COMP9311 16s2 Database Systems

Assignment 1 Stage 2

Last updated: **Thursday 4th August 10:02pm**
Most recent changes are shown in red ... older changes are shown in brown.

Introduction

This document contains the completed ER design for Stage 2 of Assignment 1. You must convert this design into a PostgreSQL relational schema (i.e. a collection of `create table` statements) and submit it via the give command. When converting from the ER design to a relational schema, you should follow the approach given in the lecture notes on "ER to Relational Mapping". You may need to investigate and/or derive mappings for the constructs that have not been discussed in the lecture.

Submission

Submission : Login to a CSE Linux machine such as `wagner` and use the give command below to submit the assignment (note that the give command does not work on `grieg`):

```
give cs9311 a1 a1.sql
```

Deadline : Sunday 21st August 23:59

Late Penalty: Late submissions will attract a 10% penalty for the first day and a 30% penalty for each subsequent day.

Notes: For fairness to all students in the class, no special considerations will be given to those:

- who claim to have submitted their assignment but the assignment has not been received by the give system above; (you can use the "classrun" command to check if your assignment has been submitted, e.g., "9311 classrun -check a1");
- who have submitted their assignment a few minutes late and request to be considered as non-late submissions; (please submit your assignment early in case of your network connection problem, computer breakdown, etc);
- who claimed that their assignments worked perfectly on their home computers but somehow did not work on the CSE linux machines; (we will only test and mark your assignments on the CSE linux machines, and will not consider the results on your own machines. So please test your assignments on CSE linux machines before submission).

Requirements on your Submission

The schema that you submit will be marked by a program (auto-marked). In order for the program to recognise what you've done as being correct, your SQL **must adhere to the following requirements**:

- all tables must have an appropriate primary key defined; all foreign keys must be identified
- use appropriate domains for each attribute (e.g. date should be defined using an SQL `date`; a counter would be done as an SQL `integer` constrained to be ≥ 0)
- if an attribute is a string, and no maximum length is specified, use PostgreSQL's (non-standard) `text` type; otherwise, use an appropriate type such as `varchar(N)` type or one of the supplied domain types; if appropriate, you can also create new domain types.
- if an attribute is a boolean value, use the SQL `boolean` type
- wherever possible, not-null, unique and domain constraints must be used to enforce constraints implied by the ER design
- derived (computed) attributes should not be included in the SQL schema
- wherever possible, participation constraints should be implemented using the appropriate SQL constructs
- map all of the entity class hierarchies in the ER design using the **ER-style** mapping (i.e. one table for each entity class).
- all relationships should be mapped using the approaches described in the lecture notes; in particular, you should avoid "over-generalising" your SQL schema relative to the ER design (e.g. a 1:n relationship should *not* be mapped in such a way that it can actually be used to form an n:m relationship)

Since the assignment is going to be auto-marked, you must use the names that the auto-marker expects (i.e., the same names that are used in the provided ER design. Please also follow the following naming conventions:

- when mapping multi-valued attributes, the new table's name is the concatenation of the entity and attribute names
- when mapping composite attributes, use the names of the "leaf" attributes
- if names in the ER diagram contain multiple words (just in case, though I believe that I have concatenated them already), concatenate them into a single word in `CamelCase` in the SQL schema.

Note: if the name you want to use clashes with a PostgreSQL keyword (e.g. `user`), you will need to write the name in double-quotes (i.e. `"user"`) and in all lower-case.

Place the schema in a file called `a1.sql` and submit it via the `give` command (see above) before the deadline. To give you a head-start, a template for the schema is available, which has (parts of) some of the required tables already defined. Note that you will need to add more tables, as well as filling out the attributes in the supplied tables. Your submission must follow this format, so save a copy of this and edit it to produce your own `a1.sql` file.

The reason for insisting on strict conformance to the above is that your submission will be auto-marked as follows:

- we will create an initially empty database (no tables, etc.)
- we will load your schema into this database
- we will use a script program to compare your schema with the expected schema

The comparison will make use of the meta-data which has been added to the database by loading your schema. Needless to say, if your schema has load-time errors, then it is not going to be possible to compare it against the correct version. Therefore, it is essential that you check that your schema can load into an *initially empty* database before you submit it.

Following the instructions above is considered to be a requirement of this assignment. If you stray from the expected schema, your submission will be marked as incorrect. Our auto-checking scripts have a little flexibility, but not much, so don't rely on it. Manual checking is to examine specific implementations that are difficult to be auto-checked, and it is not an alternative to override the results from auto-marking.

Please don't try to second-guess or "improve" the given ER design below. Even if you think it can be further improved, just translate it as given. We should have discussed most of the issues in the Aug 14 lecture. If you still think that it's incorrect or that the information supplied isn't enough to do the mapping unambiguously, either send me an email or post a message on the course MessageBoard (topic: "Assignment 1").

The ER Design

This ER design gives one possible data model for the `KensingtonCars` application introduced in the first stage of this assignment. The design presented here is based on my draft design and the discussions during the lecture in Week 3, and on my interpretation of the more ambiguous aspects of the requirements.

To make the presentation clearer, the design is broken into a number of sections.

Other notational conventions in the ER diagrams:

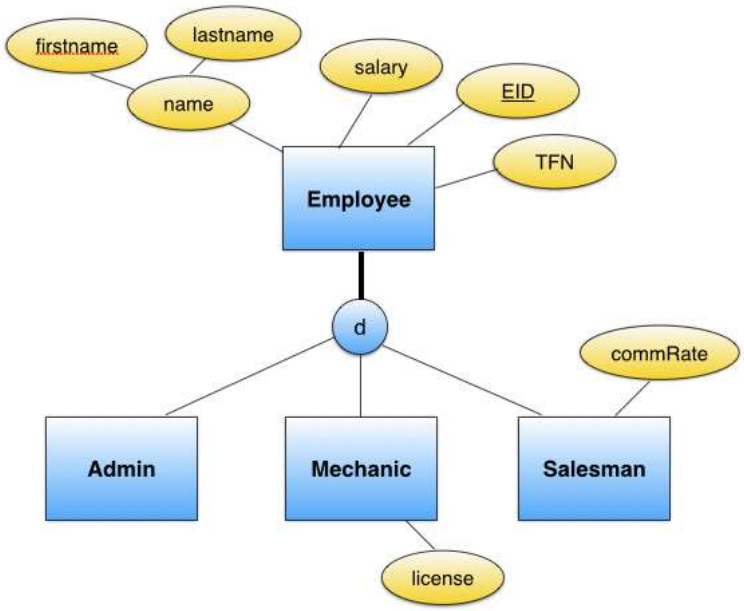
- primary key attributes for entities are underlined
- total participation in a relationship is indicated by a thick line
- an arrow indicates that at most 1 entity is involved in the relationship

Data Types

To make things easier, I've defined some useful data types using the `create domain` statement. Some of the `create domain` statements use standard SQL patterns for specifying constraints, while others use PostgreSQL-specific regular expressions for this purpose. The domain definitions are given at the top of the template file.

Employee

The following diagram shows the entities, attributes and relationships that provide the information about the employee entity for KensingtonCars.

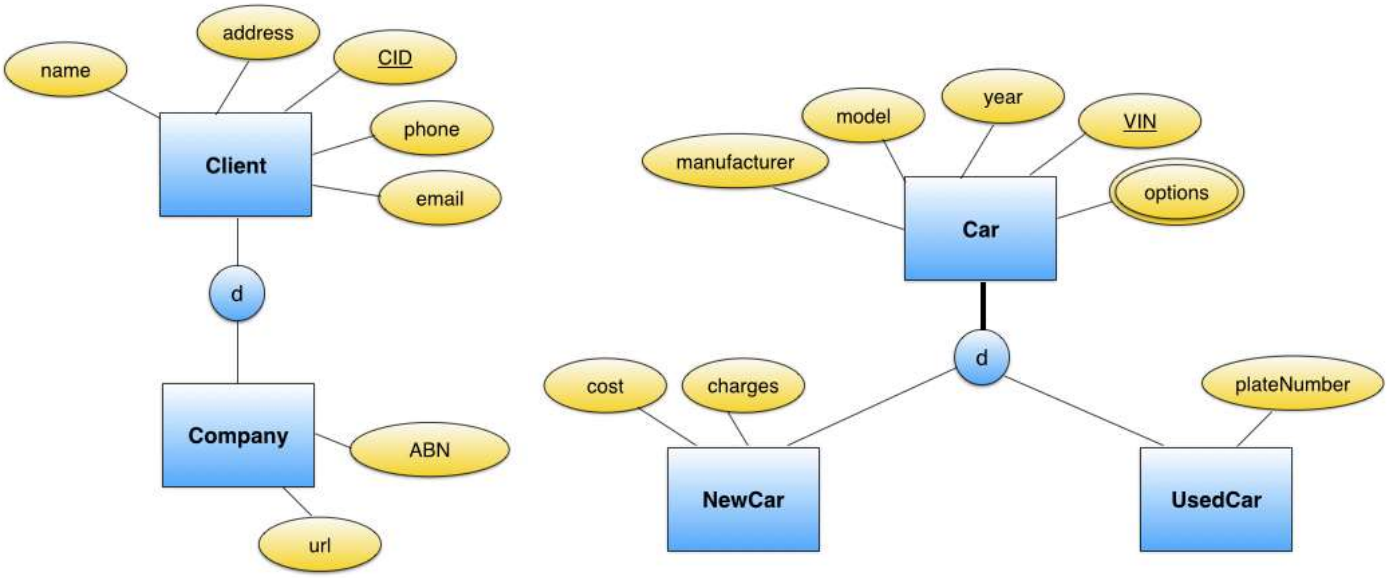


Details:

- we use a system-generated, numeric EID as a primary key, since Salesman and Mechanic will be extensively referenced in the database.
- the database should store every employee's name (consists of first name and last name), TFN and salary.
- both the salary and commission rate are integers, and must be larger than 0. Also refer to the assignment spec for further constraints on the commission rate (i.e., Each salesman will have a negotiated commission rate ranging from 5% to max 20%).
- we assume that first name and last name are no longer than 50 characters respectively.
- TFN has exactly 9 digits.
- the mechanic's license is alphanumeric, with exactly 8 characters.
- all attributes in the above ER diagram cannot have a null value.

Client and Car

The following diagram shows entitites, attributes and relationships for clients and cars.



Details (Car):

- We assume that VIN is exactly 17-character long and does not include the letters I (i), O (o), or Q (q) (to avoid confusion with numerals 1 and 0)
- Year is between 1970-2099 inclusive.
- Model and Manufacturer are of maximum 40 characters each.
- Car license is of maximum 6 alphanumeric characters.
- Except for the list of options, all other information as specified by their corresponding attributes are compulsory.
- Use the defined domain OptionType for a list of possible options.

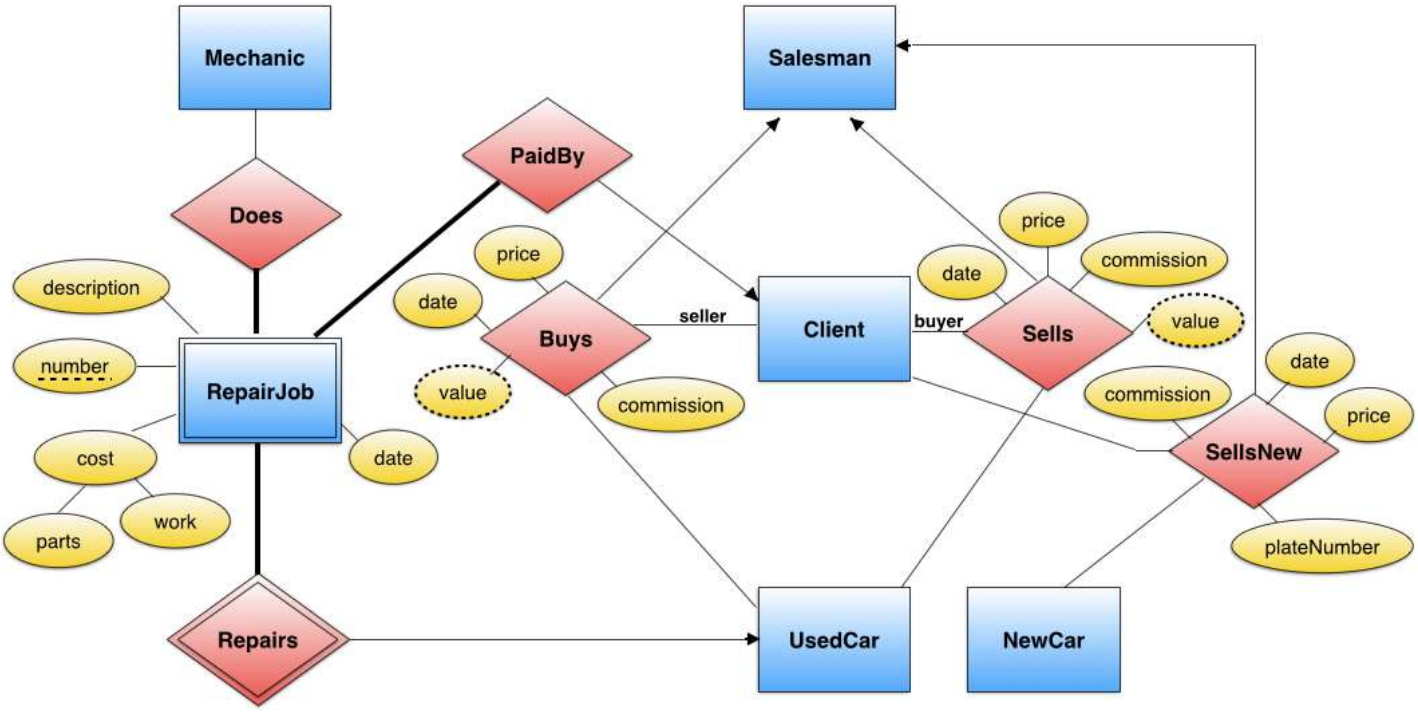
Details (Client):

- we assume that name is no longer than 100 characters.
- we use a system-generated, numeric CID as a primary key.
- address is maximum 200 characters.
- Phone number has exactly 10 digits (mobile or landline with area code).

- if a client does not have an ABN (i.e. it is not a company), all information is compulsory except the email address. The database should check if the input email is of a proper email format (using the defined EmailType domain).
- ABN has exactly 11 digits.
- If an ABN is provided (i.e., it is a company), an optional Web address of the company is stored. A URL should start with http://...

Buy, Sell and Repair

The following diagram shows entities, attributes and relationships for the rest of the design for KensingtonCars.



Details:

- Any monetary amounts should be defined using the type `numeric` with 2 decimal digits and they should all be positive numbers. All monetary amounts will not exceed 6 integral digits.
- Description has a maximum of 250 characters.
- RepairJob Number is a number between 1 and 999 inclusive.
- Phone number has exactly 10 digits (mobile or landline with area code).
- As specified in the requirements, the clients involved in a car transaction are the owners of the car.

What To Do Now

Make sure you read the above description thoroughly, and review the notes and exercises on ER-to-relational mapping. Get a copy of the a1.sql template and see what is provided there. If any aspect of this design requires further clarification, ask a question under topic "Assignment 1" on the course MessageBoard.

Reminder: before you submit, ensure that your schema will load without error if used as follows on `grieg`:

```
% dropdb a1
% createdb a1
% psql a1 -f a1.sql
... will produce notices, but should have no errors ...
% psql a1
... can start using the complete database ...
```

Penalty: If I have to fix errors in your schema before it will load, you will incur a 10 (out of total 25) mark "penalty".

Plagiarism

The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted. The penalties for such an offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined both automatically and manually for such submissions.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct.

Do not provide or show your assignment work to any other person - apart from the teaching staff of this subject. If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted you may be penalized, even if the work was submitted without your knowledge or consent. This may apply even if your work is submitted by a third party unknown to you.