Tutorial 2 — Data Definition, Security, Modelling Languages

Richard Wong
`rk2wong@edu.uwaterloo.ca`

Department of Electrical and Computer Engineering
University of Waterloo

January 22, 2018

Before we begin...

Solutions to Tutorial 1 exercises have been added to Tutorial 1 slides.

Corrections:
Exercise 1-3 E-R diagram cardinality.
Exercise 1-6 extraneous GROUP BY clause.

Exercise 2-1, Solution

Oops! Our `user` table contains passwords in plaintext! Our table looks like:

```
 CREATE TABLE user (
id int,
name varchar(100) NOT NULL,
password varchar(2000) NOT NULL,
PRIMARY KEY (id)
);
INSERT INTO user VALUES (1, 'Alice', 'abcABC123!@#');
```

How can we modify the `user` table to replace `password` with a non-nullable `hashedPassword` column, containing the result of `PASSWORD(password)` of each row?

Exercise 2-1

```
ALTER TABLE user ADD COLUMN hashedPassword varchar(100);
UPDATE TABLE user SET hashedPassword=PASSWORD(password);
ALTER TABLE user MODIFY hashedPassword varchar(100) NOT NULL;
ALTER TABLE user DROP COLUMN password;
```

Exercise 2-2

Suppose we have the following instance of the `user` table. Do the contents suggest any potential security vulnerabilities?

What could we do to improve it?

id	name	hashedPassword
1	Alice	*BEEFBEEFBEEFBEEF
2	Bob	*43F23EBECA12AD31CBA2C1BC2
3	Charlie	*BEEFBEEFBEEFBEEF
4	Donna	*43DBA275606D7A633AC28

Exercise 2-2, Solution

Alice and Charlie have the same password hash; it is very likely that they are using the same password. If somebody finds out one of their passwords and gains access to this database, they could also find out that they could use the same password to access the other’s account.

Here are three of the things we could do to make this more secure:

1. We could **use a salt**, a randomly-generated string made for each user which is combined with their plaintext password to produce a salted password hash. (saltedPassword = PASSWORD(plaintextPassword + salt), for instance)
2. We could also **restrict access to the User table** to specific people or groups, reducing the likelihood that a compromised account is able to access this information.
3. We could also **install an auditing trigger** that monitors queries made against the hashedPassword field, and write into an audit log to identify misbehaving or compromised users.

Exercise 2-3

Which of the following functions are deterministic?

```
ABS
COUNT
DATEDIFF
GETDATE
ISNULL
RAND
```

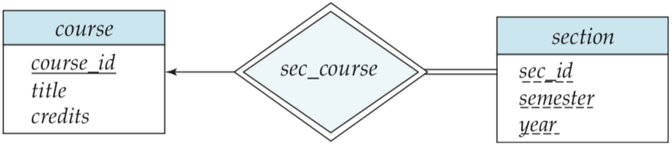
Exercise 2-3, Solution

`ABS` is deterministic.
`COUNT` is deterministic for the same input and database state.
`DATEDIFF` is deterministic.
`GETDATE` is non-deterministic.
`ISNULL` is deterministic.
`RAND` is deterministic IF you provide a constant seed and count identical output sequences given the same input to fit the definition of deterministic.

Exercise 2-4

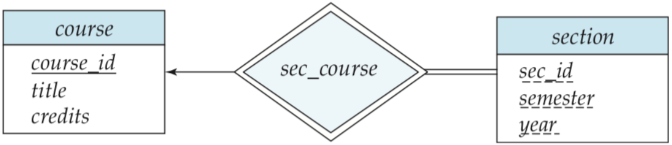
What do the following parts of this diagram mean?

- the solid underline on *course_id*
- the single arrow pointing to *course*
- the double diamond around *sec_course*
- the double lines between *sec_course* and *section*
- the dashed underlines on the attributes of *section*



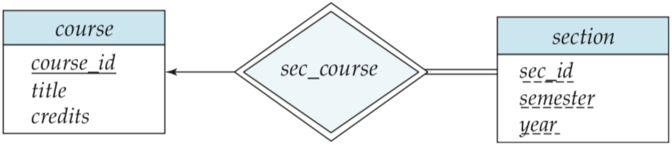
Exercise 2-4, Solution

- the solid underline on *course_id*: This is the primary key of Course.
- the single arrow pointing to *course*: Each Section is related to exactly one Course.
- the double diamond around *sec_course*: This indicates that SecCourse is a weak relationship, a relationship involving a weak entity.
- the double lines between *sec_course* and *section*: This indicates total participation of the Section in the SecCourse relationship — every Section is involved in some SecCourse. Also, no arrow indicates that many Sections can be involved in the relationship.
- the dashed underlines on the attributes of *section*: These attributes form the discriminator of the Section and are used to tell different Sections belonging to a Course apart from each other.



Exercise 2-5

What could the database schema for the following E-R diagram look like?

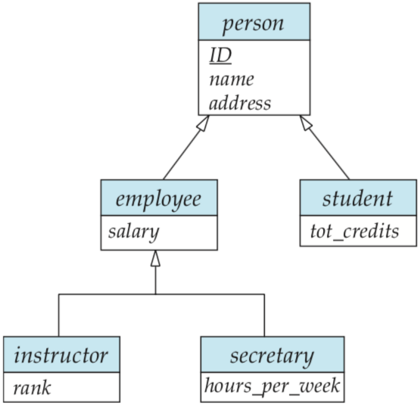


Exercise 2-5, Solution

We could give each entity (Course, Section) the attributes listed explicitly in the diagram, but also give Section foreign keys to all of Course's primary keys (in this case, there is just `course_id`).

Exercise 2-6

What could the database schema for the following E-R diagram look like?



Exercise 2-6, Solution

I know of three approaches to achieve inheritance in a relational database; don’t worry about remembering these terms unless Prof. Zarnett specifically mentions them. Approaches 1 and 2 were mentioned in lecture.

Source: https://stackoverflow.com/questions/3579079/how-can-you-represent-inheritance-in-a-database

1. Class table inheritance. Make a table for each entity, containing only the new attributes it brings to the hierarchy, and a foreign key to the parent it inherits from (or perhaps the root entity).
2. Concrete table inheritance. Each entity has a table with all of the attributes that the entity should have.
3. Single table inheritance. Combine all attributes that any entity in the inheritance hierarchy could have into one table, and add an attribute that distinguishes the type.