Chapter 9:
Data Management Layer Design

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Objectives

- Become familiar with several object-persistence formats.
- Be able to map problem domain objects to different object-persistence formats.
- Be able to apply the steps of normalization to a relational database.
- Be able to optimize a relational database for object storage and access.
- Become familiar with indexes for relational databases.
- Be able to estimate the size of a relational database.
- Understand the effect of nonfunctional requirements on the data management layer
- Be able to design the data access and manipulation classes.

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Introduction

- Applications are of little use without data
  - Data must be stored and accessed efficiently
- The data management layer includes:
  - Data access and manipulation logic
  - Storage design
- Four-step design approach:
  - Selecting the format of the storage
  - Mapping problem-domain objects to object persistence format
  - Optimizing the object persistence format
  - Designing the data access & manipulation classes

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Object Persistence Formats

- Files (sequential and random access)
- Object-oriented databases
- Object-relational databases
- Relational databases
- “NoSQL” data stores

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Electronic Files

- Sequential access files
  - Operations (read, write and search) are conducted one record after another (in sequence)
  - Efficient for report writing
  - Inefficient for searching (an average of 50% of records have to be accessed for each search)
  - Unordered files add records to the end of the file
  - Ordered files are sorted, but additions & deletions require additional maintenance
- Random access files
  - Efficient for operations (read, write and search)
  - Inefficient for report writing

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Application File Types

- Master Files
  - Store core information (e.g., order and customer data)
  - Usually held for long periods
  - Changes require new programs
- Look-up files (e.g., zip codes with city and state names)
- Transaction files
  - Information used to update a master file
  - Can be deleted once master file is updated
- Audit file—records data before & after changes
- History file—archives of past transactions

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Relational Databases

- Most popular way to store data for applications
- Consists of a collection of tables
  - Primary key uniquely identifies each row
  - Foreign keys establish relationships between tables
    - Referential integrity ensures records in different tables are matched properly
    - Example: you cannot enter an order for a customer that does not exist
- Structured Query Language (SQL) is used to access the data
  - Operates on complete tables vs. individual records
  - Allows joining tables together to obtain matched data

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Object-Relational Databases

- A standard relational database with ability to store objects added
- Must create a mapping from UML class diagrams to database schema is required.
- Accomplished using user-defined data types
  - SQL extended to handle complex data types
  - Support for inheritance varies

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Object-Oriented Databases

- Two approaches:
  - Add persistence extensions to OO programming language
  - Create a separate OO database
- Utilize extents—a collection of instances of a class
  - A table associated with a class.
  - Each row an instance of the class, and having an Object ID
  - Object IDs relate objects together
  - Another primary key may still be desirable
- Inheritance is supported but is language dependent
- Represent a small market share due to its steep learning curve

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

NoSQL Data Stores

- Newest type; used primarily for complex data types
  - Does not support SQL
  - No standards exist
  - Support very fast queries
- Data may not be consistent since there are no locking mechanisms
- Types
  - Key-value data stores
  - Document data stores
  - Columnar data stores
- Immaturity of technology prevents traditional business application support

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Selecting Persistence Formats

	Sequential and Random Access Files	Relational DBMS	Object Relational DBMS	Object-Oriented DBMS
Major Strengths	Usually part of an object-oriented programming language Files can be designed for fast performance Good for short-term data storage	Leader in the database market Can handle diverse data needs	Based on established, proven technology, e.g., SQL Able to handle complex data	Able to handle complex data Direct support for object orientation
Major Weaknesses	Redundant data Data must be updated using programs, i.e., no manipulation or query language No access control	Cannot handle complex data No support for object orientation Impedance mismatch between tables and objects	Limited support for object orientation Impedance mismatch between tables and objects	Technology is still maturing Skills are hard to find
Data Types Supported	Simple and Complex	Simple	Simple and Complex	Simple and Complex
Types of Application Systems Supported	Transaction processing	Transaction processing and decision making	Transaction processing and decision making	Transaction processing and decision making
Existing Storage Formats	Organization dependent	Organization dependent	Organization dependent	Organization dependent
Future Needs	Poor future prospects	Good future prospects	Good future prospects	Good future prospects

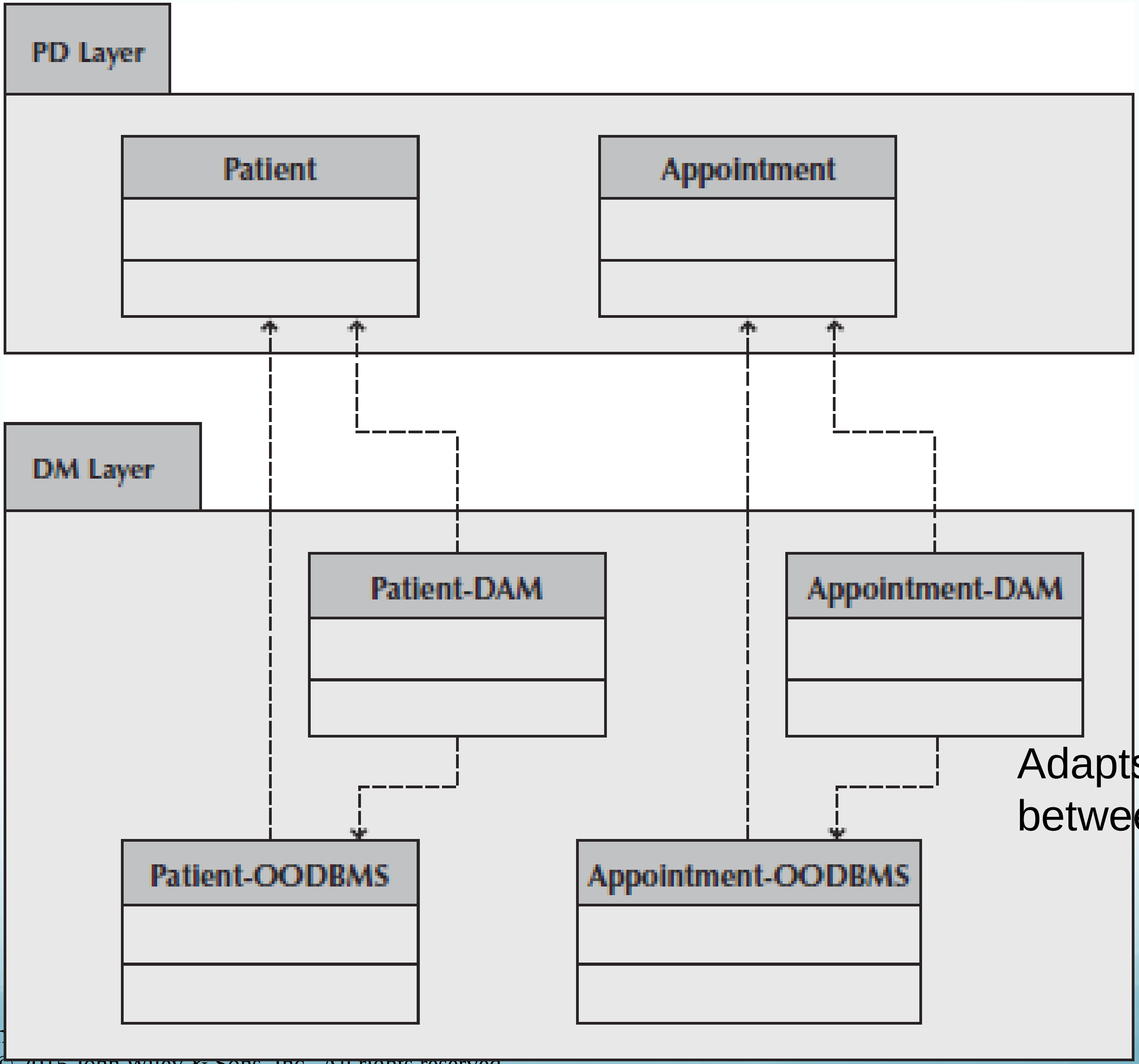
Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Mapping Problem-Domain Objects to Object-Persistence Formats

- Map objects to an OODBMS format
  - Each concrete class has a corresponding object persistence class
  - Add a data access and manipulation class to control the interaction
- Map objects to an ORDBMS format
  - Procedure depends on the level of support for object orientation by the ORDBMS
- Map objects to an RDBMS format

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Mapping to an OODBMS



Adapts between

WILEY
PowerPoin
Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Mapping to an OODBMS

May need to factor out multiple inheritance if not supported by OODBMS

Techniques similar to removing it from the design to write code when required:

Either change extra base class objects to attributes, or absorb their attributes.

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Mapping to an ORDBMS

Rule 1: Map all concrete Problem Domain classes to the ORDBMS tables. Also, if an abstract problem domain class has multiple direct subclasses, map the abstract class to an ORDBMS table.

Rule 2: Map single-valued attributes to columns of the ORDBMS tables.

Rule 3: Map methods and derived attributes to stored procedures or to program modules.

Rule 4: Map single-valued aggregation and association relationships to a column that can store an Object ID. Do this for both sides of the relationship.

Rule 5: Map multivalued attributes to a column that can contain a set of values.

Rule 6: Map repeating groups of attributes to a new table and create a one-to-many association from the original table to the new one.

Rule 7: Map multivalued aggregation and association relationships to a column that can store a set of Object IDs. Do this for both sides of the relationship.

Rule 8: For aggregation and association relationships of mixed type (one-to-many or many-to-one), on the single-valued side (1..1 or 0..1) of the relationship, add a column that can store a set of Object IDs. The values contained in this new column will be the Object IDs from the instances of the class on the multivalued side. On the multivalued side (1..* or 0..*), add a column that can store a single Object ID that will contain the value of the instance of the class on the single-valued side.

For generalization/inheritance relationships:

Rule 9a: Add a column(s) to the table(s) that represents the subclass(es) that will contain an Object ID of the instance stored in the table that represents the superclass. This is similar in concept to a foreign key in an RDBMS. The multiplicity of this new association from the subclass to the "superclass" should be 1..1. Add a column(s) to the table(s) that represents the superclass(es) that will contain an Object ID of the instance stored in the table that represents the subclass(es). If the superclasses are concrete, that is, they can be instantiated themselves, then the multiplicity from the superclass to the subclass is 0..1, otherwise, it is 1..1. An exclusive-or (XOR) constraint must be added between the associations. Do this for each superclass.

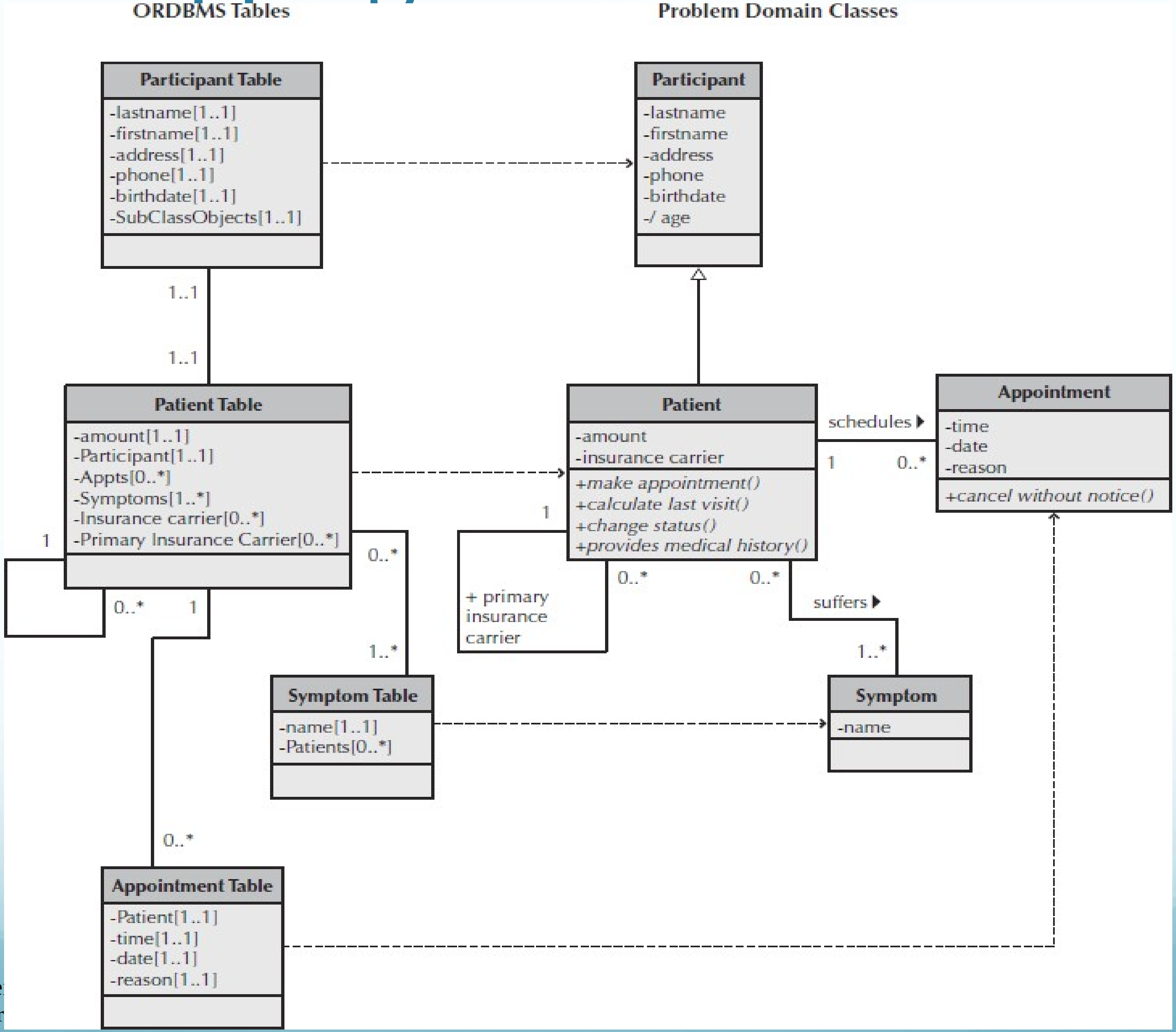
or

Rule 9b: Flatten the inheritance hierarchy by copying the superclass attributes down to all of the subclasses and remove the superclass from the design.*

*It is also a good idea to document this modification in the design so that in the future, modifications to the design can be maintained easily.

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Mapping to an ORDBMS



WILEY
Powe
Copyr

Mapping to an RDBMS

Rule 1: Map all concrete-problem domain classes to the RDBMS tables. Also, if an abstract Problem Domain class has multiple direct subclasses, map the abstract class to a RDBMS table.

Rule 2: Map single-valued attributes to columns of the tables.

Rule 3: Map methods to stored procedures or to program modules.

Rule 4: Map single-valued aggregation and association relationships to a column that can store the key of the related table, i.e., add a foreign key to the table. Do this for both sides of the relationship.

Rule 5: Map multivalued attributes and repeating groups to new tables and create a one-to-many association from the original table to the new ones.

Rule 6: Map multivalued aggregation and association relationships to a new associative table that relates the two original tables together. Copy the primary key from both original tables to the new associative table, i.e., add foreign keys to the table.

Rule 7: For aggregation and association relationships of mixed type, copy the primary key from the single-valued side (1..1 or 0..1) of the relationship to a new column in the table on the multivalued side (1..* or 0..*) of the relationship that can store the key of the related table, i.e., add a foreign key to the table on the multivalued side of the relationship.

For generalization/inheritance relationships:

Rule 8a: Ensure that the primary key of the subclass instance is the same as the primary key of the superclass. The multiplicity of this new association from the subclass to the "superclass" should be 1..1. If the superclasses are concrete, that is, they can be instantiated themselves, then the multiplicity from the superclass to the subclass is 0..1, otherwise, it is 1..1. Furthermore, an exclusive-or (XOR) constraint must be added between the associations. Do this for each superclass.

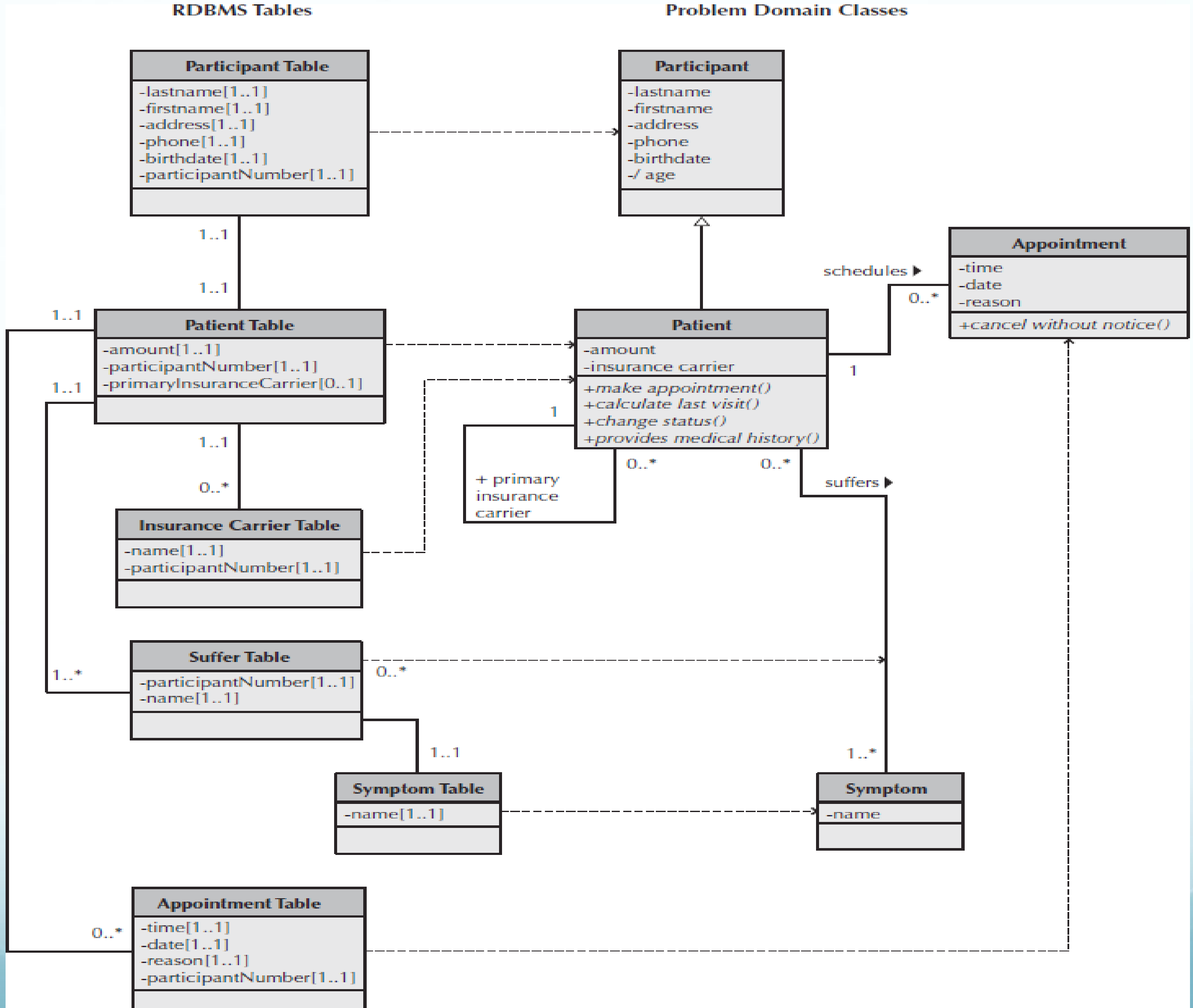
OR

Rule 8b: Flatten the inheritance hierarchy by copying the superclass attributes down to all of the subclasses and remove the superclass from the design.*

* It is also a good idea to document this modification in the design so that in the future, modifications to the design can be maintained easily.

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Mapping to an RDBMS



PowerPoint
Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Optimizing RDBMS-Based Object Storage

- Primary (often conflicting) dimensions:
  - Improve storage efficiency
    - Normalize the tables
    - Reduce redundant data and the occurrence of null values
  - Improve speed of access
    - De-normalize some tables to reduce processing time
    - Place similar records together (clustering)
    - Add indexes to quickly locate records

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Normalization

- Store each data fact only once in the database
- Reduces data redundancies and chances of errors
- First four levels of normalization are
  - 0 Normal Form: normalization rules not applied
  - 1 Normal Form: no multi-valued attributes (each cell has only a single value)
    - Eliminate fields which are arrays or repeated.
    - They can be new tables.

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Form 1 Violation

Order

-Order Number : unsigned long
-Date : Date
-Cust ID : unsigned long
-Last Name : String
-First Name : String
-State : String
-Tax Rate : float
-Product 1 Number : unsigned long
-Product 1 Desc. : String
-Product 1 Price : double
-Product 1 Qty. : unsigned long
-Product 2 Number : unsigned long
-Product 2 Desc. : String
-Product 2 Price : double
-Product 2 Qty. : unsigned long
-Product 3 Number : unsigned long
-Product 3 Desc. : String
-Product 3 Price : double
-Product 3 Qty. : unsigned long

Redundant Data

Null Cells

Sample Records:

Order Number	Date	Cust ID	Last Name	First Name	State	Tax Rate	Prod. 1 Number	Prod. 1 Desc.	Prod. 1 Price	Prod. 1 Qty.	Prod. 2 Number	Prod. 2 Desc.	Prod. 2 Price	Prod. 2 Qty.	Prod. 3 Number	Prod. 3 Desc.	Prod. 3 Price	Prod. 3 Qty.
239	11/23/00	1035	Black	John	MD	0.05	555	Cheese Tray	$45.00	2								
260	11/24/00	1035	Black	John	MD	0.05	444	Wine Gift Pack	$60.00	1								
273	11/27/00	1035	Black	John	MD	0.05	222	Bottle Opener	$12.00	1								
241	11/23/00	1123	Williams	Mary	CA	0.08	444	Wine Gift Pack	$60.00	2								
262	11/24/00	1123	Williams	Mary	CA	0.08	222	Bottle Opener	$12.00	2								
287	11/27/00	1123	Williams	Mary	CA	0.08	222	Bottle Opener	$12.00	2								
290	11/30/00	1123	Williams	Mary	CA	0.08	555	Cheese Tray	$45.00	3								
234	11/23/00	2242	DeBerry	Ann	DC	0.065	555	Cheese Tray	$45.00	2								
237	11/23/00	2242	DeBerry	Ann	DC	0.065	111	Wine Guide	$15.00	1	444	Wine Gift Pack	$60.00	1				
238	11/23/00	2242	DeBerry	Ann	DC	0.065	444	Wine Gift Pack	$60.00	1								
245	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1								
250	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1								
252	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1	444	Wine Gift Pack	$60.00	2				
253	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1	444	Wine Gift Pack	$60.00	1				
297	11/30/00	2242	DeBerry	Ann	DC	0.065	333	Jams & Jellies	$20.00	2								
243	11/24/00	4254	Bailey	Ryan	MD	0.05	555	Cheese Tray	$45.00	2								
246	11/24/00	4254	Bailey	Ryan	MD	0.05	333	Jams & Jellies	$20.00	3								
248	11/24/00	4254	Bailey	Ryan	MD	0.05	222	Bottle Opener	$12.00	1	333	Jams & Jellies	$20.00	2	111	Wine Guide	$15.00	1
235	11/23/00	9500	Chin	April	KS	0.05	222	Bottle Opener	$12.00	1								
242	11/23/00	9500	Chin	April	KS	0.05	333	Jams & Jellies	$20.00	3								
244	11/24/00	9500	Chin	April	KS	0.05	222	Bottle Opener	$12.00	2								
251	11/24/00	9500	Chin	April	KS	0.05	111	Wine Guide	$15.00	2								

WILEY

PowerPoint Presentation for Dennis, Wixom, & Tegarden *Systems Analysis and Design with UML, 5th Edition*
Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Normalization

- First four levels of normalization are
  - 2 Fields depend on whole primary key
    - New order: primary key is order and product number
    - Description depends only on product
    - New table for product info
  - 3 None of the fields depend on the primary key
    - No field depends on non-primary key.
    - Tax rate depends on state.
    - Add table of states and rates.

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Form 2 Violation

Order 237 has 2 products

Sample Records:

Order Table

Order Number	Date	Cust ID	Last Name	First Name	State	Tax Rate
239	11/23/00	1035	Black	John	MD	0.05
260	11/24/00	1035	Black	John	MD	0.05
273	11/27/00	1035	Black	John	MD	0.05
241	11/23/00	1123	Williams	Mary	CA	0.08
262	11/24/00	1123	Williams	Mary	CA	0.08
287	11/27/00	1123	Williams	Mary	CA	0.08
290	11/30/00	1123	Williams	Mary	CA	0.08
234	11/23/00	2242	DeBerry	Ann	DC	0.065
237	11/23/00	2242	DeBerry	Ann	DC	0.065
238	11/23/00	2242	DeBerry	Ann	DC	0.065
245	11/24/00	2242	DeBerry	Ann	DC	0.065
250	11/24/00	2242	DeBerry	Ann	DC	0.065
252	11/24/00	2242	DeBerry	Ann	DC	0.065
253	11/24/00	2242	DeBerry	Ann	DC	0.065
297	11/30/00	2242	DeBerry	Ann	DC	0.065
243	11/24/00	4254	Bailey	Ryan	MD	0.05
246	11/24/00	4254	Bailey	Ryan	MD	0.05
248	11/24/00	4254	Bailey	Ryan	MD	0.05
235	11/23/00	9500	Chin	April	KS	0.05
242	11/23/00	9500	Chin	April	KS	0.05
244	11/24/00	9500	Chin	April	KS	0.05
251	11/24/00	9500	Chin	April	KS	0.05

Product Order Table

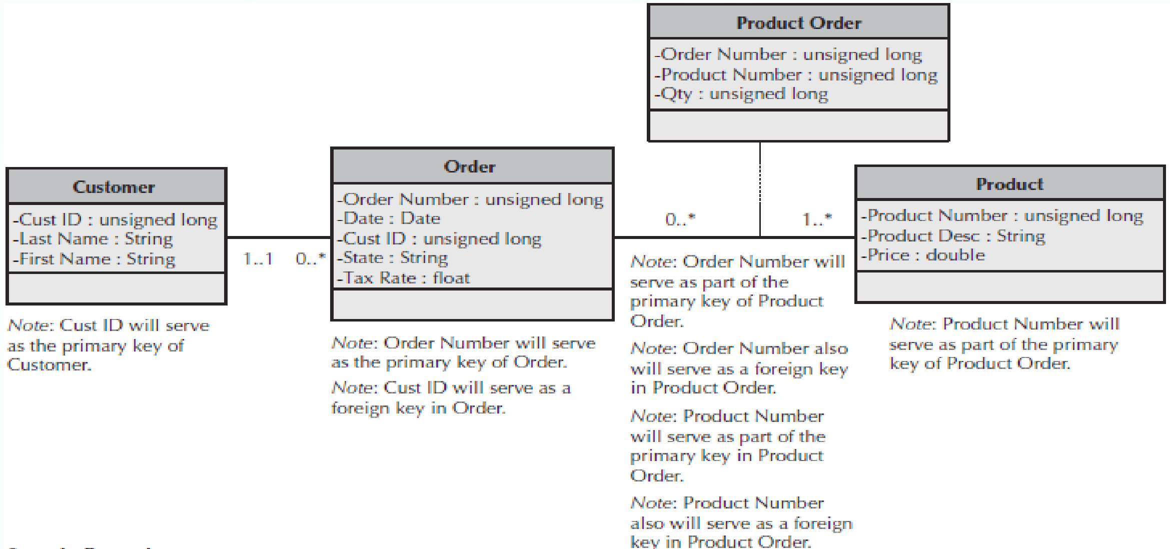
Order Number	Product Number	Product Desc	Product Price	Product Qty
239	555	Cheese Tray	$45.00	2
260	444	Wine Gift Pack	$60.00	1
273	222	Bottle Opener	$12.00	1
241	444	Wine Gift Pack	$60.00	2
262	222	Bottle Opener	$12.00	2
287	222	Bottle Opener	$12.00	2
290	555	Cheese Tray	$45.00	3
234	555	Cheese Tray	$45.00	2
237	111	Wine Guide	$15.00	1
237	444	Wine Gift Pack	$60.00	1
238	444	Wine Gift Pack	$60.00	1
245	222	Bottle Opener	$12.00	1
250	222	Bottle Opener	$12.00	1
252	222	Bottle Opener	$12.00	1
252	444	Wine Gift Pack	$60.00	2
253	222	Bottle Opener	$12.00	1
253	444	Wine Gift Pack	$60.00	1
297	333	Jams & Jellies	$20.00	2
243	555	Cheese Tray	$45.00	2
246	333	Jams & Jellies	$20.00	3
248	222	Bottle Opener	$12.00	1
248	333	Jams & Jellies	$20.00	2
248	111	Wine Guide	$15.00	1
235	222	Bottle Opener	$12.00	1
242	333	Jams & Jellies	$20.00	3
244	222	Bottle Opener	$12.00	2
251	111	Wine Guide	$15.00	2

Order 248 has 3 products

(b)

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Form 3 Violation



Sample Records:

Customer Table

Cust ID	Last Name	First Name
1035	Black	John
1123	Williams	Mary
2242	DeBerry	Ann
4254	Bailey	Ryan
9500	Chin	April

Last Name and First Name was moved to the Customer table to eliminate redundancy

Order Table

Order Number	Date	Cust ID	State	Tax Rate
239	11/23/00	1035	MD	0.05
260	11/24/00	1035	MD	0.05
273	11/27/00	1035	MD	0.05
241	11/23/00	1123	CA	0.08
262	11/24/00	1123	CA	0.08
287	11/27/00	1123	CA	0.08
290	11/30/00	1123	CA	0.08
234	11/23/00	2242	DC	0.065
237	11/23/00	2242	DC	0.065
238	11/23/00	2242	DC	0.065
245	11/24/00	2242	DC	0.065
250	11/24/00	2242	DC	0.065
252	11/24/00	2242	DC	0.065
253	11/24/00	2242	DC	0.065
297	11/30/00	2242	DC	0.065
243	11/24/00	4254	MD	0.05
246	11/24/00	4254	MD	0.05
248	11/24/00	4254	MD	0.05
235	11/23/00	9500	KS	0.05
242	11/23/00	9500	KS	0.05
244	11/24/00	9500	KS	0.05
251	11/24/00	9500	KS	0.05

Product Order Table

Order Number	Product Number	Product Qty
239	555	2
260	444	1
273	222	1
241	444	2
262	222	2
287	222	2
290	555	3
234	555	2
237	111	1
237	444	1
238	444	1
245	222	1
250	222	1
252	222	1
252	444	2
253	222	1
253	444	1
297	333	2
243	555	2
246	333	3
248	222	1
248	333	2
248	111	1
235	222	1
242	333	3
244	222	2
251	111	2

Product Table

Product Number	Product Desc	Product Price
111	Wine Guide	$15.00
222	Bottle Opener	$12.00
333	Jams & Jellies	$20.00
444	Wine Gift Pack	$60.00
555	Cheese Tray	$45.00

Product Desc and Price was moved to the Product table to eliminate redundancy

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Steps of Normalization

0 Normal Form

Do any tables have repeating fields? Do some records have a different number of columns from other records?	Yes: Remove the repeating fields. Add a new table that contains the fields that repeat.
	No: The data model is in 1NF

First Normal Form

Is the primary key made up of more than one field? If so, do any fields depend on only a part of the primary key?	Yes: Remove the partial dependency. Add a new table that contains the fields that are partially dependent.
	No: The data model is in 2NF

Second Normal Form

Do any fields depend on another nonprimary key field?	Yes: Remove the transitive dependency. Add a new table that contains the fields that are transitively dependent.
	No: The data model is in 3NF

Third Normal Form

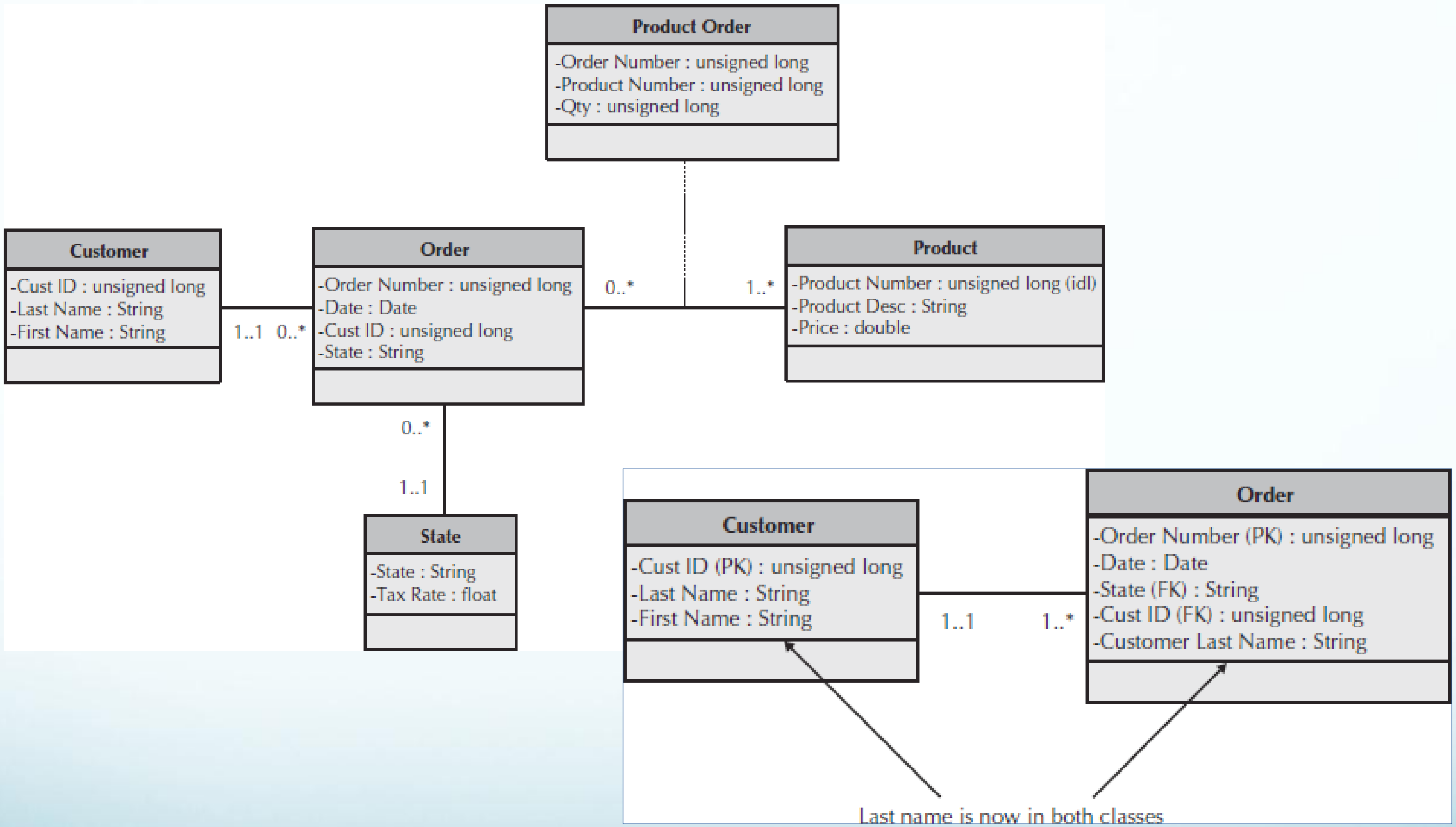
Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Optimizing Data Access Speed

- De-normalization
  - Table joins require processing
  - Add some data to a table to reduce the number of joins required (Increases data retrieval speed)
  - Creates redundancy and should be used sparingly
- Clustering
  - Place similar records close together on the disk
  - Reduces the time needed to access the disk
  - Ordering records in a table?

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Denormalization



Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Optimizing Data Access Speed (cont.)

- Indexing
  - A small file with attribute values and a pointer to the record on the disk
  - Search the index file for an entry, then go to the disk to retrieve the record
  - Accessing a file in memory is much faster than searching a disk
  - Adds overhead to update; most efficient when lookup >> update.

Use indexes sparingly for transaction systems.
Use many indexes to increase response times in decision support systems.
For each table, create a unique index that is based on the primary key.
For each table, create an index that is based on the foreign key to improve the performance of joins.
Create an index for fields that are used frequently for grouping, sorting, or criteria.

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Optimizing Data Access Storage

- Estimating Data Storage Size
  - Use volumetrics to estimate

amount of raw data + overhead requirements

  - This helps determine the necessary hardware capacity

Field	Average Size
Order Number	8
Date	7
Cust ID	4
Last Name	13
First Name	9
State	2
Amount	4
Tax Rate	2
Record Size	49
Overhead	30%
Total Record Size	63.7
Initial Table Size	50,000
Initial Table Volume	3,185,000
Growth Rate/Month	1,000
Table Volume @ 3 years	5,478,200

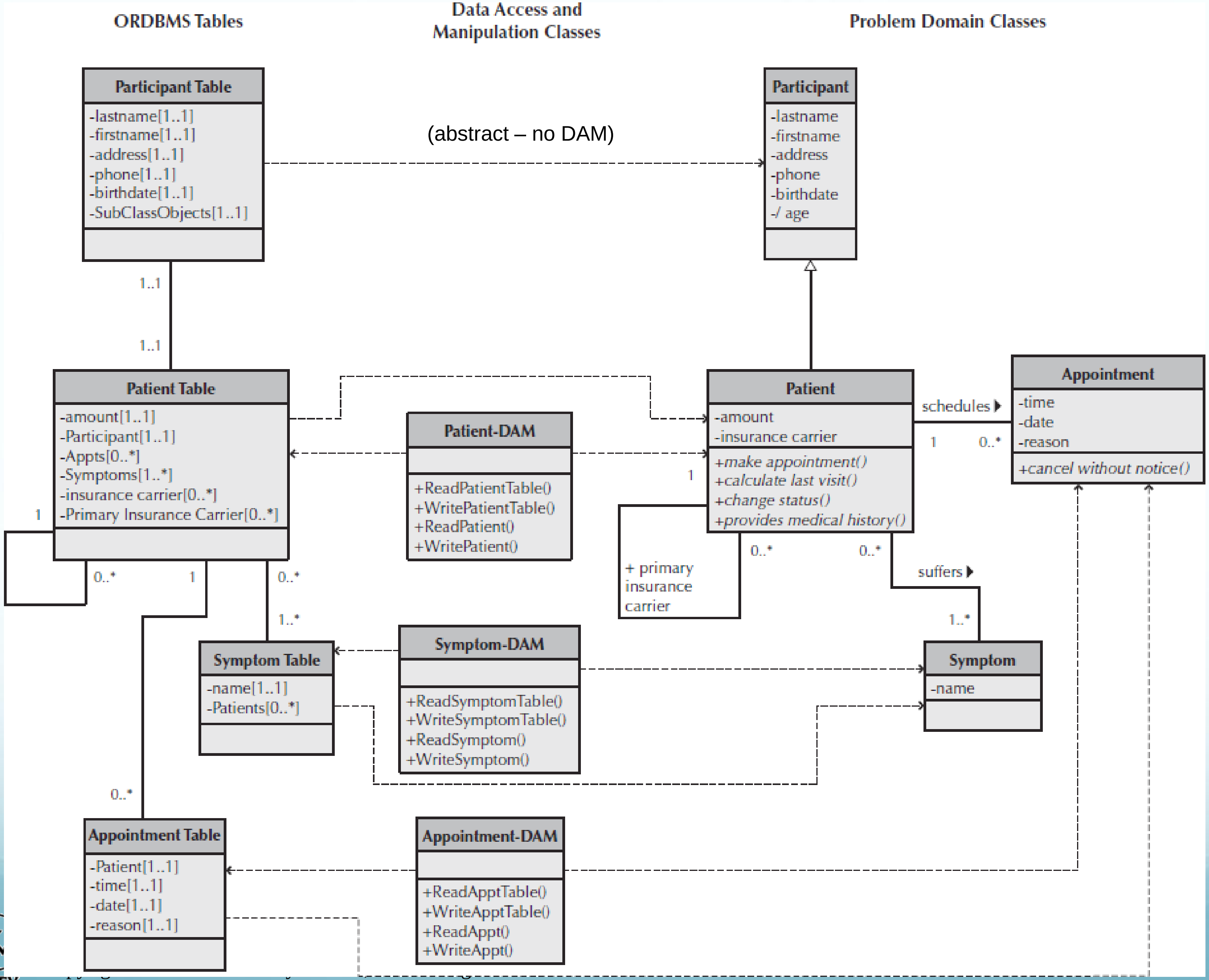
Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Designing Data Access & Manipulation Classes

- Classes that translate between the problem domain classes and object persistent classes
- ORDBMS: create one DAM for each concrete PD class
- RDBMS: may require more classes since data is spread over more tables
  - Class libraries (e.g., Hibernate) are available to help

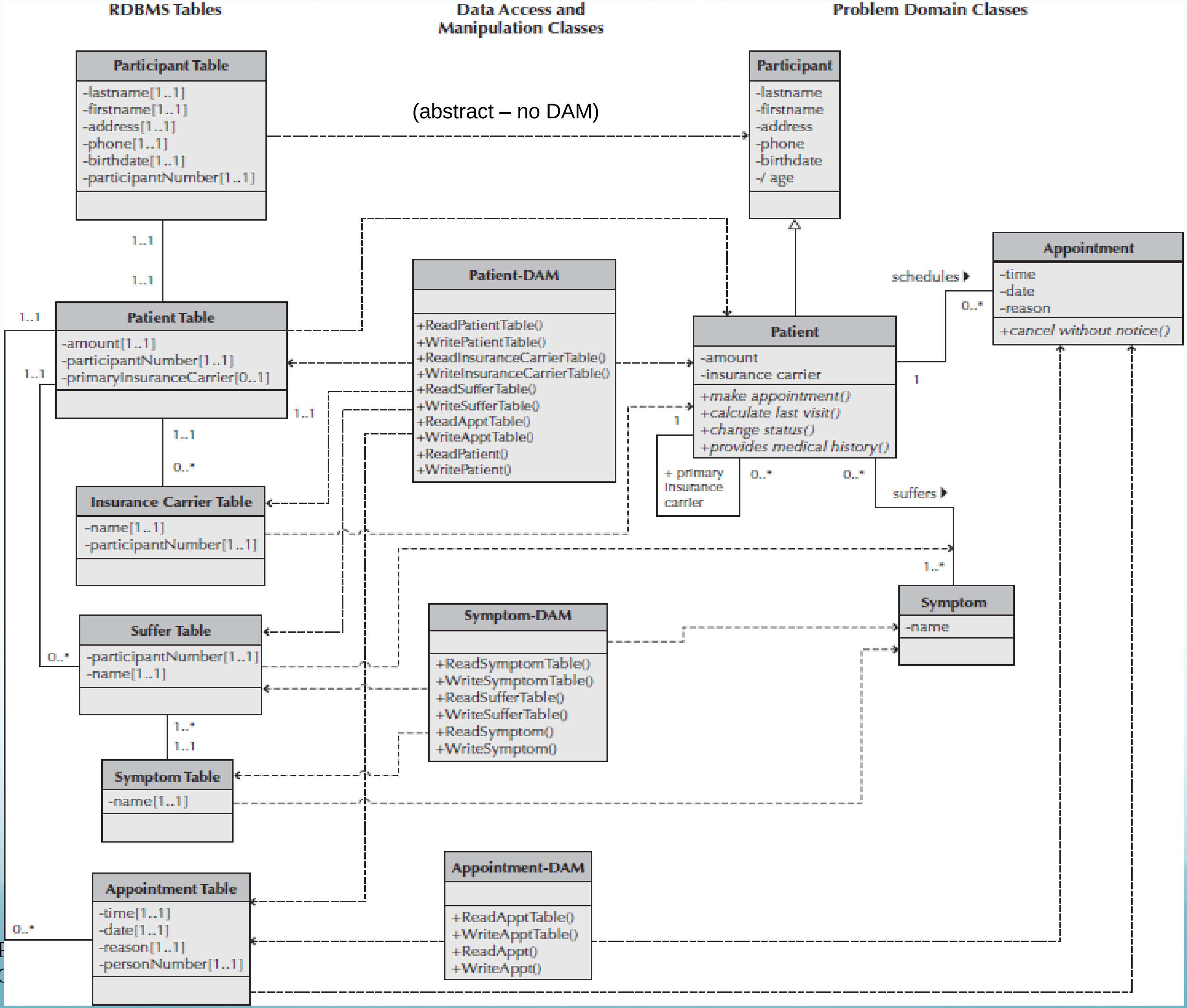
Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

ORDBMS DAM classes



WILEY

RDBMS DAM classes



Nonfunctional Requirements & Data Management Layer Design

- Operational requirements: affected by choice in hardware and operating system
- Performance requirements: speed & capacity issues
- Security requirements: access controls, encryption, and backup
- Cultural & political requirements: may affect the data storage (e.g., expected number of characters for data field, required format of a data field, local laws pertaining to data storage, etc…)

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

VERIFYING AND VALIDATING THE DATA MANAGEMENT LAYER

- Test the fidelity of the design before implementation
- Verifying and validating the design of the data management layer falls into three basic groups:
  1. Verifying and validating any changes made to the problem domain
  2. Dependency of the object persistence instances on the problem domain must be enforced
  3. The design of the data access and manipulation classes need to be tested

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.

Summary

- Object Persistence Formats
- Mapping Problem-Domain Objects to Object-Persistence Formats
- Optimizing RDBMS-Based Object Storage
- Nonfunctional Requirements and Data Management Layer Design
- Designing Data Access and Manipulation Classes
- Nonfunctional Requirements & Data Management Layer Design
- Verifying and validating the data management layer

Copyright © 2015 John Wiley & Sons, Inc. All rights reserved.