SEG4210 - Advanced Software Design and Reengineering

Topic 17
ORM: Object Relational Mapping

Rules for Mapping an Object Model to a Relational Database 1

General Rules

A. Create one table for each ordinary class

1. Make each instance a row in the table
2. Make each simple attribute or qualifier into a column
3. If a complex attribute is naturally composed of separate fields, make these into individual columns
4. If a complex attribute consumes much space but is infrequently accessed, store it in a separate table.
5. Make 1-1 or many-1 association into either
   - i. a column (using a foreign key)
   - ii. a separate table with foreign keys indicating each end of the relation

Object-Relational Mapping Rules 2

General rules cont …

A. cont …

6. Decide on the key:

- The key must uniquely distinguish each row in the table

i. If unique, one or more of the attributes may constitute the key

ii. Otherwise, generate a column that will contain a special object-id as the key

Object-Relational Mapping Rules 3

General rules cont …

B. Create one table for each association that has many-many multiplicity (and for each association class)

1. Make the keys of the tables being related to be foreign keys in this new table
2. If we are creating an association class with additional associations connecting to it, then create a special object-id as the key
3. Follow rules A1 to A5

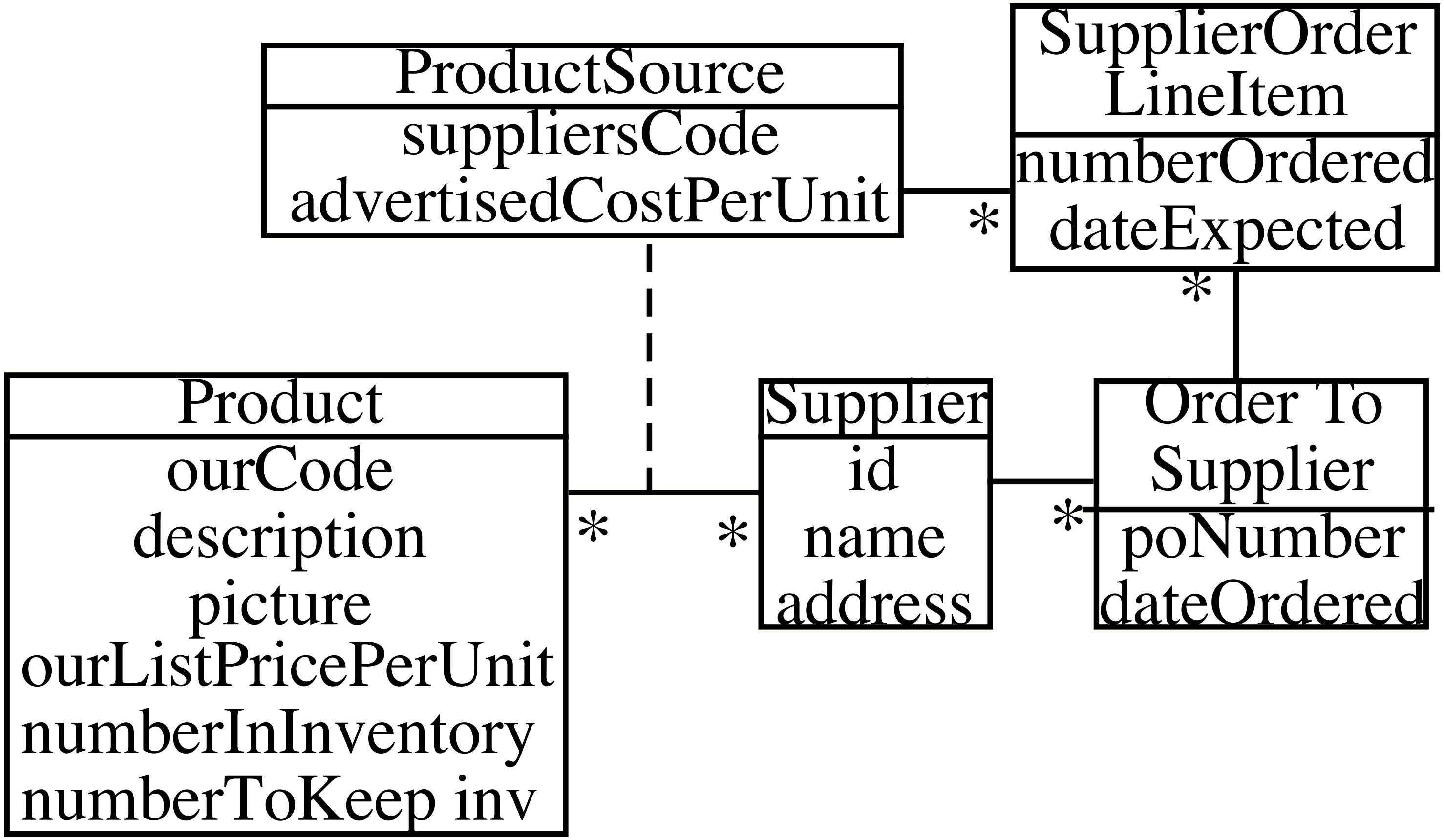
Object-Relational Mapping Rules 4

General rules cont …

C. Deal with inheritance in either of the following two ways:

1. Keep inherited material in a common table.
   - For each object, create a row in both a table for the superclass and a table for the appropriate subclass
   - Relate the two tables using a foreign key in the subclass table
   - This is the recommended approach, but has the following drawbacks:
     - » It requires a join to access any object
     - » It requires inventing an identifier
2. Inherited columns can be duplicated in all subclass tables
   - There may be no need for a table representing the superclass
   - The duplication reduces maintainability

Example class diagram for an inventory system



Inventory System Tables 1

Resulting tables:	**Reason**
Product	**A**
• product-code (key)	A2, A6i
• description	A2
• list-price-per-unit	A2
• number-in-inventory	A2
• number-to-keep-in-inv	A2
Product-Picture	**A4**
• product-code (foreign-key)	
• picture-bitmap	

Inventory System Tables 2

Supplier	**A**
• supplier-id (key)	A2, A6i
• name	A2
• street	A3
• city	A3
• postal-code	A3

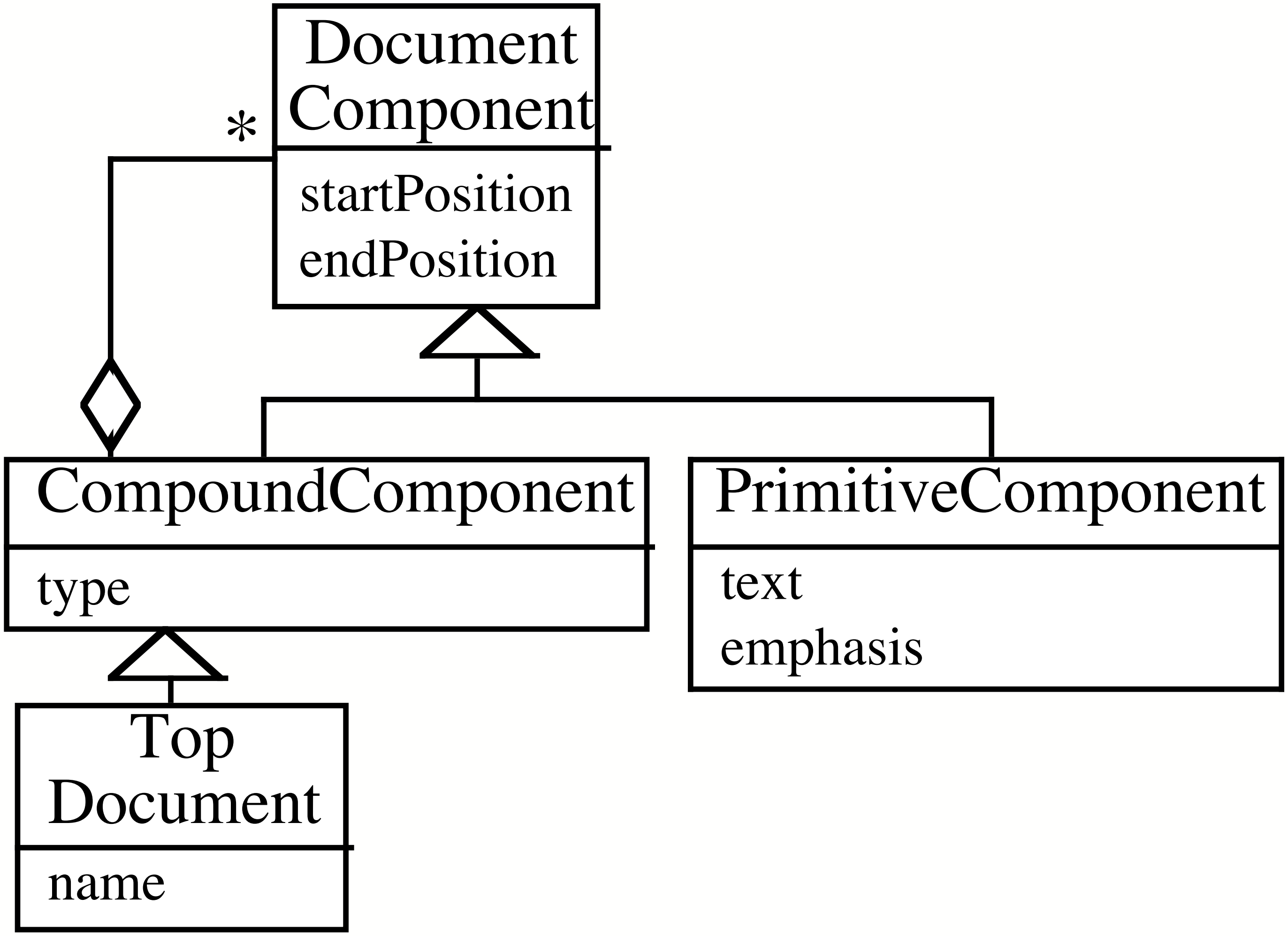
Product-Source	**B**
• product-source-id (key)	B2
• product-code (foreign-key)	B1
• supplier-id (foreign-key)	B1
• suppliers-code-for-product	A2
• advertised-cost-per-unit	A2

Inventory System Tables 3

Order-To-Supplier	**A**
• po-number (key)	A2, A6i
• supplier-id (foreign-key)	A5i
• date-ordered	A2

Supplier-Order-Line-Item	**A**
• line-item-id	A6ii
• po-number (foreign-key)	A5i
• product-source-id (foreign-key)	A5i
• number-ordered	A2
• date-expected	A2

Example Class Diagram for Document System - with Inheritance



Document System Tables 1

Resulting tables:	**Reason**
Document-Component	**A**
• component-key (key)	A6ii
• start-pos	A2
• end-pos	A2
Compound-Component	**A**
• component-key (foreign-key)	C1
• type	A2

Document System Tables 2

Primitive-Component	**A**
• component-key (foreign-key)	C1
• text	A2
• emphasis	A2
Top-Document	**A**
• component-key (foreign-key)	C2
• type	C2
• name	A2
Part-Relation	**A5ii**
• part-key (foreign-key)	
• whole-key (foreign-key)	

Data modeling vs. OO modeling

Data Modelling followed by RDBMS use	**Object-Oriented Modelling followed by OODMBS use**
Associations computed using joins - Keys have to be developed	Associations are explicit using pointers
Only primitive data stored in columns (characters, numbers)	Structured data of arbitrary complexity can be stored.
Code independently developed and in different places	Code found in classes
An ‘object’ can be distributed among tables - A problem for complex objects	An object in one place, so is fast to access
Associations are by default bi-directional - I am the son of my father and he is the father of me	Associations can be unidirectional - I know the Queen, but she doesn’t know me