




Systems Analysis and Design II (IS352)



Dr. Ahmed Awad

Spring 2017

Outline

- 
- Object Relational Mapping (ORM)
 - Workflow systems Architectures
 - Business Process Execution Language for Web Services (PBEL4WS)
 - Log-based Performance Analysis
 - Process Mining

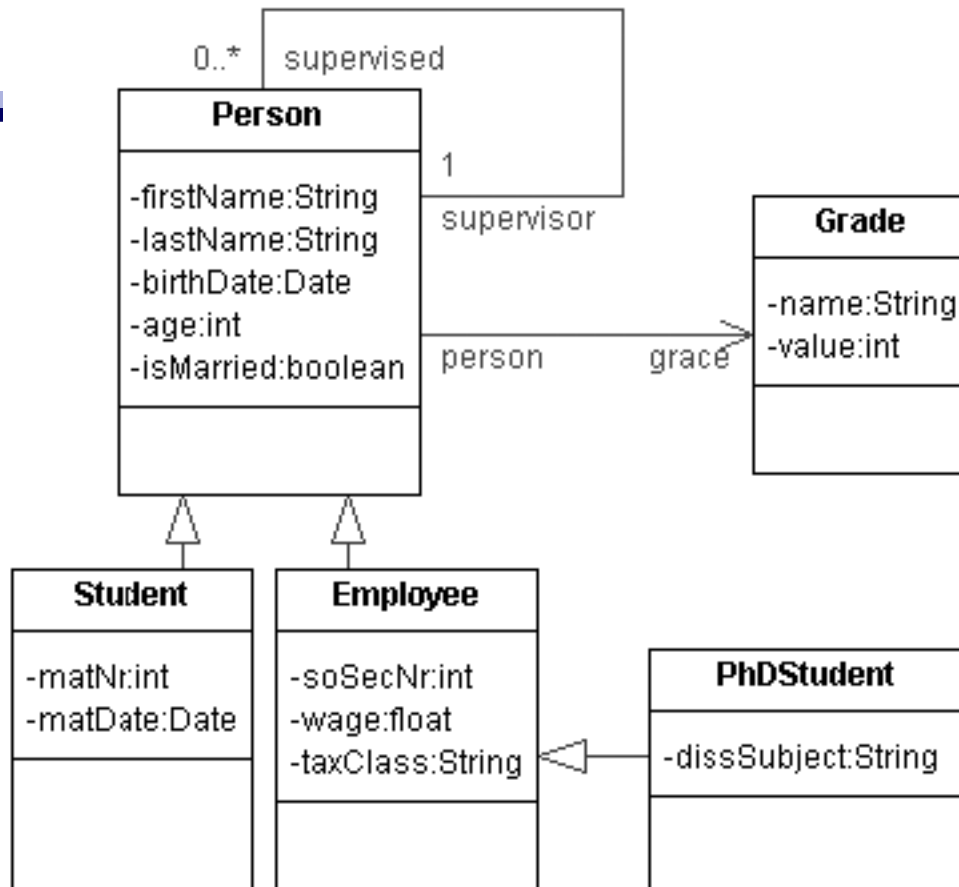
Transiting from Analysis to Design/Configuration/Enactment

Object-Relational Mapping

What is ORM?



Object-relational mapping, in the purest sense, is a programming technique that supports the conversion of incompatible types in object-oriented programming languages, specifically between a data store and programming objects.



Why ORM?



- **Productivity:** The data access code is usually a significant portion of a typical application, and the time needed to write that code can be a significant portion of the overall development schedule. When using an ORM tool, the amount of code is unlikely to be reduced—in fact, it might even go up—but the ORM tool generates 100% of the data access code automatically based on the data model you define, in mere moments.

Why ORM?



- Application design: A good ORM tool designed by very experienced software architects will implement effective design patterns that almost force you to use good programming practices in an application. This can help support a clean separation of concerns and independent development that allows parallel, simultaneous development of application layers.

Why ORM?



- **Code Reuse:** If you create a class library to generate a separate DLL for the ORM-generated data access code, you can easily reuse the data objects in a variety of applications. This way, each of the applications that use the class library need have no data access code at all.

Why ORM?



- Application Maintainability: All of the code generated by the ORM is presumably well-tested, so you usually don't need to worry about testing it extensively.

ORM in one sentence...



Never write SQL statements in your source code!

Relational Databases

- Collection of tables
 - Comprised of fields that define entities
 - Primary key has unique values in each row of a table
 - Foreign key is primary key of another table
- Tables related to each other
 - Primary key field of a table is a field of another table and called a foreign key
 - Relationship established by a foreign key of one table connecting to the primary key of another table

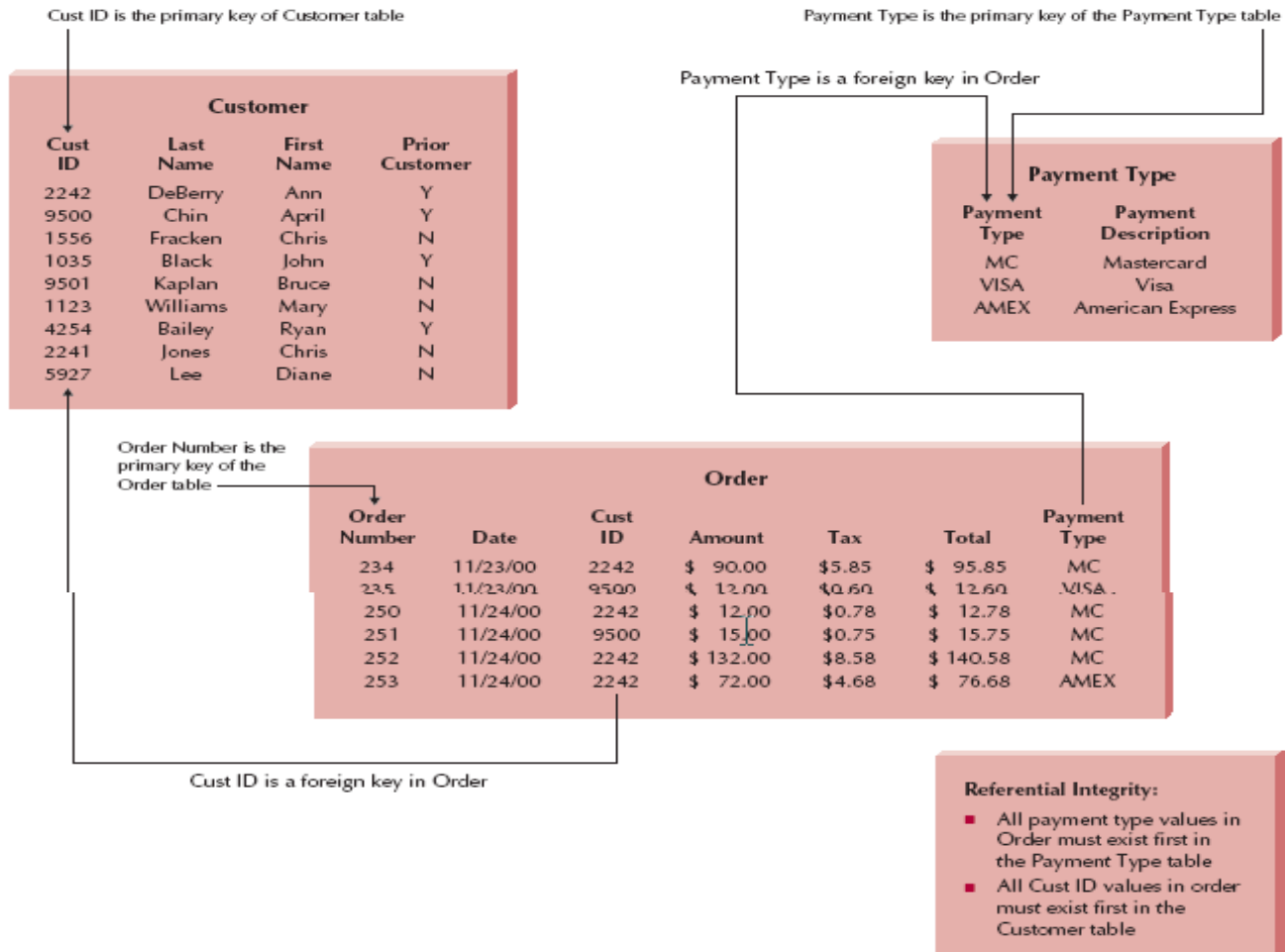
Referential Integrity

- the idea of ensuring that values linking the tables together through the primary and foreign keys are valid and correctly synchronized.

Referential Integrity Example

- Cust. ID is a primary key for the customer table
- Cust. ID is a foreign key for the order table
- A violation of referential integrity would happen if an order was entered in the order table for a Cust. ID that had not been entered into the customer table first
- An RDBMS prevents such a record from being entered

Example of Referential Integrity



Object-Relational Mapping (ORM)

- ❑ Mapping attributes
- ❑ Mapping classes
- ❑ Mapping relationships
 - Inheritance
 - Associations
- ❑ Mapping methods?

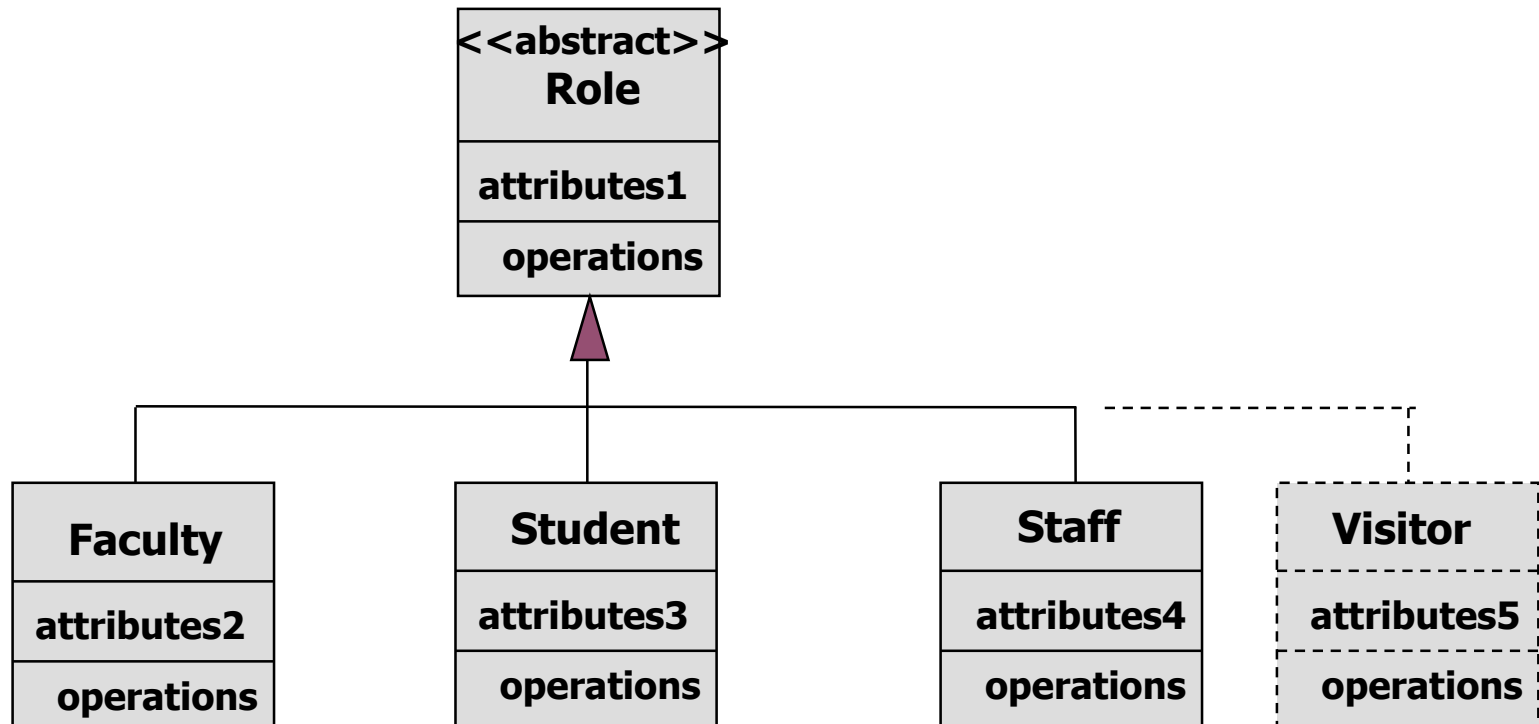
Mapping attributes

- ❑ The most straightforward mapping
- ❑ Simple types are common between object paradigm and relational models
- ❑ Strings, dates, integers are common
 - String in Java is varchar(length) in SQL Server

Mapping classes

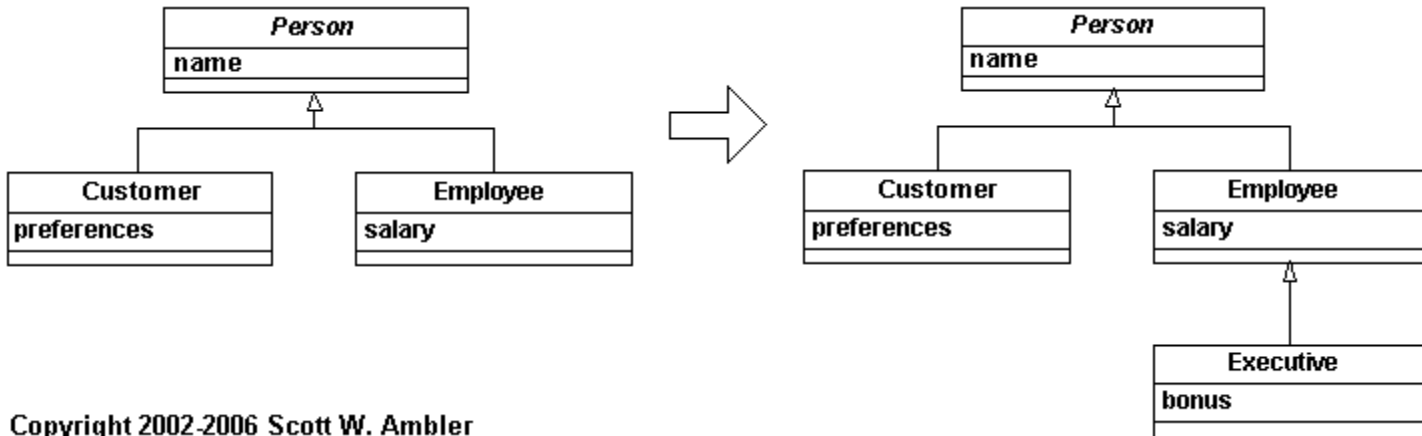
- ❑ In general, classes map to tables in one-to-one mapping
- ❑ In some cases, multiple classes can map to a single table, e.g., inheritance

Mapping Inheritance



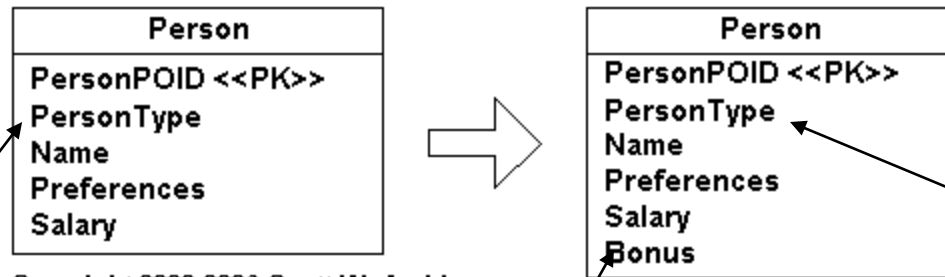
Mapping Inheritance

- ❑ Map the entire class hierarchy to a single table
- ❑ Map each concrete class to its own table
- ❑ Map each class to its own table
- ❑ Map the classes into a generic table structure



Copyright 2002-2006 Scott W. Ambler

Map Hierarchy to a single table



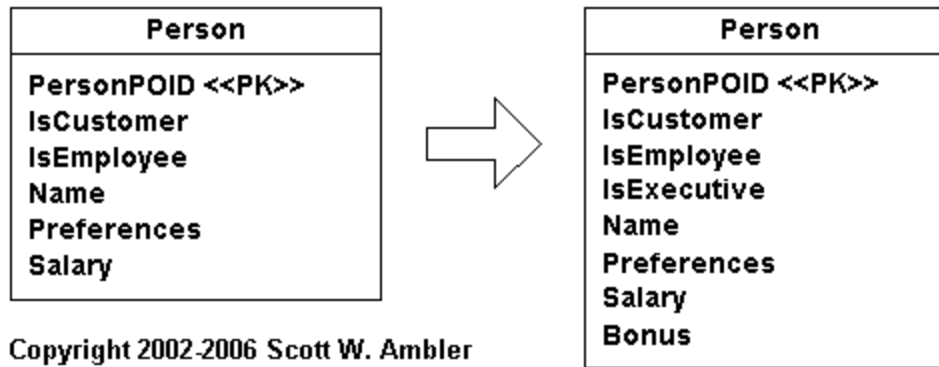
Copyright 2002-2006 Scott W. Ambler

Discriminator:
Customer, Employee

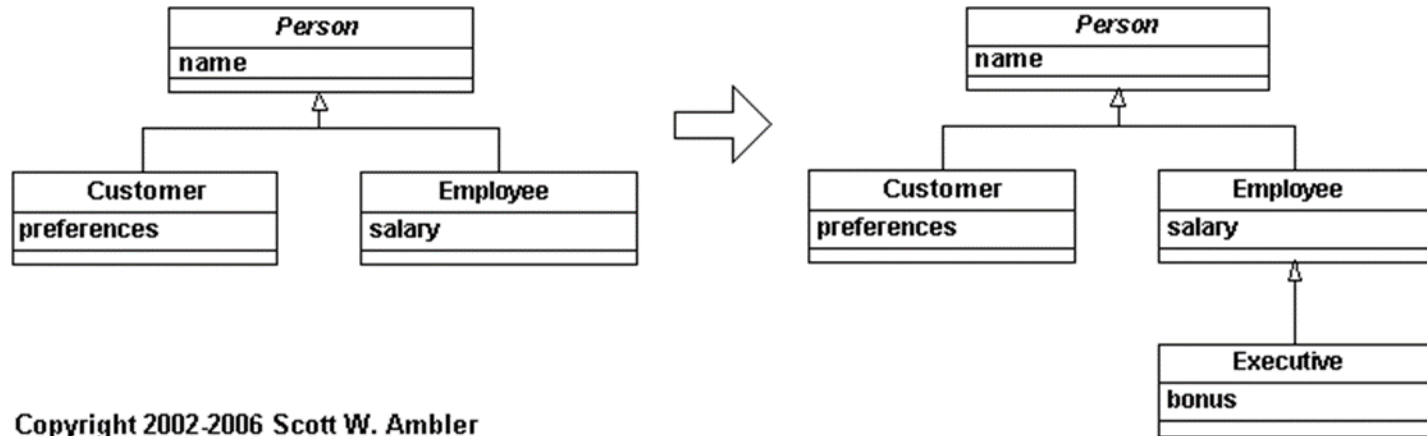
Extra column: for the executive class

Discriminator:
Customer, Employee,
Executive

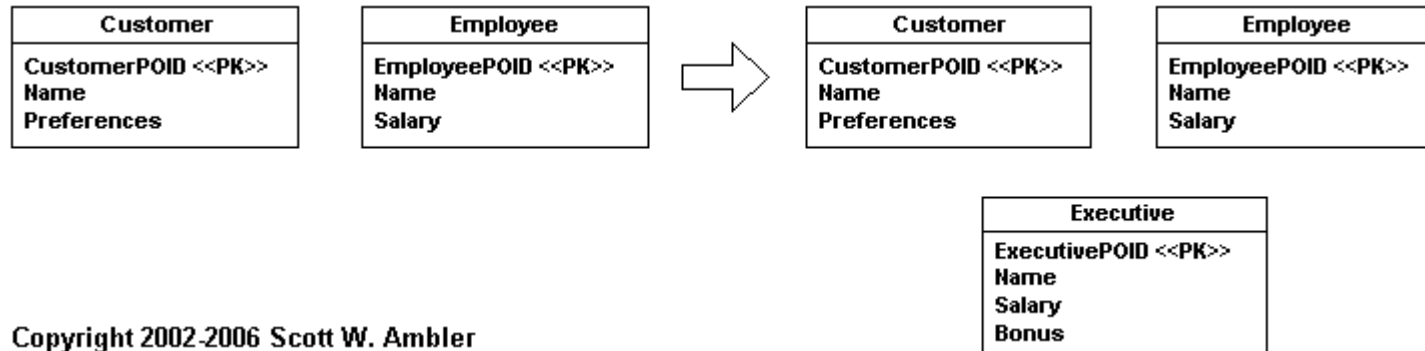
What if Types overlap



Map Each Concrete Class to its Own Table

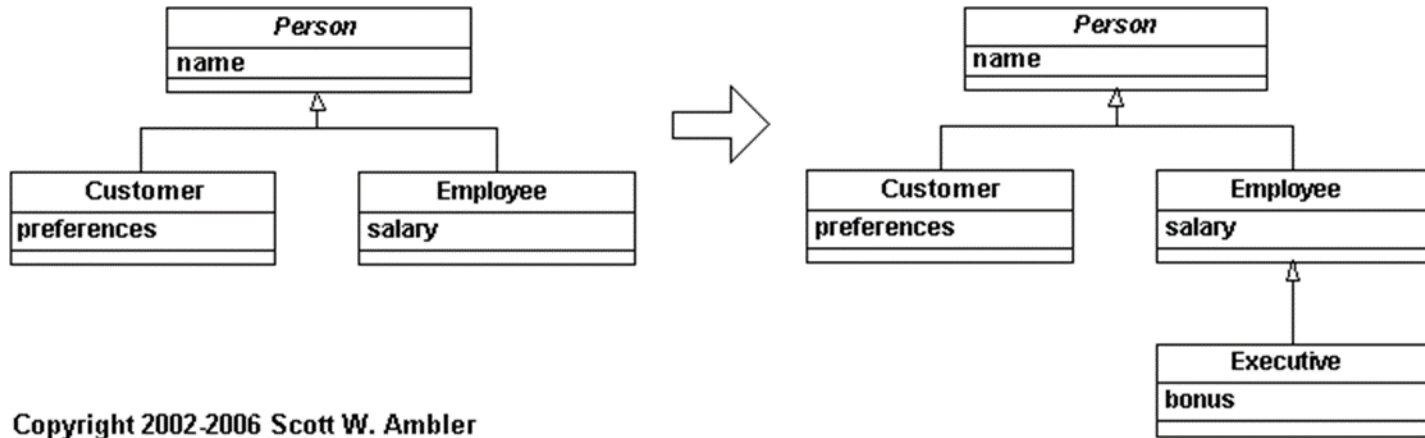


Copyright 2002-2006 Scott W. Ambler

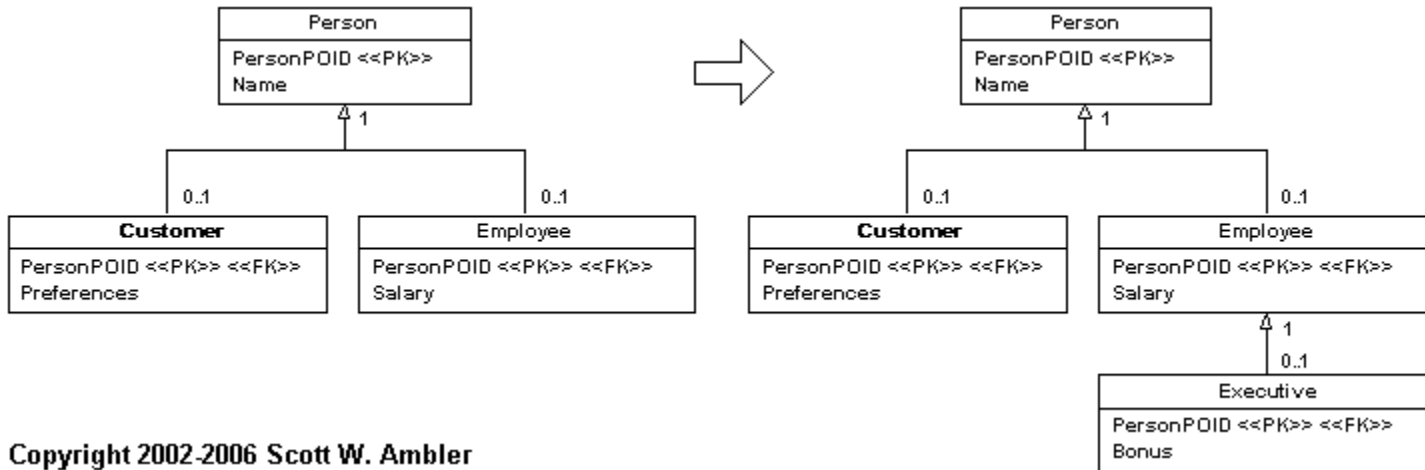


Copyright 2002-2006 Scott W. Ambler

Map Each class to its Own Table



Copyright 2002-2006 Scott W. Ambler



Copyright 2002-2006 Scott W. Ambler

Comparison

Strategy	Advantages	Disadvantages	When to Use
<p>One table per hierarchy</p>	<p>Simple approach.</p> <p>Easy to add new classes, you just need to add new columns for the additional data.</p> <p>Supports polymorphism by simply changing the type of the row.</p> <p>Data access is fast because the data is in one table.</p> <p>Ad-hoc reporting is very easy because all of the data is found in one table.</p>	<p>Coupling within the class hierarchy is increased because all classes are directly coupled to the same table.</p> <p>A change in one class can affect the table which can then affect the other classes in the hierarchy.</p> <p>Space potentially wasted in the database.</p> <p>Indicating the type becomes complex when significant overlap between types exists.</p> <p>Table can grow quickly for large hierarchies.</p>	<p>This is a good strategy for simple and/or shallow class hierarchies where there is little or no overlap between the types within the hierarchy.</p>

Comparison

Strategy	Advantages	Disadvantages	When to Use
One table per concrete class	<p>easy to do ad-hoc reporting as all the data you need about a single class is stored in only one table.</p> <p>Good performance to access a single object's data.</p>	<p>When you modify a class you need to modify its table and the table of any of its subclasses. For example if you were to add height and weight to the Person class you would need to add columns to the Customer, Employee, and Executive tables.</p> <p>Whenever an object changes its role, you need to copy the data into the appropriate table. It is difficult to support multiple roles and still maintain data integrity.</p>	When changing types and/or overlap between types is rare.

Comparison

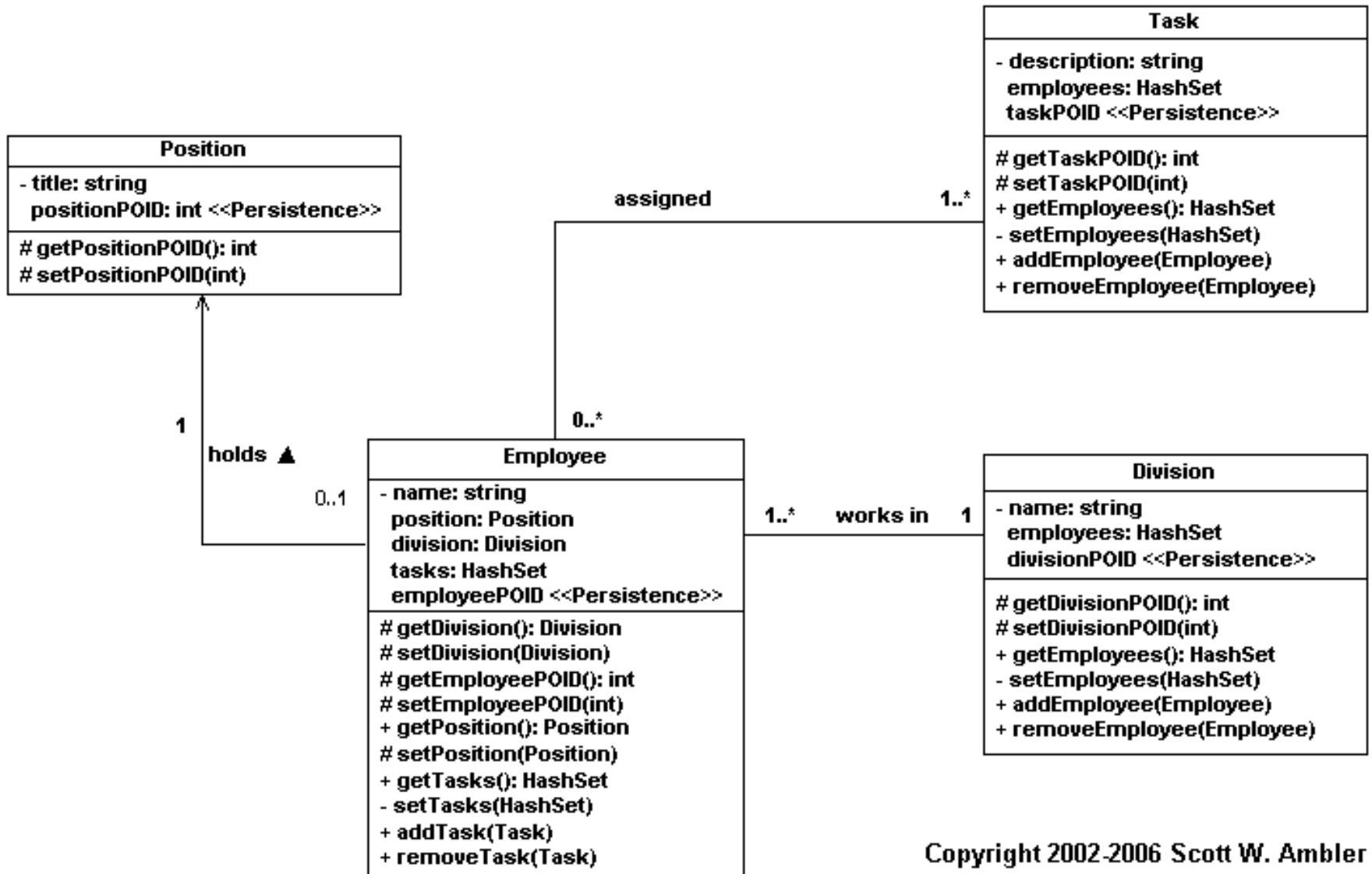
Strategy	Advantages	Disadvantages	When to Use
One table per class	<p>Easy to understand because of the one-to-one mapping.</p> <p>Supports polymorphism very well as you merely have records in the appropriate tables for each type.</p> <p>Very easy to modify superclasses and add new subclasses as you merely need to modify/add one table.</p> <p>Data size grows in direct proportion to growth in the number of objects.</p>	<p>There are many tables in the database, one for every class (plus tables to maintain relationships). Potentially takes longer to read and write data using this technique.</p> <p>Ad-hoc reporting on your database is difficult, unless you add views to simulate the desired tables.</p>	<p>When there is significant overlap between types or when changing types is common.</p>

Mapping Object Relationships

- ▣ We have:
 - ▣ Association
 - ▣ Aggregation
 - ▣ Composition
- ▣ Recall, Aggregation and composition are special types of association
- ▣ All three will be mapped to referential integrity constraints

Mapping Object Relationships

- ▣ Cardinality
 - Mapping one-to-one relationships
 - Mapping one-to-many relationships
 - Mapping many-to-many relationships
- ▣ Direction
 - Unidirectional
 - Bidirectional



Copyright 2002-2006 Scott W. Ambler

How Relational Database Relationships Are Implemented

Relationships in relational databases are maintained through the use of foreign keys. A foreign key is a data attribute(s) that appears in one table that may be part of or is coincidental with the key of another table. With a one-to-one relationship the foreign key needs to be implemented by one of the tables.

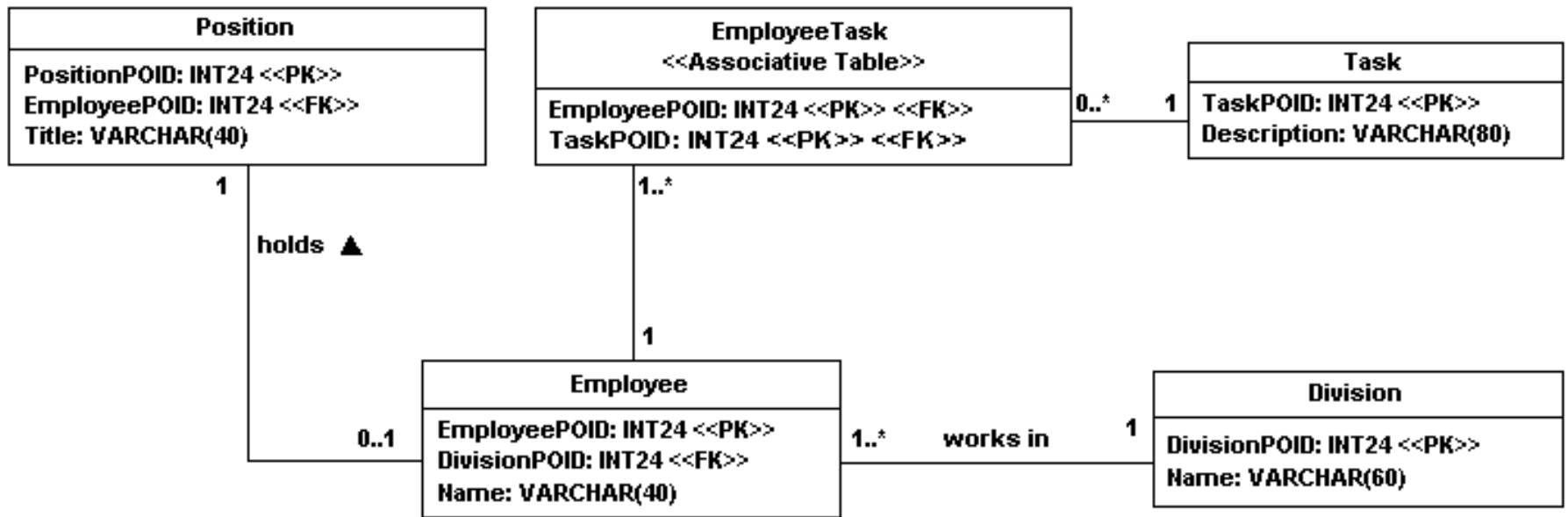
How Relational Database Relationships Are Implemented

To implement a **one-to-many** relationship you implement a foreign key from the “one table” to the “many table”. For example Employee includes a DivisionPOID column to implement the works in relationship to Division. You could also choose to overbuild your database schema and implement a one-to-many relationship via an associative table, effectively making it a many-to-many relationship.

How Relational Database Relationships Are Implemented

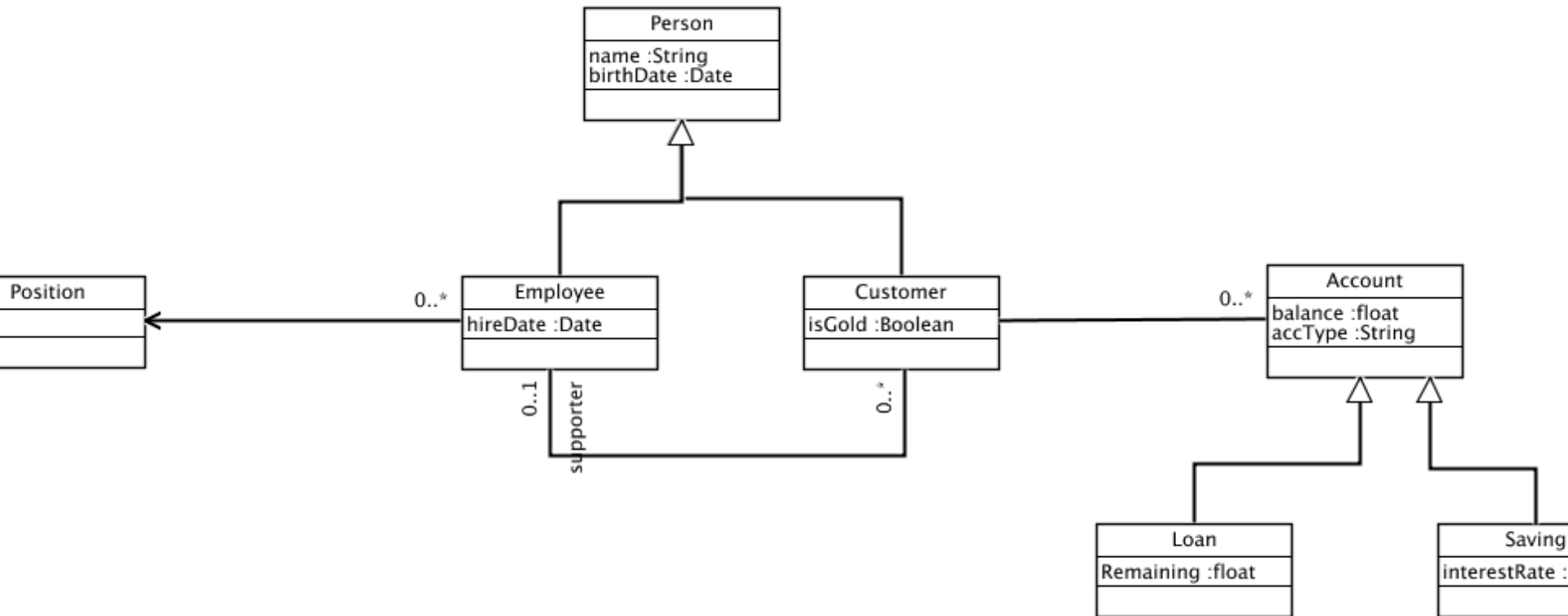
To implement **many-to-many** associations in a relational database, is to implement what is called an associative table, an example of which is *EmployeeTask*, which includes the combination of the primary keys of the tables that it associates.

The basic "trick" is that the many-to-many relationship is converted into two one-to-many relationships, both of which involve the associative table.



Copyright 2002-2006 Scott W. Ambler

More Exercises (1)



More Exercises (2)

