Entity Relationship Diagram

PB007 Software Engineering I

Bruno Rossi

14. 11. 2016

Entity-Relationship Diagram

The **Entity-Relationship Diagram (ERD)** is a data model that represents the logical structure of the database.

The essential components are:

- *Entities*
- *References/Relations*
- *Attributes*

Class Diagram vs. ERD

Class Diagram

- model data operations, etc...
- classes are interconnected by different types of relationships (associations, dependencies, inheritance, aggregation, composition)
- more often represent real-world entities

Entity-Relationship Model

- models only data
- represents the tables in the database

Object-oriented paradigm to manage data by means of the interaction of objects; *Relational paradigm* to access data through relational algebra (SQL)

Object-Relational Mapping

Object-Relational Mapping (ORM) is an approach that allows the conversion of data between relational databases and object-oriented languages.

- a persistent class defines an entity (mapped as a table)
- an object defines an entity (a row in the table)
- attributes of the class are becoming entity attributes (columns of the table)
- association / aggregation / composition between classes defined in the schema (foreign keys linking the tables)
- inheritance can be solved in several ways: mapping 1:1, inclusion in the superclass, creation of all subclasses
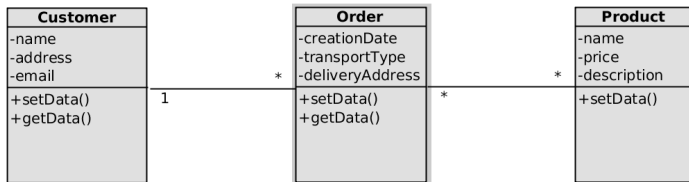
Note:

- one class can be mapped to multiple tables
- more classes can be mapped to one table
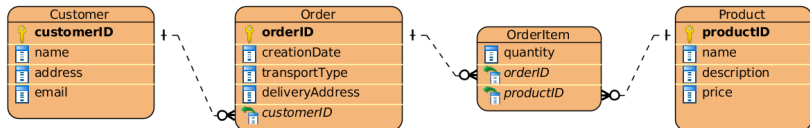- not all classes have to be persistent

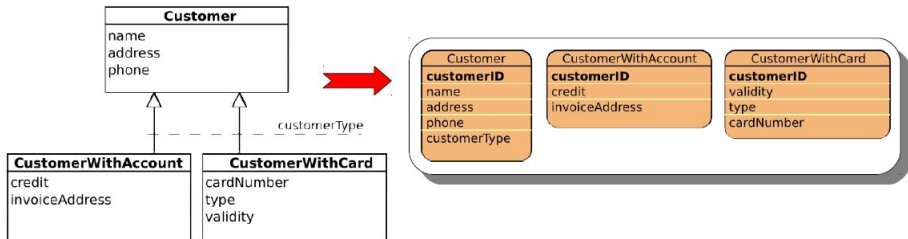# Object-Relational Mapping II
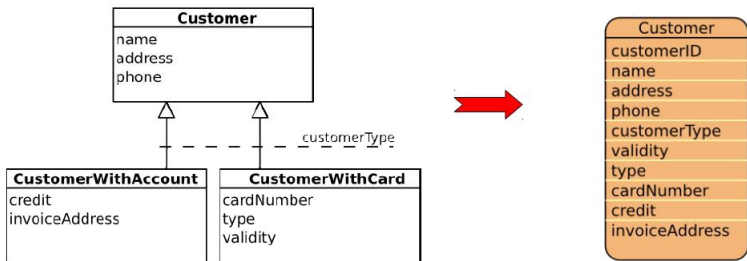
Class Diagram:

ERD:

ORM - Inheritance - mapping 1:1

- Each class becomes a table.
- ID is the discriminating attribute.
- One instance of the class is stored in multiple tables - complex data access.
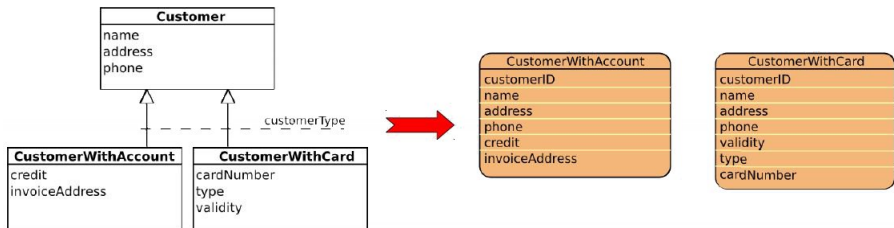
ORM - Inheritance - inclusion in the superclass

- All the attributes of the subclasses are included in one table.
- Some attributes may contain NULL - break of 4.NF.
- Generally good to be adopted with a small number of subclasses with few attributes.

- Attributes from the superclass are distributed in all the tables of the subclasses
- Generally suitable if:
  - superclass has few attributes
  - there are many subclasses
  - subclasses have many attributes

Normal Form

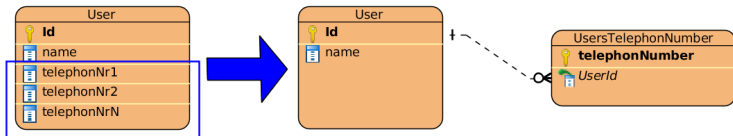**Normal Forms** are used to reach a good database design.

Note:

- eliminate repetitive data
- reduce complex tables
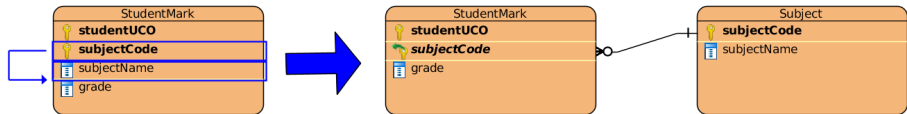- prevent anomalies (for update, insert, delete)

1. Normal Form

The **first normal form** requires that each attribute contains only atomic values, i.e. values that are further indivisible.

2. Normal Form

The **second normal form** requires that the schema is in 1.NF
and that each non-key attribute is fully dependent on the whole primary key.
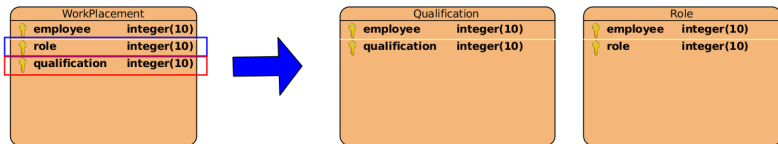
3. Normal Form

The schema is in **third normal form** if it is in 2.NF and all attributes not part of the candidate keys are directly dependent (i.e. non-transitively) upon every candidate key. This means that all non-key attributes must be independent of each other.

4. Normal Form

The schema is in **fourth normal form** if it is in 3.NF and there are no multi-valued dependencies.

Does not allow to capture qualified employees that do not have assigned a role

Additional Resources

- http://www.cs.toronto.edu/~sme/CSC340F/2005/slides/
  tutorial-classes_ERDs.pdf
- http://www.bkent.net/Doc/simple5.htm

Tasks

- Complete the state diagram based on the feedback.
- Based on the analysis class diagram create the initial ERD (virtually identical to the class diagram + eliminate inheritance), which will be further reworked.
- Unfold M: N relationships using entities.
- Identify the keys for all the entities, by trying for maximize efficiency (ie., do not create artificial id where they are not required like in related entities to identify the keys of the original entities).
- Is the model in 3. NF? If yes, which variation would put it in 2. NF? Show just such a change and the implications.
- Upload the **PDF report** into folder (**Week 08**).
  Deadline: Saturday, 19.11.16 23:59

Customization of PDF Report