Data Structures, Algorithms and Databases

Solutions

Class Test #2 [2022-23]

Data Structures, Algorithms and Databases

Question 1 Entity-Relationship modelling

A database is required for a driving school. The school maintains a number of vehicles, instructors that are qualified to teach driving on particular types of vehicles (cars, buses, trucks etc.), and technicians that maintain the vehicles.

Vehicles have registration numbers and belong to particular types. Their year of purchase and the current market value must be recorded.

Students enrol for driving courses for particular types of vehicles after obtaining a learner's permit. Once they are enrolled, they are assigned an instructor, who is responsible for giving them driving lessons. The instructor is qualified to teach on a particular type of vehicle and is allocated a vehicle of that type. The course involves a standard number of driving hours (called "lessons"). The lessons are scheduled depending on instructor availability with a target of 2–3 hours per week. If the assigned instructor is not available, substitute instructors may be allocated for individual lessons.

After the students complete the required number of driving hours, they will be allowed to take a driving test. If they do not pass the test, they will be given additional driving lessons for extra fees. The students also have the option of leaving the school.

The school needs to maintain employee information for all employees (name, address, phone number etc.). For instructors, it needs to keep track of their teaching schedules, and their performance in terms of how many students they have taught over a period of time and their success rate. ("Keeping track" means that it should be possible to calculate these pieces of informaion from the stored data.)

a. Develop an **Entity-Relationship model** for the application. Explain the key design decisions made in your choice of entities, relationships and any other (ownership or hierarchical) aspects. **[4 marks]**

   Annotate it with **multiplicities** and **hierarchy** annotations, if any. **[3 marks]**

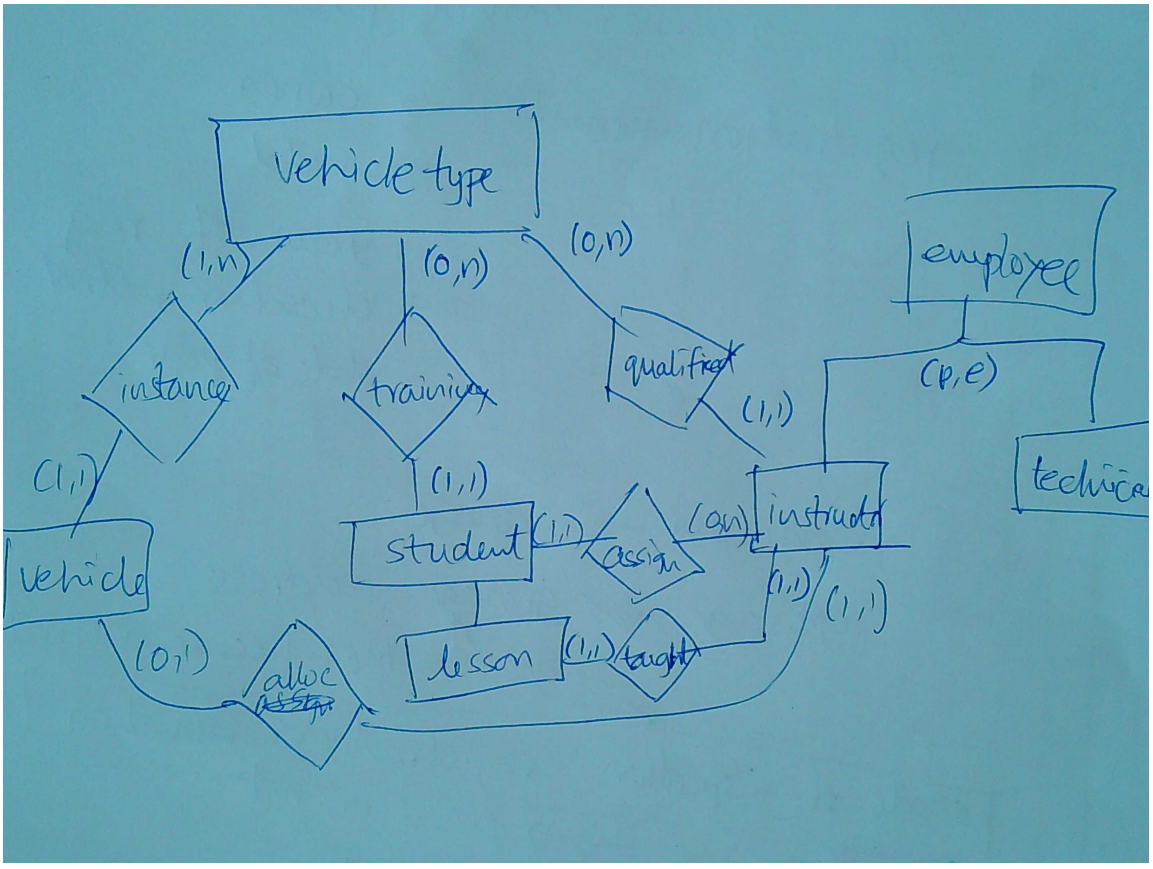
b. Carry out **logical design** for the model, representing the design with relational schemas for tables. Annotate the schemas with primary keys and the possibility of null attributes. **[4 marks]**

c. Write SQL "`CREATE TABLE`" statements for 2–3 tables. Include among them at least one table for a relationship, weak entity or subclass entity. The other table(s) may be those that are linked to this table.

   Include all the necessary constraints and "`ON DELETE`" actions. **[3 marks]**

Model answer / LOs / Creativity:

a. E-R model:



Entities: vehicle type, vehicle, employee, instructor, technician, student, lesson.

- We could have had a separate entity for courses, but there appears to be only one course, perhaps specialised to each vehicle type.
- Tests are mentioned, but the only information we need is whether (or how many times) a student might have attempted it, and whether he/she passed.
- Lesson is a weak entity, associated with the student. (Recall that a lesson is nothing but an hour of driving instruction.)

Relations:

- Vehicles are **instances** of vehicle types.
- Students are **trained for** vehicle types.
- Instructors are **qualified** to teach vehicle types.
- Students are **assigned** instructors.

- Vehicles are **allocated to** instructors.
- Students' lessons are **taught by** instructors, who may or may not be the assigned instructors. (Note that we would not need this relationship if the students were always taught by assigned instructors.)

b. Logical design:

```
vehtype(vehtype, course_hours) -- for vehicle types
vehicle(regnum, vehtype, purchase_date, value)
employee(empnum, firstname, lastname, address, phone, emptype)
instructor(empnum, vehtype, vehicle)
student(sid, firstname, lastname, permit_date, vehtype, instructor,
        num_attempts, passed)
lesson(sid, date, time, instructor)
```

There are many relationships with a maximum multiplicity of 1. They have been absorbed into the adjoining entity.

- The "instance" relationship has been absorbed into the vehicle entity.
- The "alloc" and "qualified" relationships have been absorbed into the instructor entity.
- The "training" and "assigned" relationships have been absorbed into the student entity.
- The "taught" relation is absorbed into the lesson entity.

Comments on the ER-model:

Vehicle type. This most important entity has been missed by a lot of people. All of our entities, student, instructor and vehicle are related to vehicle types. A student learns driving for a particular vehicle type; an instructor is qualified to teach a particular vehicle type. It is not possible to model this problem correctly without reference to vehicle types.

Some people tried to represent vehicles as a class hierarchy, with three subclasses of vehicle called "car", "bus" and "truck". That is intuitive. But how does that help? Don't you also need subclasses of instructor called "car instructor", "bus instructor" etc. and similarly "car student", "bus student" etc.? So, a class hierarchy is the wrong tool to use here, "wrong" because we *do not need to distinguish* between different subclasses. They are all being treated the same way.

Course. There is absolutely no need for a course entity in this problem. There is only one course, viz., the "driving course". It is of course specialised to vehicle types. So, if you do include a course entity, you would notice the relationship:

```
vehtype --- (1,1) --- requires --- (1,1) --- course
```

and the multiplicities indicate that all these three can be merged into a single table.

People that included a course entity in their design tended to view it like a University course, with a single instructor and multiple students. Some even attached "lesson" as a weak entity to the course! That is completely wrong, because driving lessons are one-on-one driving trips, individually taught by an instructor for a single student.

Lesson. A lesson is simply an hour of driving done by a student in the presence of an instructor. So, a lesson clearly belongs to the student, as a weak entity.

Test. Tests are not conducted by the school. Usually some other testing agency conducts tests. Tests have nothing to do with driving lessons either. (People could take driving tests if they knew driving somehow.) So tests do not need to be treated as an entity for the school. All that matters to the school is whether a student took the test and, if so, what the result was. If you want to keep data about tests you would want to make it a weak entity dependent on student (just like lesson). In the model solution, we are only keeping track of the number of attempts a student might have made for the test (because it affects the performance record of the instructor!).

Technician. Nothing has been said about the technicians in the problem description. They "maintain vehicles" as a general description. This doesn't give rise to a relationship called "maintains" between technicians and vehicles. What would you store in such a table? A technician might be able to maintain all vehicles, as far as we know.

Instructor information. The school needs to keep track of the teaching schedules, how many students they taught and the success rate. All of these can be calculated using the data we have. Lessons have dates and times, and the instructor who is scheduled to teach them. So, we obtain teaching schedules. We can count how many students have been assigned to an instructor. And we can also count how many of them passed their tests and how many attempts they needed to pass. None of these things can be directly stored with the instructor. They are not data the school can enter into the system. Rather it is the job of the system to figure them out.

c. CREATE TABLE statements:

```
create table vehtype(
    vehtype VARCHAR(10) not null unique,
    course_hours INTEGER not null,
    primary key (vehtype)
)
create table vehicle(
    regnum VARCHAR(6) not null unique,
    vehtype VARCHAR(15) not null REFERENCES vehtype(vehtype),
    purchase_date DATE not nll,
    value INTEGER not null,
    primary key (regnum)
)
create table employee(
    empnum INTEGER not null unique,
    firstname VARCHAR(15) not null,
    lastname VARCHAR(15) not null
    address TEXT not null,
    phone   DECIMAL(11) not null,
    emptype VARCHAR(10), -- whether instructor, technician or other
    primary key (empnum)
)
create table instructor (
    empnum INTEGER not null REFERENCES employee(empnum)
                            ON DELETE CASCADE,
    vehtype INTEGER not null REFERENCES vehtype(vehtype),
    vehicle VARCHAR(6) not null REFERENCES vehicle(regnum),
    primary key (empnum)
)
create table student(
    sid INTEGER not null unique,
    firstname VARCHAR(15) not null,
    lastname VARCHAR(15) not null,
    permit_date DATE not null,
    vehtype INTEGER not null REFERENCES vehtype(vehtype),
    instructor INTEGER not null REFERENCES instructor(empnum),
    numattempts INTEGER not null,
    passed BOOLEAN not null,
    primary key (sid)
)
create table lesson(
    sid INTEGER not null REFERENCES student(sid)
```

```
                        ON DELETE CASCADE,
    date DATE not null,
    time TIME not null,
    instructor INTEGER not null REFERENCES instructor(empnum),
    primary key (sid, date, time)
)
```

ON DELETE CASCADE is specified for two cases: a weak entity and a subclass entity. In all other cases ON DELETE NO ACTION applies (by default).

Question 2 Relational Algebra

a. Given below are two tables T_1 and T_2.

T_1 :

A	W	X	Y	B
a	1	*x*	5	*e*
b	2	*x*	7	*f*
b	3	*x*	5	*g*
a	4	*y*	5	*h*

T_2 :

D	X	W	Y
p	*x*	1	7
t	*y*	2	7
p	*y*	3	5
r	*x*	4	7

Show the table obtained by evaluating the following expression: **[3 marks]**

$$(\pi_{(A,X,Y,B)} T_1) \bowtie T_2$$

b. Given below are the schemas for a database representing staff members, courses, and a relationship of staff members lecturing courses in particular years.

```
staff(sid, title, firstname, lastname, email, office, phone)
courses(cid, level, name, credits)
lecturing(cid, sid, year)
```

We want to find the courses that were taught by the same staff member during the year 2021 as well as 2022. Write a relational algebra expression for this purpose.

(Hint: Consider calculating the cid's of such courses first.) **[3 marks]**

Model answer / LOs / Creativity:

a. The table $\pi_{(A,X,Y,B)}(T_1)$ has columns (A, X, Y, B) and T_2 has columns (D, X, W, Y). The commoon columns are X and Y.

Matching the records on these two columns gives the following join.

A	X	Y	B	D	W
b	x	7	f	p	1
b	x	7	f	r	4
a	y	5	h	p	3

b. **Method 1: Using projection**. The following join finds the sid, cid combinations that matched in years 2021 and 2022:

$$R = \pi_{\texttt{cid,sid}}(\sigma_{\texttt{year}=2021}(\texttt{lecturing})) \bowtie \pi_{\texttt{cid,sid}}(\sigma_{\texttt{year}=2022}(\texttt{lecturing}))$$

(After the projection, only the `cid` and `sid` attributes are left. They will be matched in the join.)

The names of the courses can be found using:

$$\pi_{\texttt{name}}(R \bowtie \texttt{courses})$$

Method 2: Using renaming. A variant of the table R can also be calculated using renaming as follows:

$$R' = \sigma_{\texttt{year1}=2021 \wedge \texttt{year2}=2022}(\rho_{(\texttt{cid,sid,year1})}(\texttt{lecturing}) \bowtie \rho_{(\texttt{cid,sid,year2})}(\texttt{lecturing}))$$

(We retain `cid` and `sid` fields as is, but rename the `year` field. This is less efficient than Method 1 because the selection operation comes late, after join.)

The names of the courses can be found the same way as Method 1:

$$\pi_{\texttt{name}}(R' \bowtie \texttt{courses})$$