Databases — Database Design

Gilles Peiffer (24321600), Thomas Ponchau (03271500), Liliya Semerikova (64811600)

***Abstract*—This paper explains the design choices made for the second project about database design. It also addresses the comments made by reviewers on the conceptual diagrams submitted for the first part of the project. Finally, we show how we converted the conceptual diagram into a physical schema for a database.**

Introduction

The first section addresses the design choices made by our team, while the second section addresses the various comments that were made by our reviewers. The third section shows the transformation from the conceptual diagrams to the physical schema of the database.

I. Design Choices

The following section explains the various design choices.

A. Request State

In order to distinguish between a request which has been accepted, declined, or not answered yet, each request has a "State" attribute.

B. Coordinates as Attribute

The starting and ending locations of a request are indicated as attributes, so customers can indicate where they want to pick up and where they want to leave the vehicles they rent.

C. Representing Friendships

Friendship between users of the application is indicated as an $M : N$ relationship with optional participation.

D. Allowing Carpooling

Carpooling is represented using the "WANTS_TO_JOIN" relationship.

E. Time as Attribute

The time is indicated by giving a starting time and an ending time. Additionally, a derived attribute "Duration" can also be computed from the former two.

F. Price as Derived Attribute

In order to implement the pricing scheme, a derived attribute was added to the "REQUESTS" relationship. This derived attribute can be computed based on the starting time and duration of any given rental slot, but can also include other factors such as the type of the car.

G. Vehicle Type as Attribute

Some diagrams use hierarchies to distinguish between different types of vehicles, whereas our diagram uses a simple attribute. This hierarchy makes sense when one considers that different types of vehicles may use different types of fuels, for example, but since the fuel type is not an attribute mentioned in the project statement, our team preferred to avoid complicating the diagram further by adding a hierarchy.

H. "RENTAL_SLOT" as Weak Entity

The "RENTAL_SLOT" entity groups together many attributes which are relevant for a given request. Without a request to bind it to, there would be no rental slot. For this reason, it is indicated as being a weak entity.

I. Mandatory Ownership for Vehicles

As indicated on the diagram, each vehicle must have an owner, but not every customer must be the owner of a vehicle. This makes sense in the real world.

J. Removed Hierarchy

Most reviewers agreed that the hierarchy was too complicated, and that the role of a customer could adequately be understood using only the various relationships.

II. Addressing Comments

This section addresses the various comments made by reviewers.

A. Reviewer 1

1) Aspects 1 & 2

The requirement here is simply that a weak entity be related to at least one weak relationship, which is satisfied.

2) Aspect 8

It is already possible, by setting the "State" attribute to "moving".

3) Aspect 9

A rental slot has both a set of starting coordinates and a set of final coordinates, meaning the customer can specify where they will pick the vehicle up and where they will leave it. Additionally, a vehicle has coordinates for when it is stationary.

Favourite vehicles are not present in the diagram, but since the question asked whether it was easy to determine, we don't think this is a valid reason to deduct points either.

The "Price" attribute is a derived attribute, hence it can be computed from the time and duration of the rental slot. Additionally, other factors like the vehicle type can also be taken into account.

4) Aspect 10

As mentioned before, the weak entity is actually indicated correctly.

The remark on the usefulness of the hierarchy was taken into account, and the latter has been removed from the updated diagrams.

Subattributes are as essential as "regular" attributes, and are indicated in the diagrams of [1] as well.

5) Aspect 11

As said before, there is no valid reason to not indicate subattributes.

6) Aspect 12

This apparent inconsistency is due to the inability to have higher-order relationships on ER/Crow's Foot diagrams. The functional dependencies for the ER/Chen diagram are (in the diagram without the hierarchy)

$$\begin{aligned} \text{RENTAL\_SLOT}, \text{VEHICLE} &\rightarrow \text{CUSTOMER}, \\ \text{RENTAL\_SLOT}, \text{CUSTOMER} &\rightarrow \text{VEHICLE}, \end{aligned}$$

whereas for the ER/Crow's Foot diagram, they are

$$\begin{aligned} \text{REQUEST} &\rightarrow \text{CUSTOMER}, \\ \text{REQUEST} &\rightarrow \text{VEHICLE}, \\ \text{REQUEST} &\rightarrow \text{RENTAL\_SLOT}. \end{aligned}$$

By thinking about these FDs, one can see clearly that both make sense and represent the same model.

B. Reviewer 2

1) Aspect 6

While it is true that the hierarchy was deemed superfluous upon revision, it does not fit this specific description. Abstractly speaking, it is strictly similar to distinguishing between a manager and employee, as it is done in the slides detailing enhanced ER diagrams.

2) Aspect 8
As with the first reviewer, the "State" attribute can be used to indicate a vehicle is moving.

3) Aspect 9
Again, this comment brings up the same design choices as the first reviewer, and labels them as "mistakes", while they are in fact just choices. The diagram clearly shows what is possible and what is not, and should hence be judged based on its readability and not its completeness (as mentioned in the question: "[...] Rate how easy you found it to read these additional choices from the diagram [...]").

4) Aspect 10
The grade indicates something is missing, but the comment only mentions positive aspects.

5) Aspect 11
The hierarchy has been removed, in accordance with this comment.

C. Reviewer 3
Not a single comment was made in this evaluation.

III. From Conceptual Diagram to Physical Schema
This section shows the conversion between the conceptual and the physical designs.

A. Step 1: Converting Entities
This step is quite trivial: for every (non weak) entity in the conceptual diagram, a relation is added to the database, with the attributes as columns.

B. Step 2: Converting Weak Entities
The "RENTAL_SLOT" entity is weak, hence to convert it, a new relation was added to the database, which includes all the attributes but also contains foreign keys to link this entity with the "VEHICLE" and "CUSTOMER" entities. As one can see, the primary key for "RENTAL_SLOT" is a triple $(\text{Renter}, \text{VehicleId}, \text{Start\_time})$.

C. Step 3: Converting $M : N$ relationships
For each such relationship, a new table is added to the database, with foreign keys to identify the entities (in our case, either the id's of customers which are friends or the keys of a request and a customer who wishes to join it). One can also use this approach for $1 : N$ relationships.

D. Verifying Requirements
1) Keys Are Properly Defined
The physical schema has been obtained using the traditional procedure, hence all keys are properly defined. Where necessary, `UNIQUE` constraints are used.

2) Sensible Types
The only NULLable fields are the location of a vehicle, as a moving vehicle cannot have a location. For this, a special constraint was included in the SQL queries.

Where possible, adapted data types were used: for example, latitude and longitude are represented using `DECIMAL(10, 8)` and `DECIMAL(11, 8)`, respectively; times are represented using the `DATETIME` type.

3) Third Normal Form
Third normal form requires that for every nontrivial full functional dependency $X \to A$, either X is a primary key or A is a prime attribute. This is the case, as every such functional dependency satisfies the first condition, i.e., has a primary key in its left-hand side.

4) Database Reflects Conceptual Design
Since our team used the traditional procedure outline in [1] to construct the physical schema, the schema necessarily matches the conceptual design closely.

Conclusion
Many design choices need to be made in order to model a database, and there is no single "right" answer. Through making these diagrams and schemas, we were able to better grasp the various difficulties of such a task.

Reviewing other people's work was a great way to get better at understanding a diagram made by someone else, but also allowed us to consider new perspectives that hadn't been considered before. Similarly, using the feedback from our reviewers, we were able to simplify and improve our diagram.

References
[1] Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems (7th Edition)*, 2015.