triggers	23
8.1	Before User Delete	23

1 Description

Our application seeks to fill the needs of students everywhere. ZhangBank's goal is to organize class material study guides; basically anything that can help the class rise up and meet the expectations of their Professors. Identifiable entities include user accounts, Roles, Documents (in many formats), Courses, Professors, and semesters. An organized way to find and view Documents will be implemented, as well as add content. A user's profile will keep track of which Courses students are taking or are interested in. An interesting problem would be correctly displaying each arbitrary Document. Data for our application can be generated from our own Courses and other free online Courses.

2 Design

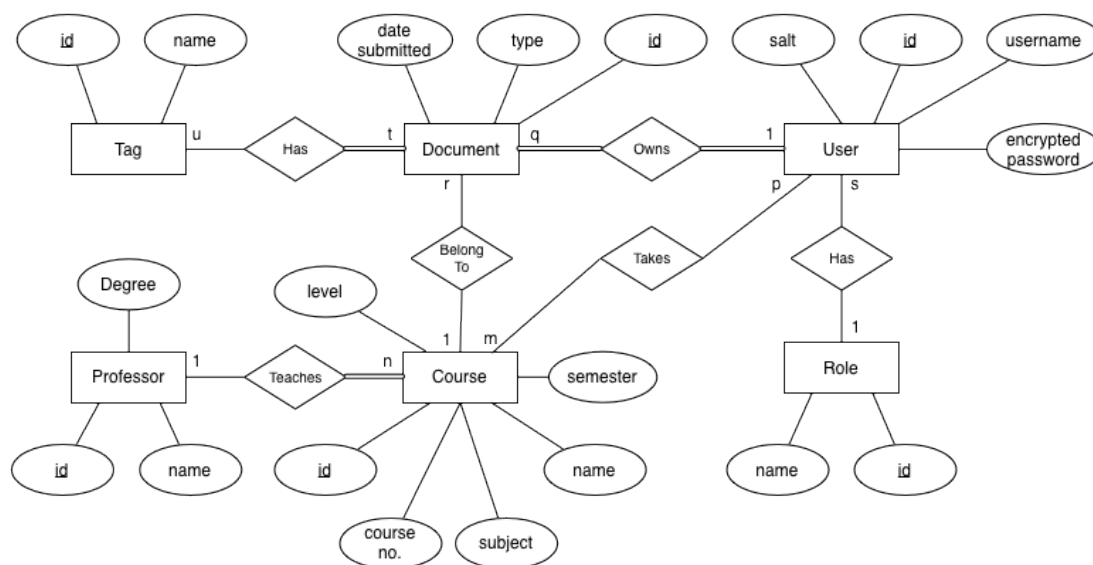


Figure 1: ER Diagram

2.1 Users

Each User creates a username a password during account creation, a security salt is also generated with each account. User can be identified uniquely by an assigned id.

All Users have one Role associated with it to allow authorization of application management. All Users can take many Courses which will allow Users to keep track of the Documents of the Courses they're taking. Each User can upload many Documents. The Documents they own can be managed by them.

2.2 Roles

A list of Roles is maintained, each with different capabilities in our application. Normal Users can add and manage their own Documents. An admin can manage all Documents, Courses, and Professors. It's identified by a generated id and a provided name.

Each Role can have many Users to allow roll based authorization.

2.3 Documents

Each Document has a type associated with it to allow the application to display Documents appropriately. It stores the date submitted to help with organization in the application and can be uniquely identified by a generated id.

All Documents are owned by one User each, the original uploader. All Documents belong to one Course each. This allows for the indexing of Documents by Course. All Documents can have many tags each. This allows documents to be organized in a tag based fashion.

2.4 Tags

Each tag has a provided name and an id. This allows for an indexing of tags by name. Every tag has multiple Documents each. This allows Documents to be organized for each course.

2.5 Professors

The Professor entity has two primary attributes, a provided name and a generated id.

Each Professor entity Teaches many Courses. This will allow Users to run a search on a specific Professor to view Documentation for any Course that he may have previously taught.

2.6 Courses

The Course entity has four primary attributes; a provided name, a generated id, the semester the course is held, and a provided course number.

All Courses are Taught by one Professor each to allow the indexing of Documents based on Professor. Each Course has many Documents to provide indexing of Documents based on each Course. All Courses are Taken by many Users each. This allows for Users to save which classes they're taking.

3 Relational Schema

3.1 User Table

Table 1: User Table

<u>id</u>	username	password	salt
-----------	----------	----------	------

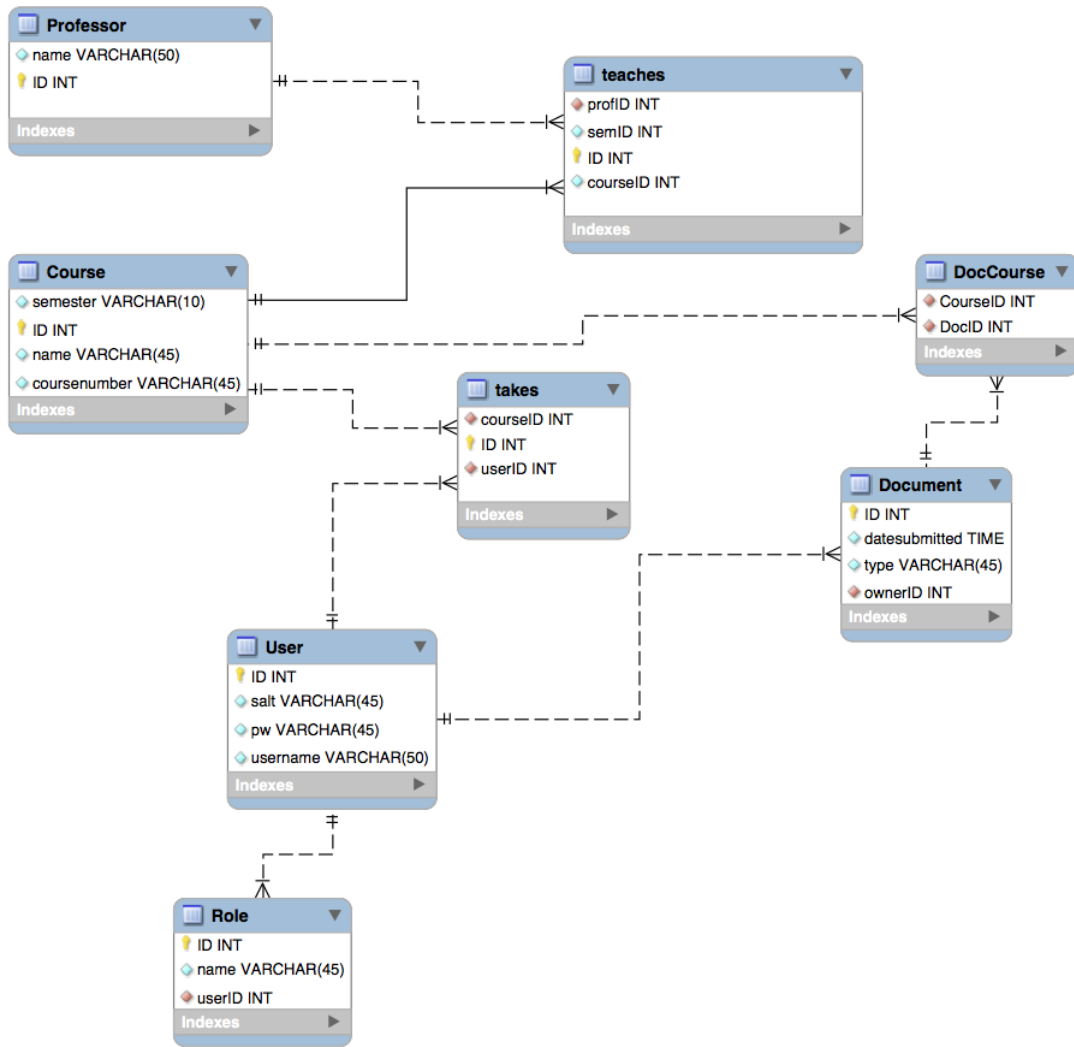


Figure 2: Schema Diagram

3.1.1 Keys

Primary key: id

Candidate keys: id, username

3.1.2 Functional Dependencies

$id \rightarrow username, password, salt$

$username \rightarrow id, password, salt$

3.1.3 Normal Form

BCNF

3.2 Role Table

Table 2: Role Table

<u>id</u>	name
-----------	------

3.2.1 Keys

Primary key: id

Candidate keys: id, name

3.2.2 Functional Dependencies

$\text{id} \rightarrow \text{name}$

3.2.3 Normal Form

BCNF

3.3 UserRoles Table

Table 3: UserRole Table

<u>*user_id*</u>	role_id
------------------	----------------

3.3.1 Keys

Primary key: user_id

Candidate keys: user_id

Foreign keys: user_id \rightarrow User.id, role_id \rightarrow Role.id

3.3.2 Functional Dependencies

$\text{user_id} \rightarrow \text{role_id}$

3.3.3 Normal Form

BCNF

3.4 Professor Table

Table 4: Professor Table

<u>id</u>	name
-----------	------

3.4.1 Keys

Primary key: id

Candidate keys: id

3.4.2 Functional Dependencies

$\text{id} \rightarrow \text{name}$

3.4.3 Normal Form

BCNF

3.5 Course Table

Table 5: Course Table

<u>id</u>	course_no.	name	semester
-----------	------------	------	----------

3.5.1 Keys

Primary key: id

Candidate keys: id

3.5.2 Functional Dependencies

$\text{id} \rightarrow \text{course_no}, \text{name}, \text{semester}$

3.5.3 Normal Form

BCNF

Table 6: Takes Table

id | **course_id** | **user_id**

3.6 Takes Table

3.6.1 Keys

Primary key: id

Candidate keys: id

Foreign keys: $\text{course_id} \rightarrow \text{Course.id}$, $\text{user_id} \rightarrow \text{User.id}$

3.6.2 Functional Dependencies

$\text{id} \rightarrow \text{course_id}, \text{user_id}$

3.6.3 Normal Form

BCNF

3.7 Teaches Table

Table 7: Teaches Table

course_id | **professor_id**

3.7.1 Keys

Primary key: course_id

Candidate keys: course_id

Foreign keys: $\text{course_id} \rightarrow \text{Course.id}$, $\text{professor_id} \rightarrow \text{Professor.id}$

3.7.2 Functional Dependencies

$\text{id} \rightarrow \text{course_id}, \text{professor_id}$

3.7.3 Normal Form

BCNF

Table 8: Document Table

id | type | date_submitted

3.8 Document Table

3.8.1 Keys

Primary key: id

Candidate keys: id

3.8.2 Functional Dependencies

$\text{id} \rightarrow \text{type}, \text{date_submitted}$

3.8.3 Normal Form

BCNF

3.9 UserDocs Table

Table 9: UserDoc Table

document_id | user_id

3.9.1 Keys

Primary key: document_id

Candidate keys: document_id

Foreign keys: document_id \rightarrow Document.id, user_id \rightarrow User.id

3.9.2 Functional Dependencies

$\text{document_id} \rightarrow \text{user_id}$

3.9.3 Normal Form

BCNF

3.10 Tag Table

3.10.1 Keys

Primary key: id

Candidate keys: id, name

Table 10: Tag Table

id | name

3.10.2 Functional Dependencies

$id \rightarrow name$

3.10.3 Normal Form

BCNF

3.11 DocTag Table

Table 11: DocTag Table

id | document_id | tag_id

3.11.1 Keys

Primary key: id

Candidate keys: id

Foreign keys: document_id \rightarrow Document.id, tag_id \rightarrow Tag.id

3.11.2 Functional Dependencies

$id \rightarrow document_id, tag_id$

3.11.3 Normal Form

BCNF

4 Database

4.1 User

```
CREATE TABLE "USERS"  
(  
    "ID" NUMBER NOT NULL ENABLE,  
    "SALT" VARCHAR2(45) NOT NULL ENABLE,  
    "PW" VARCHAR2(45) NOT NULL ENABLE,  
    "USERNAME" VARCHAR2(50) NOT NULL ENABLE,  
    CONSTRAINT "USERS_PK" PRIMARY KEY ("ID") ENABLE  
)
```

```

/

CREATE OR REPLACE TRIGGER "BI_USERS"
  before insert on "USERS"
  for each row
begin
  select "USERS_SEQ".nextval into :NEW.ID from dual;
end;

/

ALTER TRIGGER "BI_USERS" ENABLE

/

```

4.2 Role

```

CREATE TABLE "ROLE"
(
  "ID" NUMBER,
  "NAME" VARCHAR2(45),
  CONSTRAINT "ROLE_PK" PRIMARY KEY ("ID") ENABLE
)

/

CREATE OR REPLACE TRIGGER "BI_ROLE"
  before insert on "ROLE"
  for each row
begin
  select "ROLE_SEQ".nextval into :NEW.ID from dual;
end;

/

ALTER TRIGGER "BI_ROLE" ENABLE

/

```

4.3 UserRoles

```

CREATE TABLE "USERROLE"
(
  "USER_ID" NUMBER NOT NULL ENABLE,
  "ROLE_ID" NUMBER NOT NULL ENABLE,
  CONSTRAINT "USERROLE_PK" PRIMARY KEY ("USER_ID") ENABLE,
  CONSTRAINT "USERROLE_FK" FOREIGN KEY ("USER_ID")
    REFERENCES "USERS" ("ID") ENABLE,
  CONSTRAINT "USERROLE_FK2" FOREIGN KEY ("ROLE_ID")
    REFERENCES "ROLE" ("ID") ENABLE
)

/

```

```

CREATE OR REPLACE TRIGGER "BI_USERROLE"
  before insert on "USERROLE"
  for each row
begin
  select "USERROLE_SEQ".nextval into :NEW.USER_ID from dual;
end;

/
ALTER TRIGGER "BI_USERROLE" ENABLE
/

```

4.4 Professor

```

CREATE TABLE "PROFESSOR"
(
  "NAME" VARCHAR2(50) NOT NULL ENABLE,
  "ID" NUMBER(*,0) NOT NULL ENABLE,
  "DEGREE" VARCHAR2(45),
  PRIMARY KEY ("ID") ENABLE
)
/

```

4.5 Course

```

CREATE TABLE "COURSE"
(
  "SEMESTER" VARCHAR2(10) NOT NULL ENABLE,
  "ID" NUMBER(*,0) NOT NULL ENABLE,
  "TITLE" VARCHAR2(45) NOT NULL ENABLE,
  "COURSENUMBER" VARCHAR2(45) NOT NULL ENABLE,
  "ACADEMICLEVEL" NUMBER,
  "SUBJECT" VARCHAR2(50) NOT NULL ENABLE,
  PRIMARY KEY ("ID") ENABLE
)
/

```

4.6 Takes

```

CREATE TABLE "TAKES"
(
  "COURSEID" NUMBER NOT NULL ENABLE,
  "ID" NUMBER NOT NULL ENABLE,
  "USERID" NUMBER NOT NULL ENABLE,
  CONSTRAINT "TAKES_PK" PRIMARY KEY ("ID") ENABLE,
  CONSTRAINT "TAKES_FK" FOREIGN KEY ("COURSEID")
    REFERENCES "COURSE" ("ID") ENABLE,
  CONSTRAINT "TAKES_FK2" FOREIGN KEY ("USERID")
    REFERENCES "USERS" ("ID") ENABLE
)

```

```

/

CREATE OR REPLACE TRIGGER  "BI_TAKES"
    before insert on "TAKES"
    for each row
begin
    select "TAKES_SEQ".nextval into :NEW.ID from dual;
end;

/

ALTER TRIGGER  "BI_TAKES" ENABLE

/

```

4.7 Teaches

```

CREATE TABLE  "TEACHES"
(
    "PROFID" NUMBER NOT NULL ENABLE,
    "COURSEID" NUMBER NOT NULL ENABLE,
    CONSTRAINT "TEACHES_FK" FOREIGN KEY ("PROFID")
        REFERENCES  "PROFESSOR" ("ID") ENABLE,
    CONSTRAINT "TEACHES_FK2" FOREIGN KEY ("COURSEID")
        REFERENCES  "COURSE" ("ID") ENABLE
)

/

CREATE OR REPLACE TRIGGER  "BI_TEACHES"
    before insert on "TEACHES"
    for each row
begin
    select "TEACHES_SEQ".nextval into :NEW.ID from dual;
end;

/

ALTER TRIGGER  "BI_TEACHES" ENABLE

/

```

4.8 UserDocs

```

CREATE TABLE  "USERDOC"
(
    "DOCUMENT_ID" NUMBER NOT NULL ENABLE,
    "USER_ID" NUMBER NOT NULL ENABLE,
    CONSTRAINT "USERDOC_PK" PRIMARY KEY ("DOCUMENT_ID") ENABLE,
    CONSTRAINT "USERDOC_FK" FOREIGN KEY ("DOCUMENT_ID")
        REFERENCES  "DOCUMENT" ("ID") ENABLE,
    CONSTRAINT "USERDOC_FK2" FOREIGN KEY ("USER_ID")
        REFERENCES  "USERS" ("ID") ENABLE
)

```

```

    )
/

CREATE OR REPLACE TRIGGER  "BI_USERDOC"
    before insert on "USERDOC"
    for each row
begin
    select "USERDOC_SEQ".nextval into :NEW.DOCUMENT_ID from dual;
end;

/

ALTER TRIGGER  "BI_USERDOC" ENABLE
/

```

4.9 Tag

```

CREATE TABLE  "TAG"
(
    "ID" NUMBER NOT NULL ENABLE,
    "NAME" VARCHAR2(45) NOT NULL ENABLE,
    CONSTRAINT "TAG_PK" PRIMARY KEY ("ID") ENABLE
)
/

CREATE OR REPLACE TRIGGER  "BI_TAG"
    before insert on "TAG"
    for each row
begin
    select "TAG_SEQ".nextval into :NEW.ID from dual;
end;

/

ALTER TRIGGER  "BI_TAG" ENABLE
/

```

4.10 DocTag

```

CREATE TABLE  "DOCTAG"
(
    "ID" NUMBER NOT NULL ENABLE,
    "DOCUMENT_ID" NUMBER NOT NULL ENABLE,
    "TAG_ID" NUMBER NOT NULL ENABLE,
    CONSTRAINT "DOCTAG_PK" PRIMARY KEY ("ID") ENABLE,
    CONSTRAINT "DOCTAG_FK" FOREIGN KEY ("DOCUMENT_ID")
        REFERENCES  "DOCUMENT" ("ID") ENABLE,
    CONSTRAINT "DOCTAG_FK2" FOREIGN KEY ("TAG_ID")
        REFERENCES  "TAG" ("ID") ENABLE
)

```

```

/

CREATE OR REPLACE TRIGGER "BI_DOCTAG"
  before insert on "DOCTAG"
  for each row
begin
  select "DOCTAG_SEQ".nextval into :NEW.ID from dual;
end;

/

ALTER TRIGGER "BI_DOCTAG" ENABLE

/

```

5 Views

5.1 Takes

```

select  "COURSE"."ID" as "ID",
        "COURSE"."SEMESTER" as "SEMESTER",
        "COURSE"."TITLE" as "TITLE",
        "COURSE"."COURSENUMBER" as "COURSENUMBER",
        "COURSE"."ACADEMICLEVEL" as "ACADEMICLEVEL",
        "COURSE"."SUBJECT" as "SUBJECT",
        "TAKES"."ID" as "ID",
        "TAKES"."COURSEID" as "COURSEID",
        "TAKES"."USERID" as "USERID",
        "USERS"."ID" as "ID_1",
        "USERS"."SALT" as "SALT",
        "USERS"."PW" as "PW",
        "USERS"."USERNAME" as "USERNAME"
from    "USERS" "USERS",
        "TAKES" "TAKES",
        "COURSE" "COURSE"

```

5.2 UserDocs

```

select  "USERS"."ID" as "ID",
        "USERS"."PW" as "PW",
        "USERS"."SALT" as "SALT",
        "USERS"."USERNAME" as "USERNAME",
        "USERDOC"."DOCUMENT_ID" as "DOCUMENT_ID",
        "USERDOC"."USER_ID" as "USER_ID",
        "DOCUMENT"."ID" as "ID",
        "DOCUMENT"."TYPE" as "TYPE"
from    "DOCUMENT" "DOCUMENT",

```



```

"USERDOC" "USERDOC",
"USERS" "USERS"

```

5.3 Document Tags

```

select  "DOCUMENT"."ID" as "ID",
        "DOCUMENT"."TYPE" as "TYPE",
        "DOCTAG"."ID" as "ID",
        "DOCTAG"."DOCUMENT_ID" as "DOCUMENT_ID",
        "DOCTAG"."TAG_ID" as "TAG_ID",
        "TAG"."ID" as "ID_1",
        "TAG"."NAME" as "NAME"
from    "TAG" "TAG",
        "DOCTAG" "DOCTAG",
        "DOCUMENT" "DOCUMENT"

```

5.4 Professor Documents

```

select  "PROFESSOR"."ID" as "ID",
        "DOCCOURSE"."COURSEID" as "COURSEID",
        "DOCCOURSE"."DOCID" as "DOCID",
        "COURSE"."ID" as "ID",
        "COURSE"."SEMESTER" as "SEMESTER",
        "COURSE"."TITLE" as "TITLE",
        "COURSE"."COURSENUMBER" as "COURSENUMBER",
        "COURSE"."ACADEMICLEVEL" as "ACADEMICLEVEL",
        "COURSE"."SUBJECT" as "SUBJECT",
        "DOCUMENT"."ID" as "ID_1",
        "DOCUMENT"."TYPE" as "TYPE",
        "TEACHES"."PROFID" as "PROFID",
        "TEACHES"."COURSEID" as "COURSEID",
        "PROFESSOR"."NAME" as "NAME",
        "PROFESSOR"."DEGREE" as "DEGREE"
from    "DOCCOURSE" "DOCCOURSE",
        "COURSE" "COURSE",
        "DOCUMENT" "DOCUMENT",
        "TEACHES" "TEACHES",
        "PROFESSOR" "PROFESSOR"
group by PROFESSOR.ID

```

5.5 Course Documents

```

select  "DOCCOURSE"."COURSEID" as "COURSEID",
        "DOCCOURSE"."DOCID" as "DOCID",
        "DOCUMENT"."ID" as "ID_1",
        "DOCUMENT"."TYPE" as "TYPE",

```

```

        "COURSE"."ID" as "ID",
        "COURSE"."SEMESTER" as "SEMESTER",
        "COURSE"."TITLE" as "TITLE",
        "COURSE"."COURSENUMBER" as "COURSENUMBER",
        "COURSE"."ACADEMICLEVEL" as "ACADEMICLEVEL",
        "COURSE"."SUBJECT" as "SUBJECT"
from    "COURSE" "COURSE",
        "DOCCOURSE" "DOCCOURSE",
        "DOCUMENT" "DOCUMENT"

```

5.6 User Roles

```

select  "USERS"."ID" as "ID",
        "USERS"."SALT" as "SALT",
        "USERS"."PW" as "PW",
        "USERS"."USERNAME" as "USERNAME",
        "USERROLE"."USER_ID" as "USER_ID",
        "USERROLE"."ROLE_ID" as "ROLE_ID",
        "ROLE"."ID" as "ID",
        "ROLE"."NAME" as "NAME"
from    "ROLE" "ROLE",
        "USERROLE" "USERROLE",
        "USERS" "USERS"

```

6 Spool

```

SQL>  CREATE TABLE  "USERS"
      2      (      "ID" NUMBER NOT NULL ENABLE,
      3              "SALT" VARCHAR2(45) NOT NULL ENABLE,
      4              "PW" VARCHAR2(45) NOT NULL ENABLE,
      5              "USERNAME" VARCHAR2(50) NOT NULL ENABLE,
      6              CONSTRAINT "USERS_PK" PRIMARY KEY ("ID") ENABLE
      7      )
      8  /

```

Table created.

```

SQL>
SQL>  CREATE OR REPLACE TRIGGER  "BI_USERS"
      2      before insert on "USERS"
      3      for each row
      4  begin
      5      select "USERS_SEQ".nextval into :NEW.ID from dual;
      6  end;
      7

```

```

8      /
SQL>  ALTER TRIGGER  "BI_USERS" ENABLE
2      /

```

Trigger altered.

```

SQL>
SQL>  CREATE TABLE  "ROLE"
2      (      "ID" NUMBER,
3              "NAME" VARCHAR2(45),
4              CONSTRAINT "ROLE_PK" PRIMARY KEY ("ID") ENABLE
5      )
6      /

```

Table created.

```

SQL>
SQL>  CREATE OR REPLACE TRIGGER  "BI_ROLE"
2      before insert on "ROLE"
3      for each row
4      begin
5          select "ROLE_SEQ".nextval into :NEW.ID from dual;
6      end;
7
8      /

```

```

SQL>  ALTER TRIGGER  "BI_ROLE" ENABLE
2      /

```

Trigger altered.

```

SQL>
SQL>  CREATE TABLE  "USERROLE"
2      (      "USER_ID" NUMBER NOT NULL ENABLE,
3              "ROLE_ID" NUMBER NOT NULL ENABLE,
4              CONSTRAINT "USERROLE_PK" PRIMARY KEY ("USER_ID") ENABLE,
5              CONSTRAINT "USERROLE_FK" FOREIGN KEY ("USER_ID")
6                  REFERENCES  "USERS" ("ID") ENABLE,
7              CONSTRAINT "USERROLE_FK2" FOREIGN KEY ("ROLE_ID")
8                  REFERENCES  "ROLE" ("ID") ENABLE
9      )
10     /

```

Table created.

```

SQL>
SQL> CREATE OR REPLACE TRIGGER  "BI_USERROLE"
      2      before insert on "USERROLE"
      3      for each row
      4      begin
      5          select "USERROLE_SEQ".nextval into :NEW.USER_ID from dual;
      6      end;
      7
      8      /

```

```

SQL> ALTER TRIGGER  "BI_USERROLE" ENABLE
      2      /

```

Trigger altered.

```

SQL>
SQL> CREATE TABLE  "PROFESSOR"
      2      (      "NAME" VARCHAR2(50) NOT NULL ENABLE,
      3              "ID" NUMBER(*,0) NOT NULL ENABLE,
      4              "DEGREE" VARCHAR2(45),
      5              PRIMARY KEY ("ID") ENABLE
      6      )
      7      /

```

Table created.

```

SQL>

```

```

SQL> CREATE TABLE  "COURSE"
      2      (      "SEMESTER" VARCHAR2(10) NOT NULL ENABLE,
      3              "ID" NUMBER(*,0) NOT NULL ENABLE,
      4              "TITLE" VARCHAR2(45) NOT NULL ENABLE,
      5              "COURSENUMBER" VARCHAR2(45) NOT NULL ENABLE,
      6              "ACADEMICLEVEL" NUMBER,
      7              "SUBJECT" VARCHAR2(50) NOT NULL ENABLE,
      8              PRIMARY KEY ("ID") ENABLE
      9      )
     10      /

```

Table created.

```

SQL>
SQL> CREATE TABLE  "TAKES"
      2      (      "COURSEID" NUMBER NOT NULL ENABLE,

```

```

3          "ID" NUMBER NOT NULL ENABLE,
4          "USERID" NUMBER NOT NULL ENABLE,
5          CONSTRAINT "TAKES_PK" PRIMARY KEY ("ID") ENABLE,
6          CONSTRAINT "TAKES_FK" FOREIGN KEY ("COURSEID")
7              REFERENCES "COURSE" ("ID") ENABLE,
8          CONSTRAINT "TAKES_FK2" FOREIGN KEY ("USERID")
9              REFERENCES "USERS" ("ID") ENABLE
10     )
11 /

```

Table created.

```

SQL>
SQL> CREATE OR REPLACE TRIGGER "BI_TAKES"
2     before insert on "TAKES"
3     for each row
4     begin
5         select "TAKES_SEQ".nextval into :NEW.ID from dual;
6     end;
7
8 /

```

```

SQL> ALTER TRIGGER "BI_TAKES" ENABLE
2 /

```

Trigger altered.

```

SQL>
SQL> CREATE TABLE "TEACHES"
2     (
3         "PROFID" NUMBER NOT NULL ENABLE,
4         "COURSEID" NUMBER NOT NULL ENABLE,
5         CONSTRAINT "TEACHES_FK" FOREIGN KEY ("PROFID")
6             REFERENCES "PROFESSOR" ("ID") ENABLE,
7         CONSTRAINT "TEACHES_FK2" FOREIGN KEY ("COURSEID")
8             REFERENCES "COURSE" ("ID") ENABLE
9     )
10 /

```

Table created.

```

SQL>
SQL> CREATE OR REPLACE TRIGGER "BI_TEACHES"
2     before insert on "TEACHES"
3     for each row
4     begin

```

```

5         select "TEACHES_SEQ".nextval into :NEW.ID from dual;
6     end;
7
8     /

SQL> ALTER TRIGGER "BI_TEACHES" ENABLE
2     /

Trigger altered.

SQL>
SQL> CREATE TABLE "USERDOC"
2     (
3         "DOCUMENT_ID" NUMBER NOT NULL ENABLE,
4         "USER_ID" NUMBER NOT NULL ENABLE,
5         CONSTRAINT "USERDOC_PK" PRIMARY KEY ("DOCUMENT_ID") ENABLE,
6         CONSTRAINT "USERDOC_FK" FOREIGN KEY ("DOCUMENT_ID")
7             REFERENCES "DOCUMENT" ("ID") ENABLE,
8         CONSTRAINT "USERDOC_FK2" FOREIGN KEY ("USER_ID")
9             REFERENCES "USERS" ("ID") ENABLE
10    )
11    /
12    REFERENCES "DOCUMENT" ("ID") ENABLE,
13    *

SQL>
SQL> CREATE OR REPLACE TRIGGER "BI_USERDOC"
2     before insert on "USERDOC"
3     for each row
4     begin
5         select "USERDOC_SEQ".nextval into :NEW.DOCUMENT_ID from dual;
6     end;
7
8     /
9     before insert on "USERDOC"
10    *

SQL> ALTER TRIGGER "BI_USERDOC" ENABLE
2     /
3     ALTER TRIGGER "BI_USERDOC" ENABLE
4     *

SQL>
SQL> CREATE TABLE "TAG"
2     (
3         "ID" NUMBER NOT NULL ENABLE,
4         "NAME" VARCHAR2(45) NOT NULL ENABLE,
5         CONSTRAINT "TAG_PK" PRIMARY KEY ("ID") ENABLE

```

```
5      )
6    /
```

Table created.

```
SQL>
SQL> CREATE OR REPLACE TRIGGER  "BI_TAG"
2     before insert on "TAG"
3     for each row
4     begin
5         select "TAG_SEQ".nextval into :NEW.ID from dual;
6     end;
7
8    /
```

```
SQL> ALTER TRIGGER  "BI_TAG" ENABLE
2    /
```

Trigger altered.

```
SQL>
SQL> CREATE TABLE  "DOCTAG"
2     (      "ID" NUMBER NOT NULL ENABLE,
3           "DOCUMENT_ID" NUMBER NOT NULL ENABLE,
4           "TAG_ID" NUMBER NOT NULL ENABLE,
5           CONSTRAINT "DOCTAG_PK" PRIMARY KEY ("ID") ENABLE,
6           CONSTRAINT "DOCTAG_FK" FOREIGN KEY ("DOCUMENT_ID")
7             REFERENCES  "DOCUMENT" ("ID") ENABLE,
8           CONSTRAINT "DOCTAG_FK2" FOREIGN KEY ("TAG_ID")
9             REFERENCES  "TAG" ("ID") ENABLE
10    )
11    /
           REFERENCES  "DOCUMENT" ("ID") ENABLE,
           *
```

```
SQL>
SQL> CREATE OR REPLACE TRIGGER  "BI_DOCTAG"
2     before insert on "DOCTAG"
3     for each row
4     begin
5         select "DOCTAG_SEQ".nextval into :NEW.ID from dual;
6     end;
7
8    /
    before insert on "DOCTAG"
    *
```

```
SQL> ALTER TRIGGER "BI_DOCTAG" ENABLE
      2 /
      ALTER TRIGGER "BI_DOCTAG" ENABLE
*
```

```
SQL> Spool off
```

7 SQL Queries

Select all the Users enrolled in Systems Programming.

```
select  "USERS"."ID" as "ID",
from    "USERS" "USERS",
        "TAKES" "TAKES",
        "COURSE" "COURSE"
where   "COURSE"."TITLE" = "Systems Programming"
```

7.1 Updates

1. Insert tuples to a table (both single tuple and by a query);

```
INSERT INTO COURSE
(ID, SEMESTER, TITLE, COURSENUMBER, ACADEMICLEVEL, SUBJECT)
VALUES
(300, 'Fall 2010', 'Cloud Computing', '4132', 4, 'CS');
```

1. Delete tuples from a table (both single tuple and by a query);

```
SQL> DELETE FROM USERS WHERE USERNAME = 'isaac';
```

```
1 row deleted.
```

```
SQL> Spool off
```

8 Triggers

8.1 Before User Delete

Will assign all documents of user being deleted to a permanent user account to retain referential integrity.

```
CREATE OR REPLACE TRIGGER "DELETEUSER"
BEFORE
delete on "USERS"
for each row
begin
```



```
UPDATE USERDOC
SET USER_ID=0
WHERE USER_ID=:OLD.ID;
end;
/
ALTER TRIGGER "DELETEUSER" ENABLE
/
```