

Comp2400/Comp6240 Relational Databases Assignment One Due 5pm 24 August 2007

MARKING GUIDE

Greg O'Keefe

September 17, 2007

There are basically 3 things to look at for the Comp2400 students

1. model
2. code
3. doco

For the Comp6240 students, we also need to consider

4. method explanation, and suggestions

Each of the following sections covers one of these points, suggesting things to look for and how to treat them. Only work in increments of .5 of a mark, any less is petty time-wasting.

1 Model

4 marks (3 for post-grads)

The 4 marks are for the following points. I've phrased them positively, failure to satisfy a point should generally cost half a mark. I've indicated (1) where this is different, but use your discretion.

1. generally being a clear simple complete model and scenario
 - (a) you can see how its supposed to work within 30 seconds of seeing it
 - (b) a good solution won't have more than 4 or 5 classes

- (c) absence of pointless and confusing elements
For example, there is no need for a Customer class, or product prices. Employee is borderline.
Some students read too much into my hint and have a class Timings. Feed them to the sharks. (OK, take a mark off)
- (d) plausible understandable scenario with several products, an order for several products per order, several orders

2. class diagram specifics

- (a) recognisable as a UML class diagram (1)
- (b) the only non-uml notation we will tolerate is underlining of primary keys (you can't just make stuff up)
- (c) class names are nouns with clear interpretation in described business
- (d) association classes must have names too
- (e) attributes really belong with their class, and have clear meaning
- (f) classes should not have operations (unless they have made sense of it somehow)
- (g) associations should have a name, or at least one end-name
- (h) multiplicity marked at both ends, seems correct for the assoc
- (i) showing attrib for artificial primary key is optional, as is underlining it, but should be consistent for all classes
- (j) should *not* show attributes that are also association ends (ie foreign keys) (at least ones which aren't there for some other reason)

3. object diagram/scenario specifics

- (a) name:class or :class heading for all objects, where class is in the class diagram
- (b) name is an attribute of that class for any name = value pairs in it
- (c) any class attributes not shown in the object can sensibly be NULL
- (d) links should be labelled with names or end names
- (e) links should *not* be labelled with multiplicities

4. documentation specifics

- (a) the text should say what the class and object diagrams are, and the role they play in the process
- (b) any model elements whose meaning is not completely obvious has a sentence or two defining it

2 Code

4 marks (3 for post-grads)

1. translation (1)
 - (a) there is a statement of how the schema was obtained from the class diagram, and the schema matches this
2. solution (2)
 - (a) it solves the 4 system requirements (give them 2)
 - (b) it looks like it would run and be helpful with small modifications (give them 1/2)
3. niceness (1)
 - the language/dbms features used to solve the problem are sensible and appropriate
 - the solution is about as simple as it could be
 - the right code is in the right file
in particular `queries.sql` can be run daily, it either doesn't make a mess, or it cleans up after itself

3 Doco

2 marks

Imagine you pick the report up from your in-tray, which contains maintenance schedules for fork-lift trucks, memos about EEO policy, invoices from printing companies, ...

1. You can quickly work out what this document is about.
2. You can quickly understand how it will solve the problem, and what all the deliverables are for.
3. There are helpful explanations of the details of the model and the code.
4. Unclear parts of the request for proposals (the assignment) have been resolved by explicit assumptions.

4 Language Selection and Suggestions

2 marks for post-grads (1 bonus mark for undergrads)

A statement justifying the choice of UML language features to use. This should mention things like

1. ability to naturally express the problem

2. simplicity of translation

A statement justifying the choice of UML to schema translation procedure. The should mention things like

1. simplicity of procedure, for accuracy or automation
2. directness of translation allowing the model to serve as documentation
3. simplicity of resulting schema
4. space efficiency or performance of the resulting schema

Some suggested suggestions

1. automated stock reordering based on delivery lead times and demand
2. treating packing materials as a product, ordered *implicitly*
3. calculating the number of casual staff to call in
4. collecting historical data and using it to improve estimates of work, forecasts of product requirements
5. generating estimate code from a model of the order filling process (!)

Give them a mark if there are at least 2 clearly stated plausible helpful sounding suggestions.

I don't want to sub-allocate marks here: a really good treatment of either part along with an OK effort at the other should get full marks.