Section 5.3
Mapping to Storage

1. Overview
2. Relational database concepts
3. Mapping classes and attributes
4. Mapping associations
5. Mapping inheritance relationships
6. ARENA case study

5.3.1 Overview

- What objects do we map to storage?
  - map persistent objects to structures in data management system
    - the data management system is selected during system design
    - the persistent data structures may be:
      - flat files
      - relational or OO database

- How do we do this?
  - for flat files and relational database, the object model must be transformed to *storage schema*

Overview (cont.)

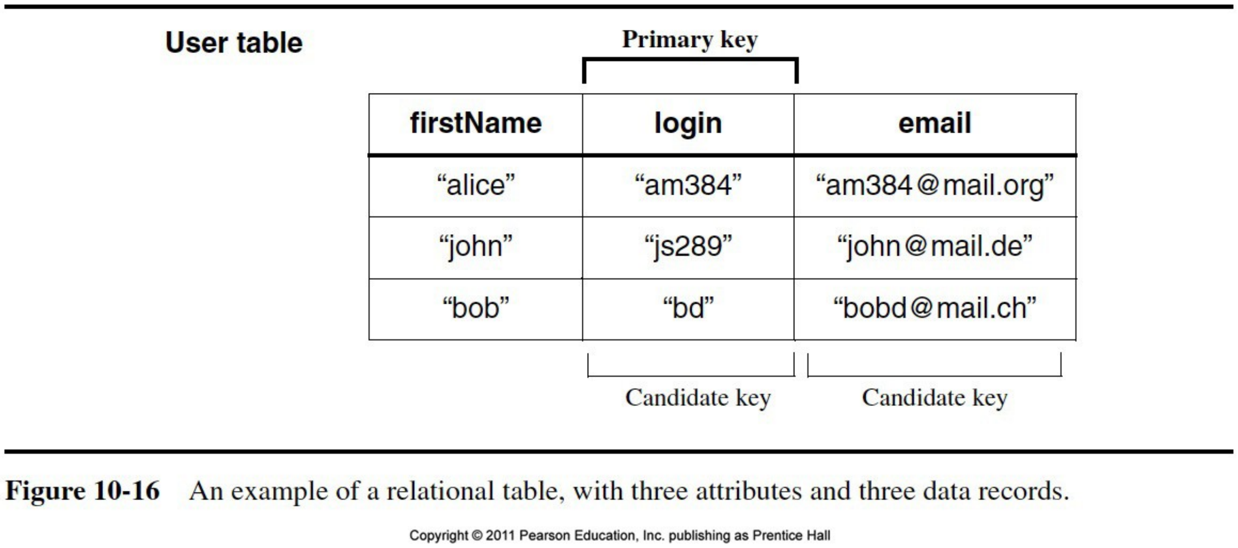
- Using relational databases to store data

➢ table:	collection of data records
➢ rows:	data records
➢ columns:	attributes
➢ cell:	value of the attribute for the corresponding record

5.3.2 Relational Database Concepts

- Schema
  - it represents a description of the data
  - it is the set of attributes that are stored for each object
  - the schema is also known as the *meta-model* for the data
- Primary key
  - a set of attributes whose values uniquely identify a data record
  - they are used to refer unambiguously to a specific data record
- Foreign key
  - an attribute that references a primary key in another table
  - links a data record in one table to more records in another table

Relational Database Concepts (cont.)



User table

Primary key

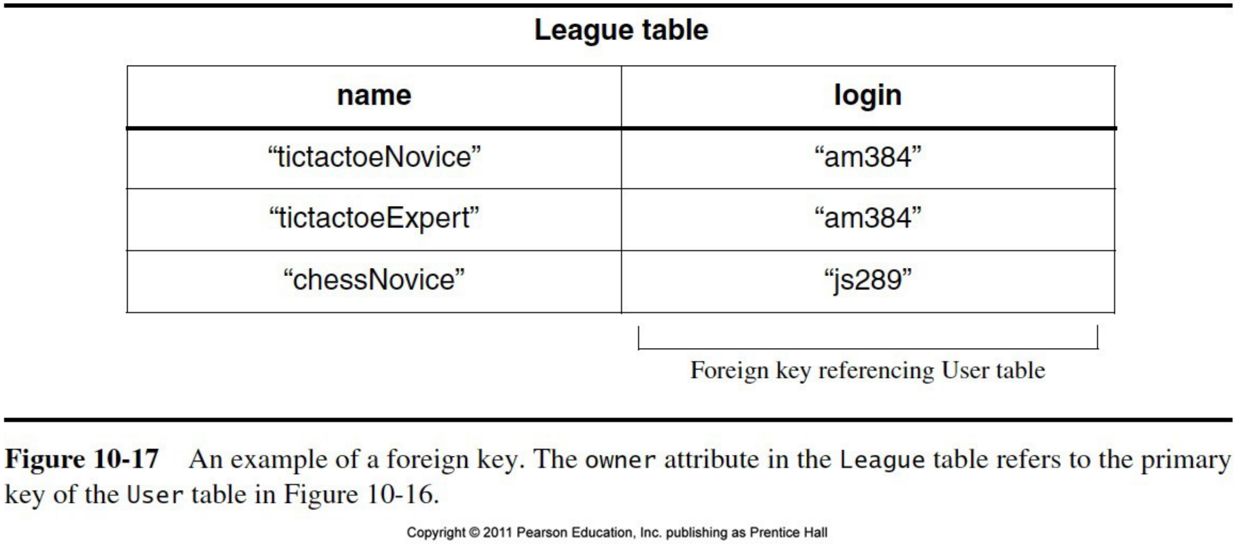
firstName	login	email
“alice”	“am384”	“am384@mail.org”
“john”	“js289”	“john@mail.de”
“bob”	“bd”	“bobd@mail.ch”

Candidate key

Candidate key

Figure 10-16 An example of a relational table, with three attributes and three data records.

Copyright © 2011 Pearson Education, Inc. publishing as Prentice Hall



League table

name	login
“tictactoeNovice”	“am384”
“tictactoeExpert”	“am384”
“chessNovice”	“js289”

Foreign key referencing User table

Figure 10-17 An example of a foreign key. The owner attribute in the League table refers to the primary key of the User table in Figure 10-16.

Copyright © 2011 Pearson Education, Inc. publishing as Prentice Hall

5.3.3 Mapping Classes and Attributes

- Correspondences between object model and schema
  - class: table
  - attribute: column
  - instance: row

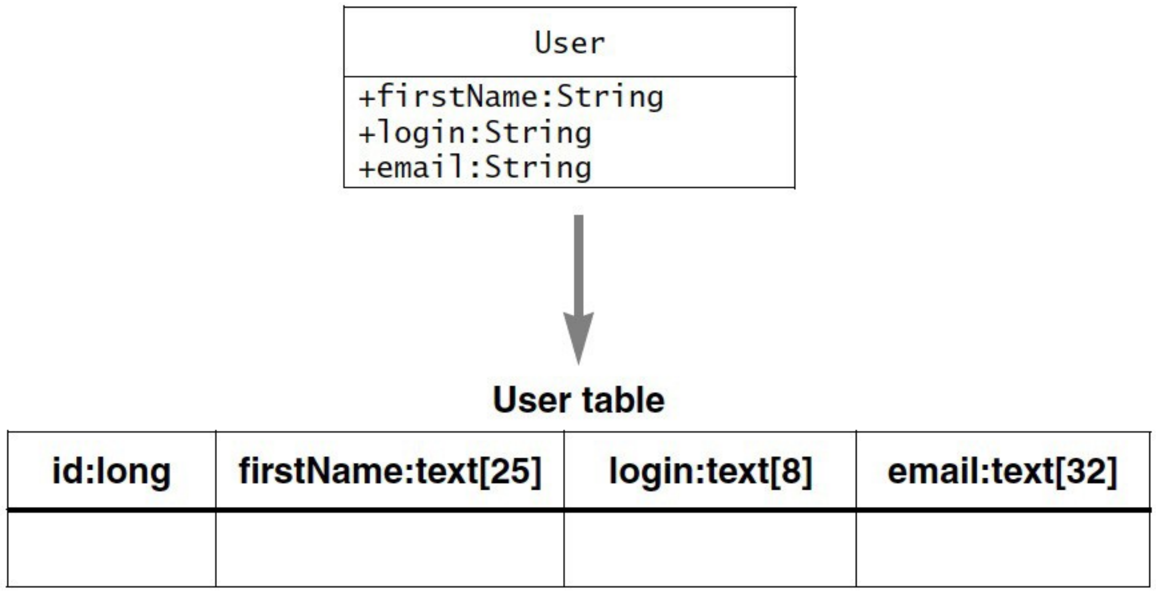
- Match the same names in the object model and schema
  - provides traceability

Mapping Classes and Attributes (cont.)

- Mapping attribute types
  - some constraints may have to be added to the object model
    - e.g. maximum string length

- Primary key
  - choose a set of class attributes
    - this is a problem if the key values change
    - this is a problem if the application domain changes
  - add a unique identifier
    - more robust

Mapping Classes and Attributes (cont.)



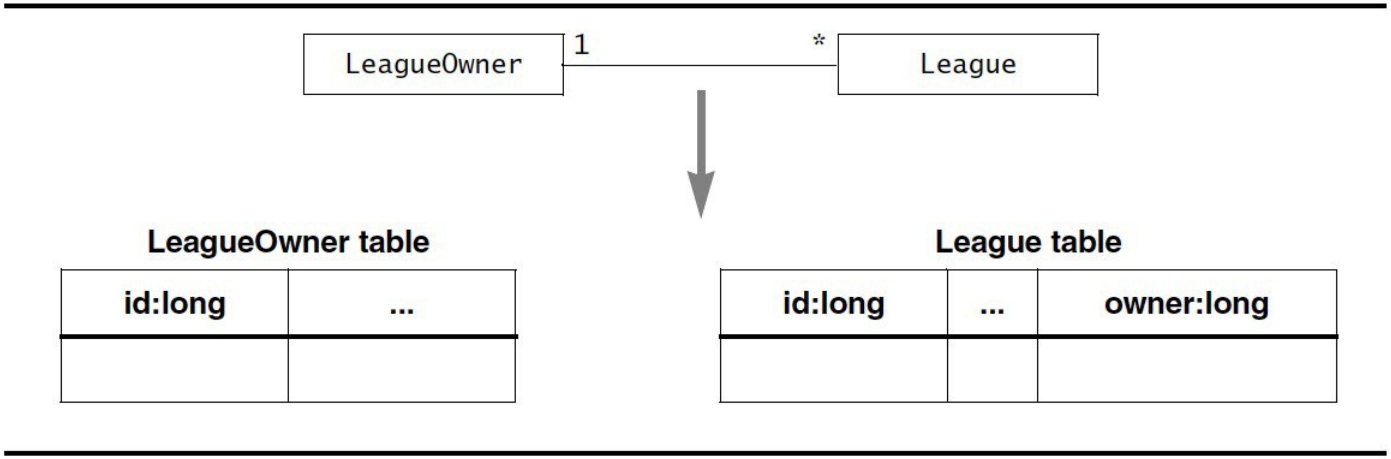
id:long	firstName:text[25]	login:text[8]	email:text[32]

Figure 10-18 Forward engineering of the User class to a database table.

Copyright © 2011 Pearson Education, Inc. publishing as Prentice Hall

5.3.4 Mapping Associations

- Buried association
  - used to implement one-to-one and one-to-many associations
  - one-to-one: include the foreign key of the destination object in the record of the source object (and vice-versa for bidirectional association)
  - one-to-many: include the foreign key of the source object (“one” side) in the records of the destination objects (“many” side)



LeagueOwner table

id:long	...

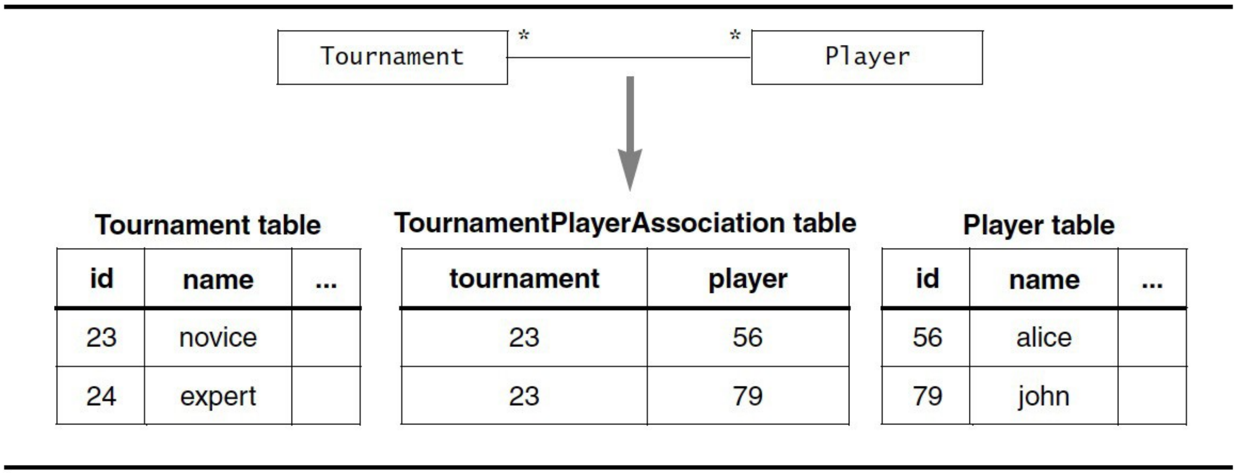
League table

id:long	...	owner:long

Figure 10-19 Mapping of the LeagueOwner/League association as a buried association.

Mapping Associations (cont.)

- Association table
  - it is used to implement many-to-many associations
  - we create a new two-column table with foreign keys for both classes in the association
  - each row corresponds to one link
  - it can be used for one-to-one and one-to-many associations
    - it increases the number of tables
    - it increases the time required to traverse the associations



Tournament table

id	name	...
23	novice	
24	expert	

TournamentPlayerAssociation table

tournament	player
23	56
23	79

Player table

id	name	...
56	alice	
79	john	

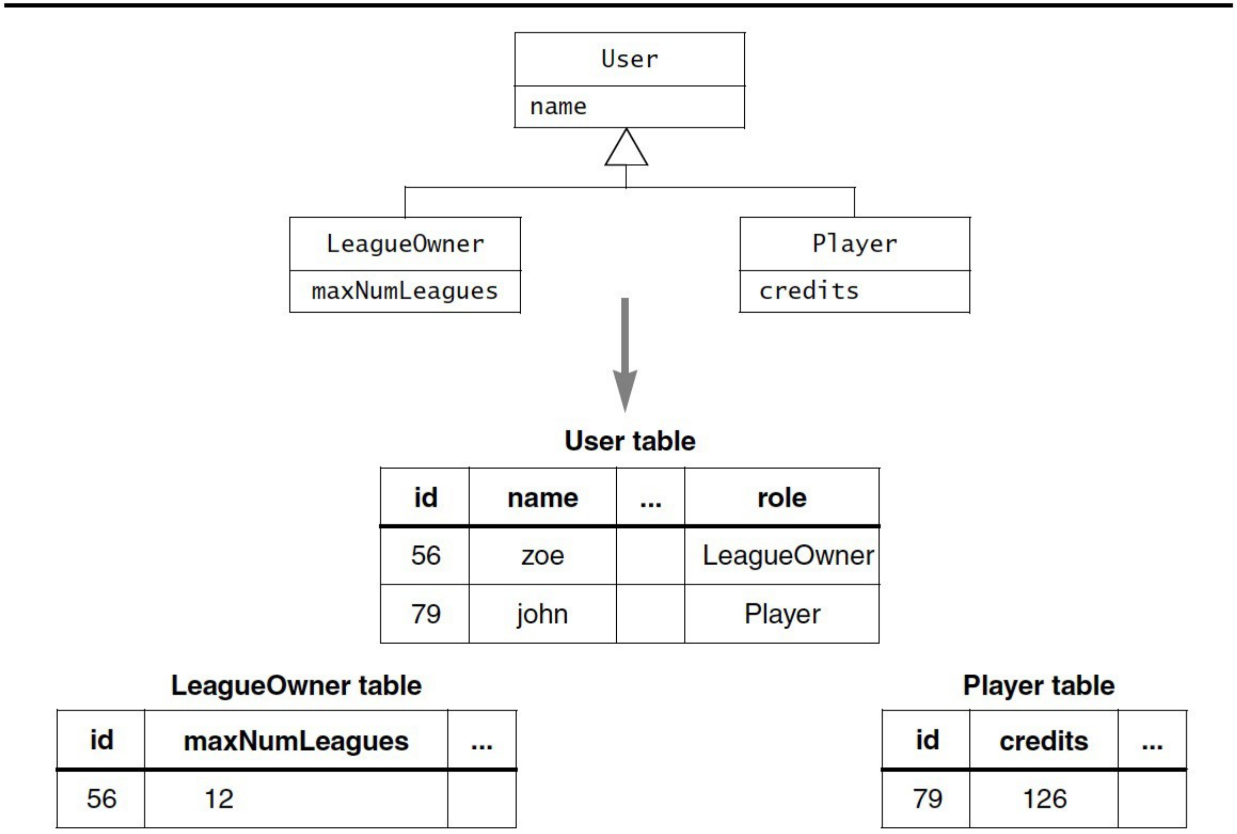
Figure 10-20 Mapping of the `Tournament/Player` association as a separate table.

5.3.5 Mapping Inheritance Relationships

- Vertical mapping
  - superclass and subclass each have their own table
  - superclass table:
    - contains superclass attributes
    - includes an additional attribute for name of record's actual subclass
  - subclass table:
    - contains subclass attributes
    - shares the *same key* as the superclass table
  - access to one object involves multiple table retrievals

Mapping Inheritance Relationships (cont.)

- Vertical mapping (cont.)



User table

id	name	...	role
56	zoe		LeagueOwner
79	john		Player

LeagueOwner table

id	maxNumLeagues	...
56	12	

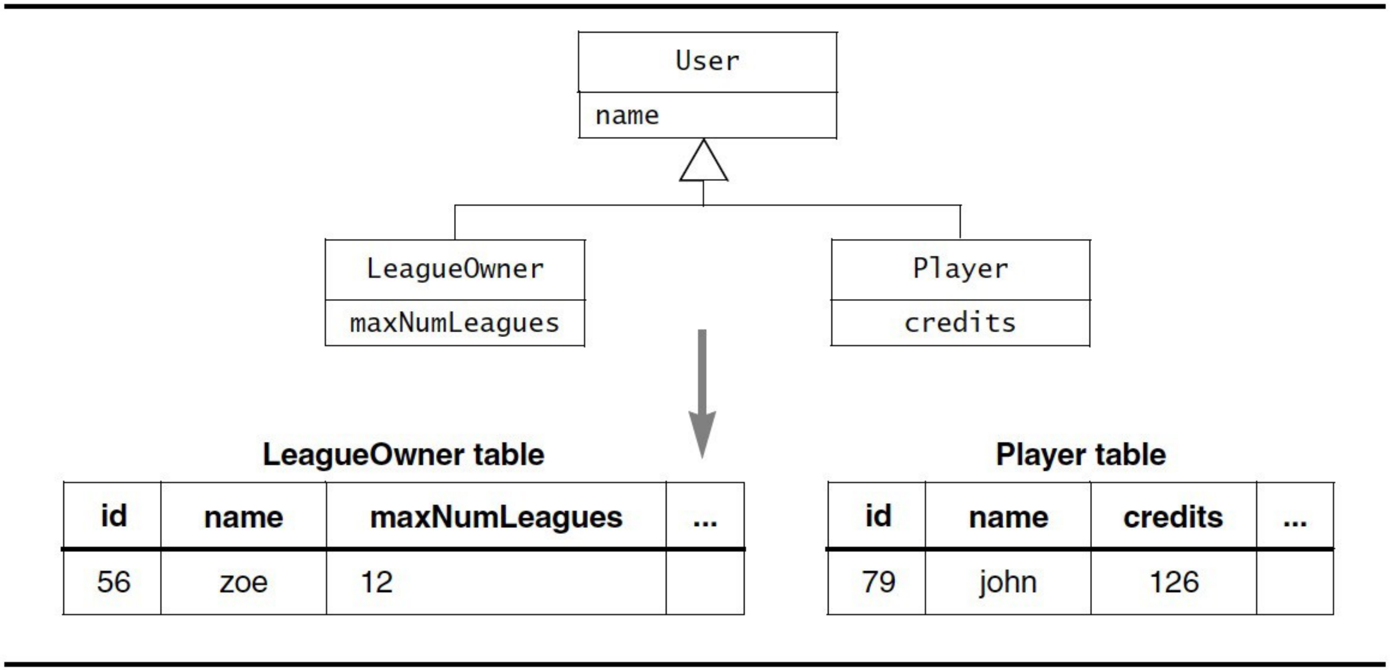
Player table

id	credits	...
79	126	

Figure 10-21 Realizing the `User` inheritance hierarchy with a separate table.

Mapping Inheritance Relationships (cont.)

- Horizontal mapping
  - only the subclass has a table
  - that table includes the attributes from the superclass and subclass
  - access to one object involves a single table retrieval



LeagueOwner table

id	name	maxNumLeagues	...
56	zoe	12	

Player table

id	name	credits	...
79	john	126	

Figure 10-22 Realizing the User inheritance hierarchy by duplicating columns.

Copyright © 2011 Pearson Education, Inc. publishing as Prentice Hall

Mapping Inheritance Relationships (cont.)

- Trade-offs
  - vertical mapping
    - adds to access time with multiple table retrievals
    - facilitates modifiability, e.g. when adding attributes to superclass
  - horizontal mapping
    - duplicates superclass columns for each subclass
    - schema modifications are more complex
    - queries are faster, especially with deep inheritance

5.3.6 ARENA Case Study

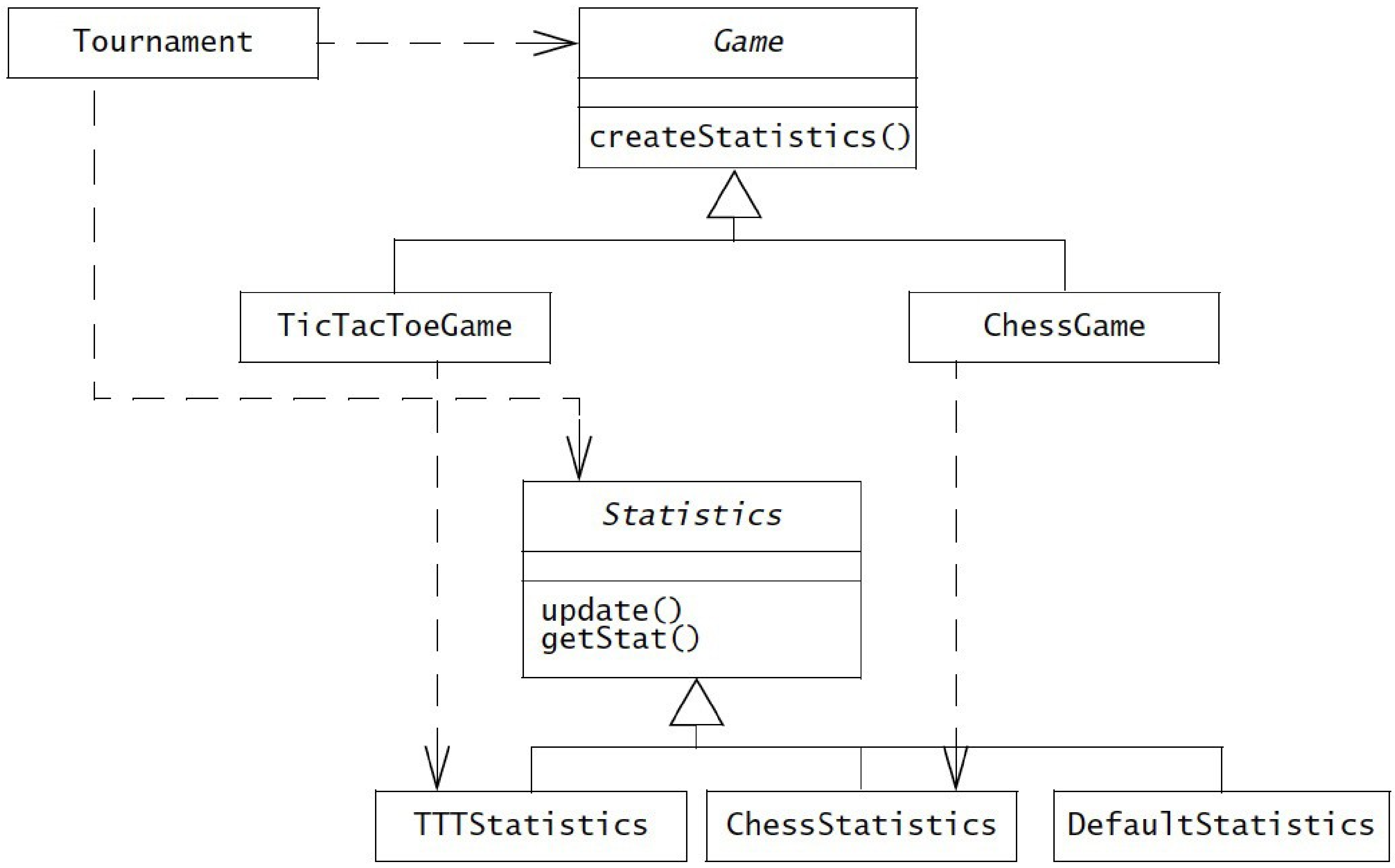


Figure 10-23 Statistics as a product in the *Game* Abstract Factory (UML class diagram).

ARENA Case Study (cont.)

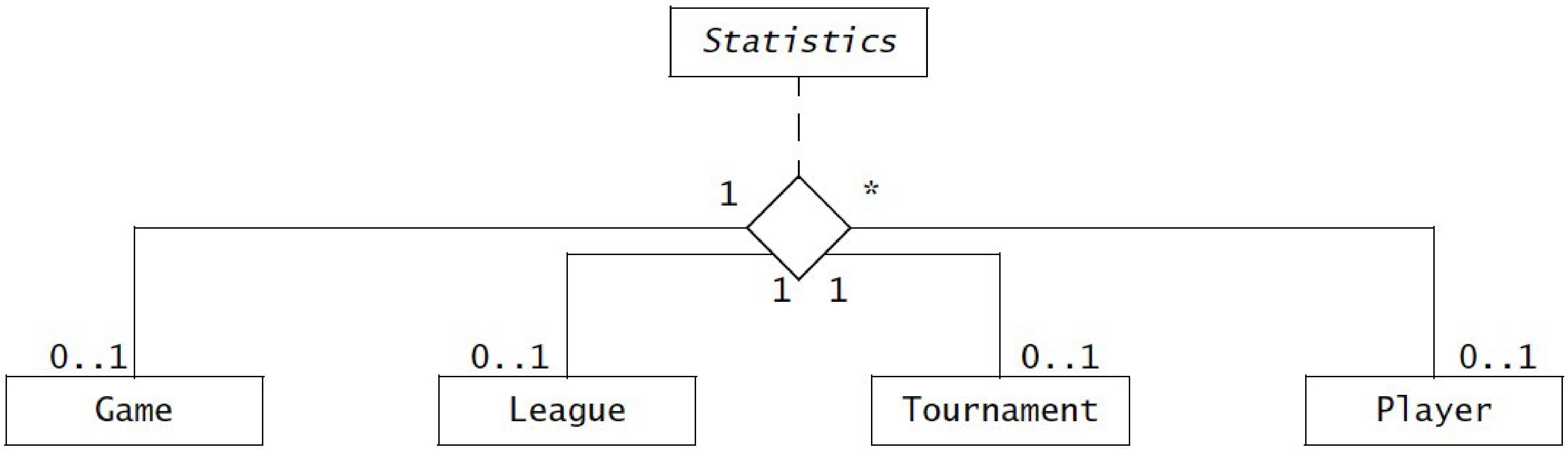


Figure 10-24 N-ary association class Statistics relating League, Tournament, and Player (UML class diagram).

ARENA Case Study (cont.)

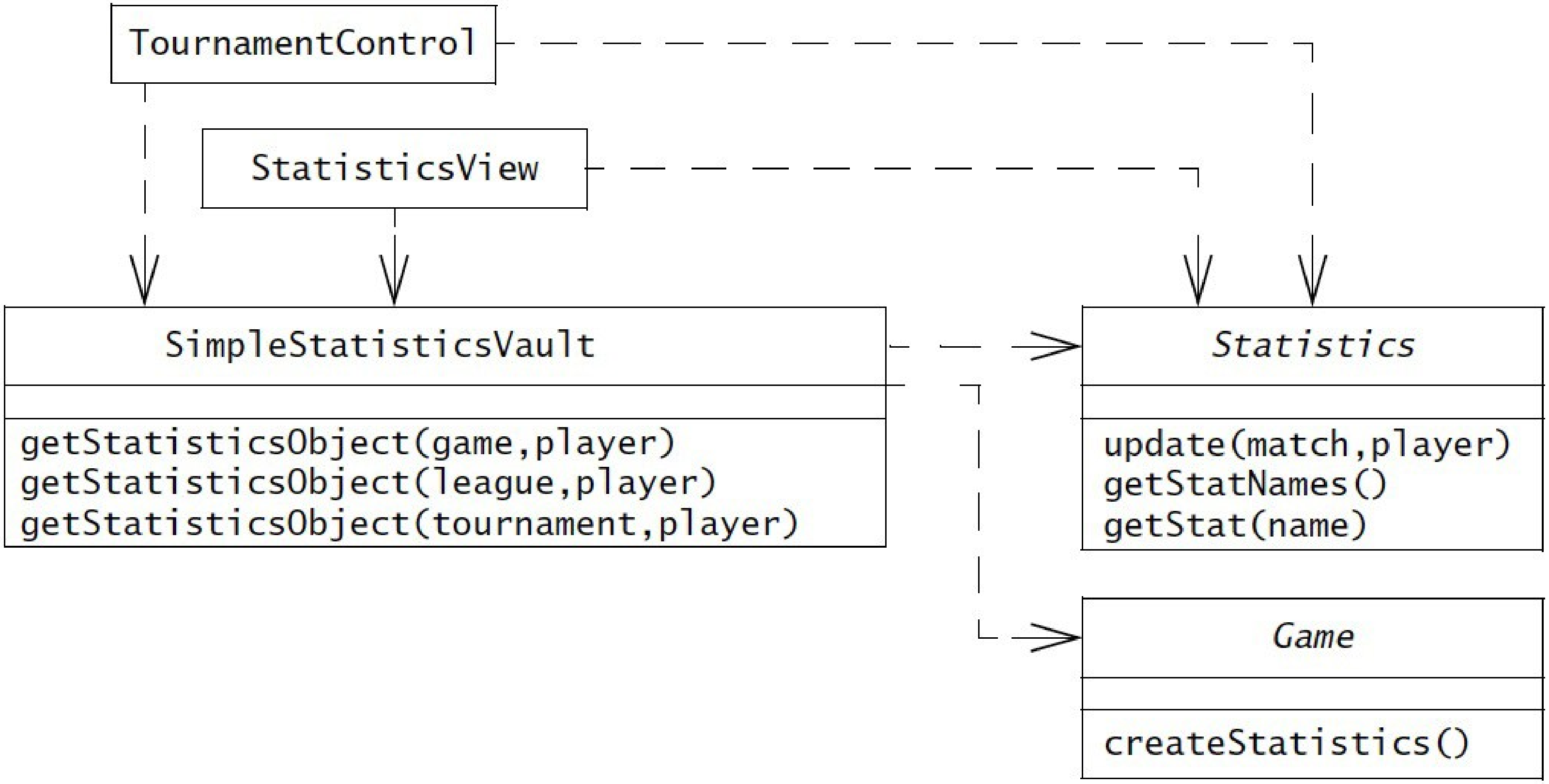


Figure 10-25 SimpleStatisticsVault object realizing the N-ary association of Figure 10-24.

ARENA Case Study (cont.)

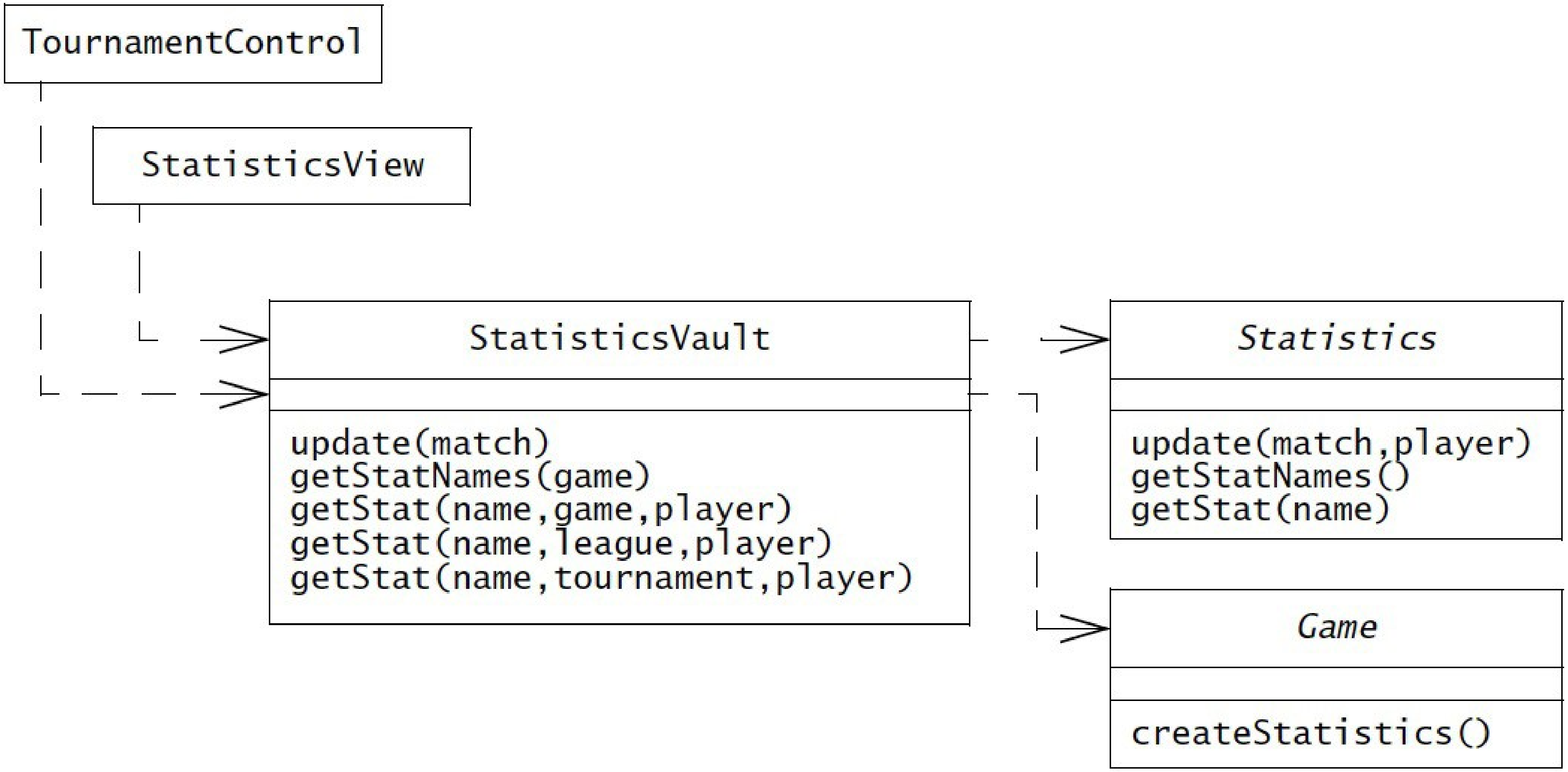
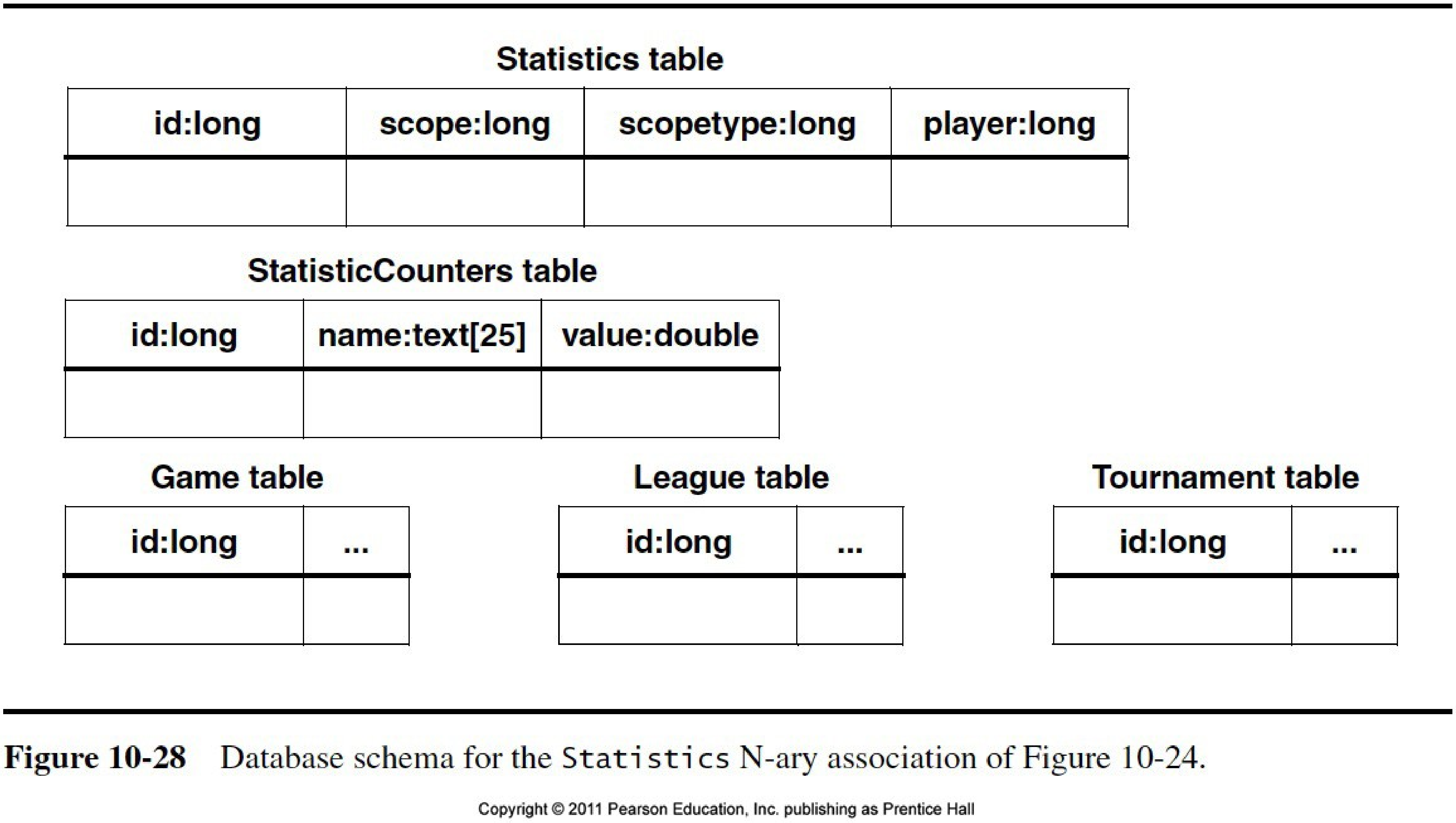


Figure 10-26 StatisticsVault as a Facade shielding the control and boundary objects from the Statistics storage and computation (UML class diagram).

ARENA Case Study (cont.)



Statistics table

id:long	scope:long	scopetype:long	player:long

StatisticCounters table

id:long	name:text[25]	value:double

Game table

id:long	...

League table

id:long	...

Tournament table

id:long	...

Figure 10-28 Database schema for the `Statistics` N-ary association of Figure 10-24.

Implementation Recap

- What we learned:
  - understand strategies for mapping models to code
  - understand strategies for mapping models to persistent storage
    - mapping associations to collections
    - mapping contracts to exceptions
    - mapping object model to storage