Systems Analysis and Design With UML 2.0

An Object-Oriented Approach, Second Edition

Chapter 11: Data Management Layer Design

Alan Dennis, Barbara Wixom, and David Tegarden

WILEY

Copyright © 2005 John Wiley & Sons, Inc.

Data Management Layer Design

Chapter 11

WILEY

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.
- Be able to design the data access and manipulation classes.

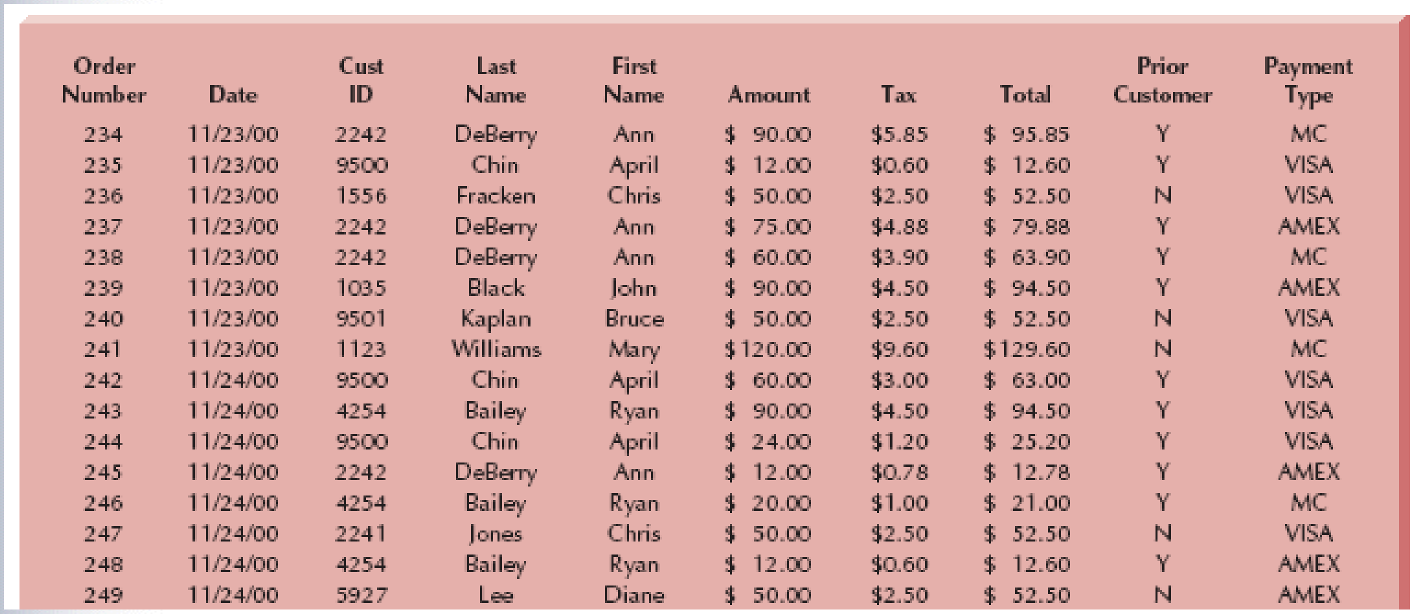
Data Management Layer

- Choose object-persistence format to support the system
  - Problem domain objects drive object storage design
- Design of Data Storage
  - Must optimize processing efficiency
- Data access and manipulation
  - Separate problem domain classes from storage format
  - Handle all communication with the database

Object Persistence Formats

- Files (Sequential and Random)
- Relational databases
- Object-relational databases
- Object-oriented databases

Customer Order File



Order Number	Date	Cust ID	Last Name	First Name	Amount	Tax	Total	Prior Customer	Payment Type
234	11/23/00	2242	DeBerry	Ann	$ 90.00	$5.85	$ 95.85	Y	MC
235	11/23/00	9500	Chin	April	$ 12.00	$0.60	$ 12.60	Y	VISA
236	11/23/00	1556	Fracken	Chris	$ 50.00	$2.50	$ 52.50	N	VISA
237	11/23/00	2242	DeBerry	Ann	$ 75.00	$4.88	$ 79.88	Y	AMEX
238	11/23/00	2242	DeBerry	Ann	$ 60.00	$3.90	$ 63.90	Y	MC
239	11/23/00	1035	Black	John	$ 90.00	$4.50	$ 94.50	Y	AMEX
240	11/23/00	9501	Kaplan	Bruce	$ 50.00	$2.50	$ 52.50	N	VISA
241	11/23/00	1123	Williams	Mary	$120.00	$9.60	$129.60	N	MC
242	11/24/00	9500	Chin	April	$ 60.00	$3.00	$ 63.00	Y	VISA
243	11/24/00	4254	Bailey	Ryan	$ 90.00	$4.50	$ 94.50	Y	VISA
244	11/24/00	9500	Chin	April	$ 24.00	$1.20	$ 25.20	Y	VISA
245	11/24/00	2242	DeBerry	Ann	$ 12.00	$0.78	$ 12.78	Y	AMEX
246	11/24/00	4254	Bailey	Ryan	$ 20.00	$1.00	$ 21.00	Y	MC
247	11/24/00	2241	Jones	Chris	$ 50.00	$2.50	$ 52.50	N	VISA
248	11/24/00	4254	Bailey	Ryan	$ 12.00	$0.60	$ 12.60	Y	AMEX
249	11/24/00	5927	Lee	Diane	$ 50.00	$2.50	$ 52.50	N	AMEX

Sequential and Random Access Files

- Sequential access files allow sequential operations
  - Read, write, and search
  - Efficient for report writing
  - Searches are not efficient because an average of 50% of records have to be accessed
  - Two versions
    - Ordered
    - unordered

Random Access Files

- Allow only random or direct file operations
- Good for finding and updating a specific object
- Inefficient report writing

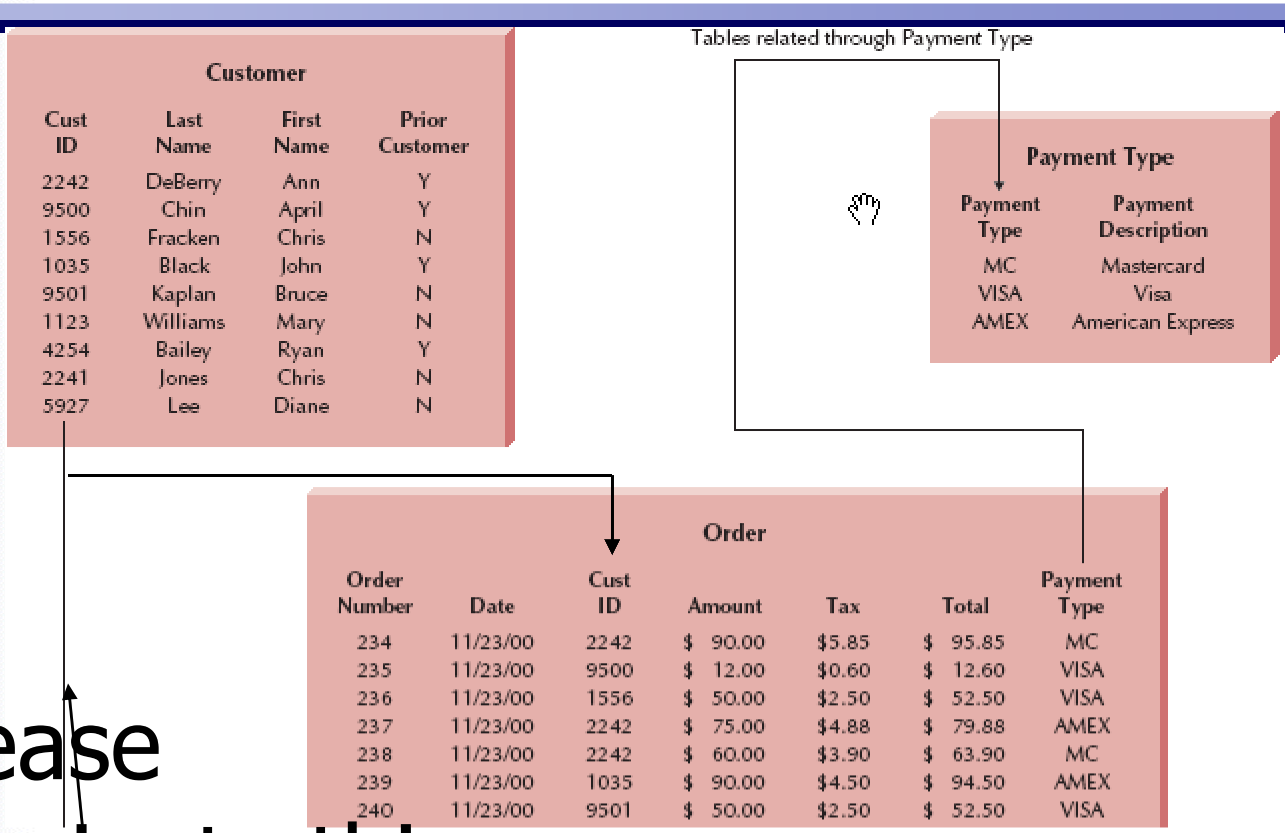
Application File Types

- Master Files
- Look-up files
- Transaction files
- Audit file
- History file

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

Customer Order Database



Tables related through Payment Type

Customer

Cust ID	Last Name	First Name	Prior Customer
2242	DeBerry	Ann	Y
9500	Chin	April	Y
1556	Fracken	Chris	N
1035	Black	John	Y
9501	Kaplan	Bruce	N
1123	Williams	Mary	N
4254	Bailey	Ryan	Y
2241	Jones	Chris	N
5927	Lee	Diane	N

Payment Type

Payment Type	Payment Description
MC	Mastercard
VISA	Visa
AMEX	American Express

Order

Order Number	Date	Cust ID	Amount	Tax	Total	Payment Type
234	11/23/00	2242	$ 90.00	$5.85	$ 95.85	MC
235	11/23/00	9500	$ 12.00	$0.60	$ 12.60	VISA
236	11/23/00	1556	$ 50.00	$2.50	$ 52.50	VISA
237	11/23/00	2242	$ 75.00	$4.88	$ 79.88	AMEX
238	11/23/00	2242	$ 60.00	$3.90	$ 63.90	MC
239	11/23/00	1035	$ 90.00	$4.50	$ 94.50	AMEX
240	11/23/00	9501	$ 50.00	$2.50	$ 52.50	VISA

Please Eliminate this line

Database Management System (DBMS)

- Software that creates and manipulates a database
- RDBMS is a DBMS for a relational database
- RDBMS usually support Referential Integrity

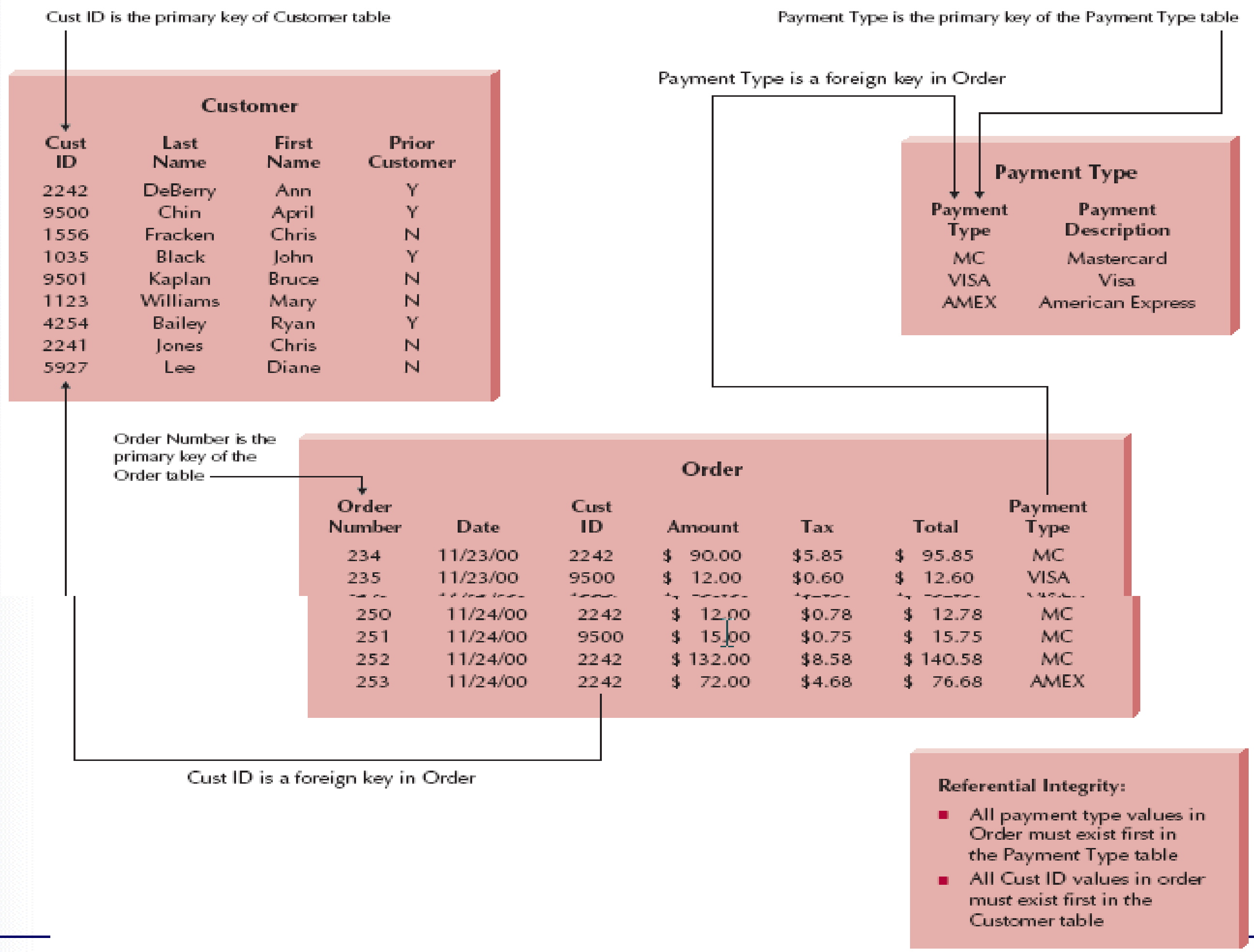
Referential Integrity

- 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



Cust ID is the primary key of Customer table

Payment Type is the primary key of the Payment Type table

Payment Type is a foreign key in Order

Customer

Cust ID	Last Name	First Name	Prior Customer
2242	DeBerry	Ann	Y
9500	Chin	April	Y
1556	Fracken	Chris	N
1035	Black	John	Y
9501	Kaplan	Bruce	N
1123	Williams	Mary	N
4254	Bailey	Ryan	Y
2241	Jones	Chris	N
5927	Lee	Diane	N

Payment Type

Payment Type	Payment Description
MC	Mastercard
VISA	Visa
AMEX	American Express

Order Number is the primary key of the Order table

Order

Order Number	Date	Cust ID	Amount	Tax	Total	Payment Type
234	11/23/00	2242	$ 90.00	$5.85	$ 95.85	MC
235	11/23/00	9500	$ 12.00	$0.60	$ 12.60	VISA
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
250	11/24/00	2242	$ 12.00	$0.78	$ 12.78	MC
251	11/24/00	9500	$ 15.00	$0.75	$ 15.75	MC
252	11/24/00	2242	$ 132.00	$8.58	$ 140.58	MC
253	11/24/00	2242	$ 72.00	$4.68	$ 76.68	AMEX

Cust ID is a foreign key in Order

Referential Integrity:

- All payment type values in Order must exist first in the Payment Type table
- All Cust ID values in order must exist first in the Customer table

Structured Query Language (SQL)

- Standard language for accessing data in tables
- SQL Commands
  - Create, edit, and delete tables
  - Add, edit, and delete data
  - Display data from one or more related tables
  - Display data computed from data in one or more related tables

Object-Relational Databases

- Relational database management systems with extensions that handle object storage in the relational table structure
- This is done by user defined types
  - Example: Create a map data type

Vendors Support ORDBMS

- SQL designed for simple data types
- Vendors extend SQL to handle user data types in Object Relational Databases
- Usually they don’t support most object oriented features e.g. inheritance

Object-Oriented Databases (OODBMS)

- Add persistence extensions to an object-oriented programming language
- Create a entirely separate database management system

OODBMS Terminology

- Extent is a collection of objects
  - Set of instances associated with a particular class (RDBMS table)
  - Each instance of a class has a unique identifier called an object ID
  - Referential integrity still important
  - Supports a form of inheritance

OODBMS Support

- Allow repeating groups or multivalued attributes
- Supports multimedia or other complex data applications
  - CAD/CAM
  - Financial services
  - Geographic information systems
  - Health care

Major Strengths & Weaknesses

- Files
  - Very efficient for given task
  - Manipulation done by OOPL
  - Redundant data usually results
- RDBMS
  - Proven commercial technology
  - Handle diverse data
  - No support for object orientation

More Strengths and Weaknesses

- ORDBMS
  - Inherit RDBMS strengths
  - Support complex data types
  - Limited support for object-orientation (vendor dependent)
- OODBMS
  - Support complex data types
  - Support object-orientation directly
  - Still maturing (lacks skilled labor and may have a steep learning curve)

Criteria for Object Persistence Formats

- Data types supported
- Types of application systems (transaction processing, DSS, …)
- Existing Storage Formats
- Future Needs
- Other miscellaneous Criteria (cost, concurrency control, …)

Mapping Objects to Object-Persistence Formats

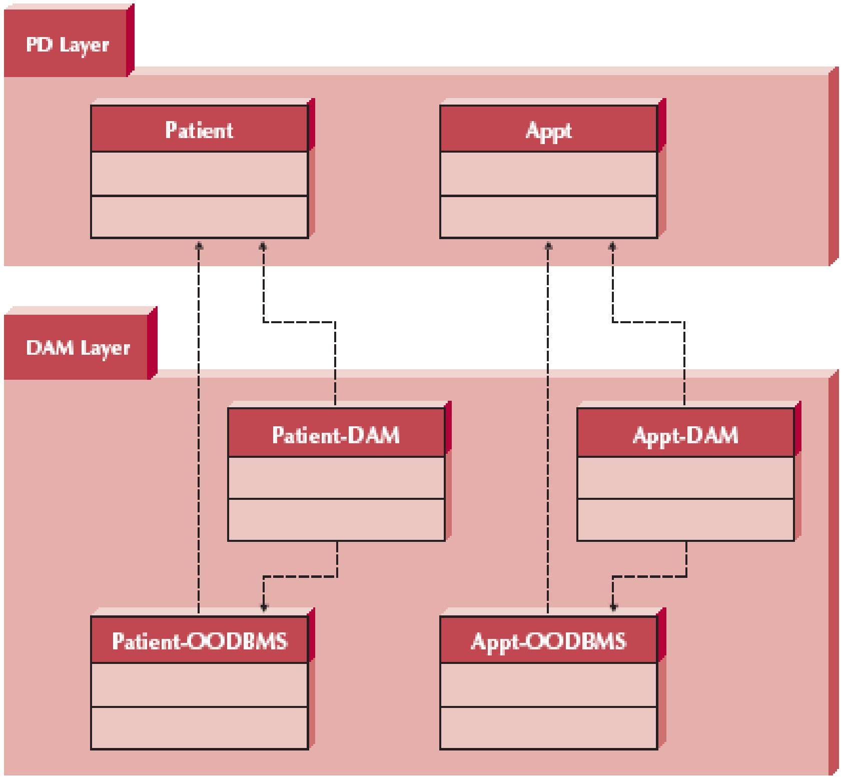


FIGURE 11-5
Appointment System Problem Domain and Data Access and Management Layers

Multiple Inheritance Effect Rules

- Results when you have more than one super class
- Rule 1a. Add an attribute(s) to the OODBMS class to represent the additional super class
- Rule 1b. Flatten the inheritance hierarchy and remove additional super classes from the design

Mapping to Single I-B OODBMS

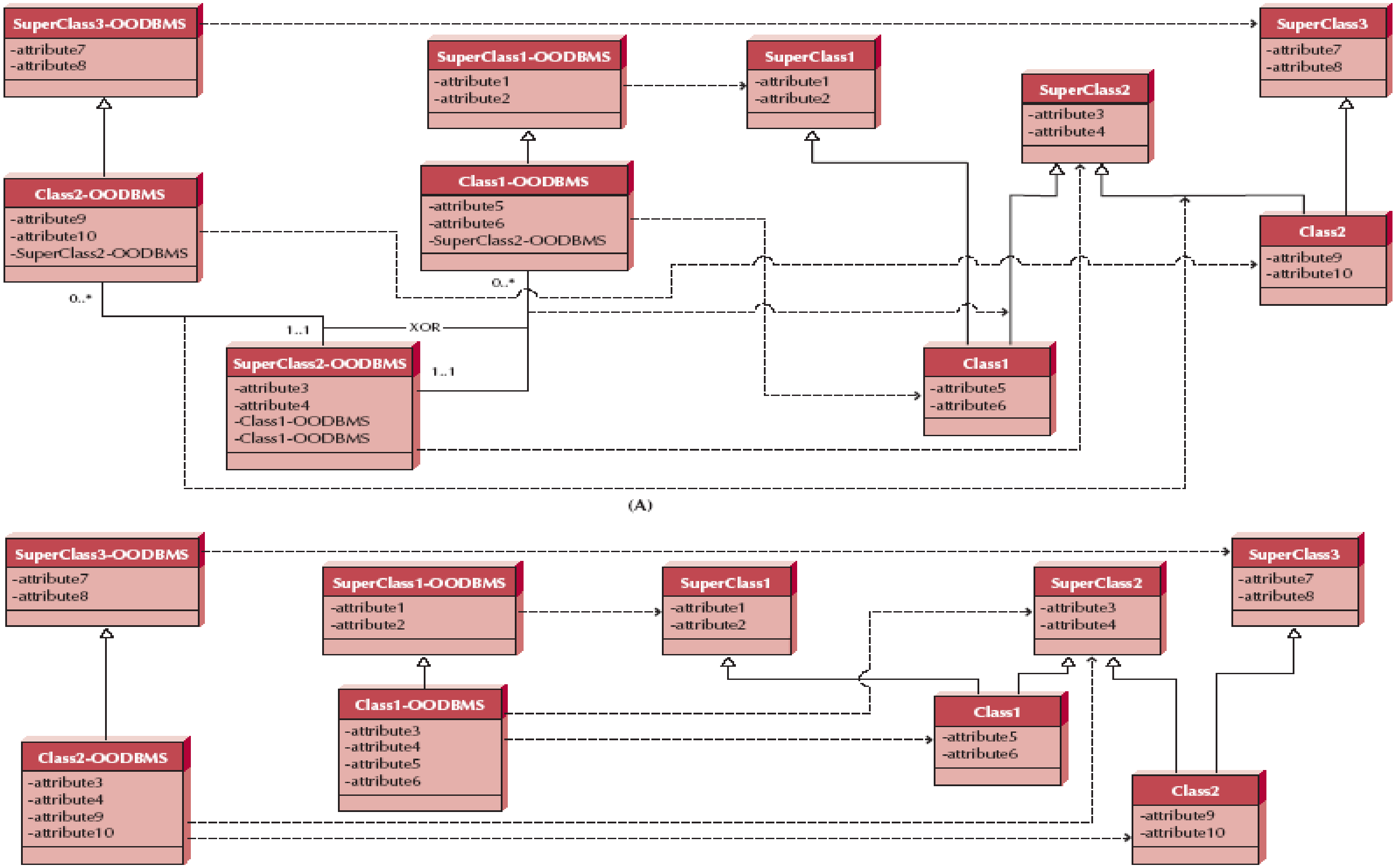
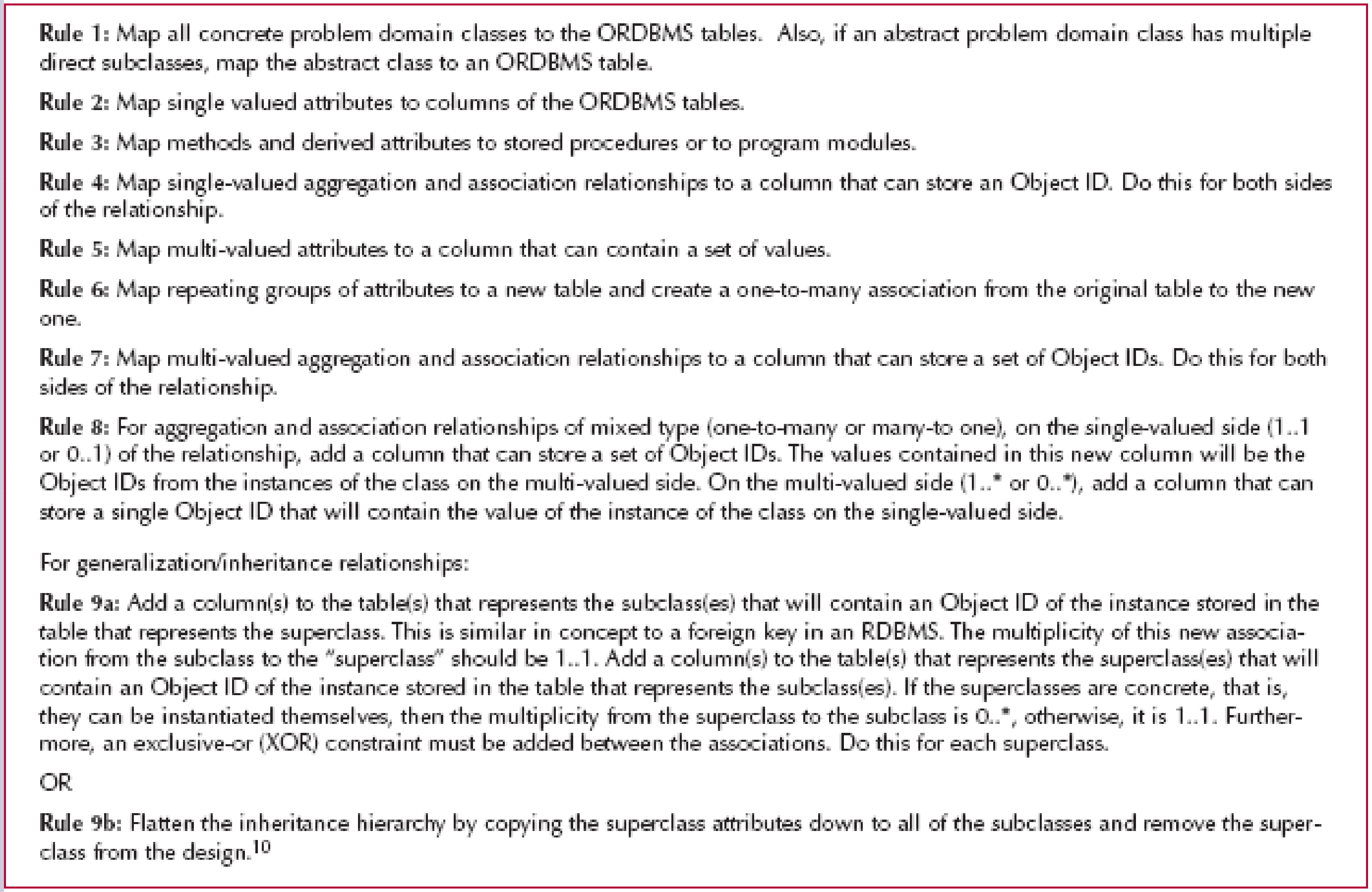


FIGURE 11-6 Mapping Problem Domain Objects to Single Inheritance-Based OODBMS

Using Rule 1a

- Added an attribute to Class1-OODBMS that represents an association with Super-Class2-OODBMS,
- Added attributes to Class2-OODBMS that represents an association with Super-Class2-OODBMS,
- Added a pair of attributes to SuperClass2-OODBMS that represents an association with Class1-OODBMS and Class2-OODBMS, for completeness sale, and
- Added associations between Class2-OODBMS and SuperClass2-OODBMS and Class1-OODBMS and SuperClass2-OODBMS that have the correct multiplicities and the XOR constraint explicitly shown.

Mapping PDO to 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 multi-valued 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 multi-valued 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 multi-valued side. On the multi-valued 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..*, otherwise, it is 1..1. Furthermore, 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.[10]

FIGURE 11-7 Mapping Problem Domain Objects to ORDBMS Schema

Mapping Table to PD Classes

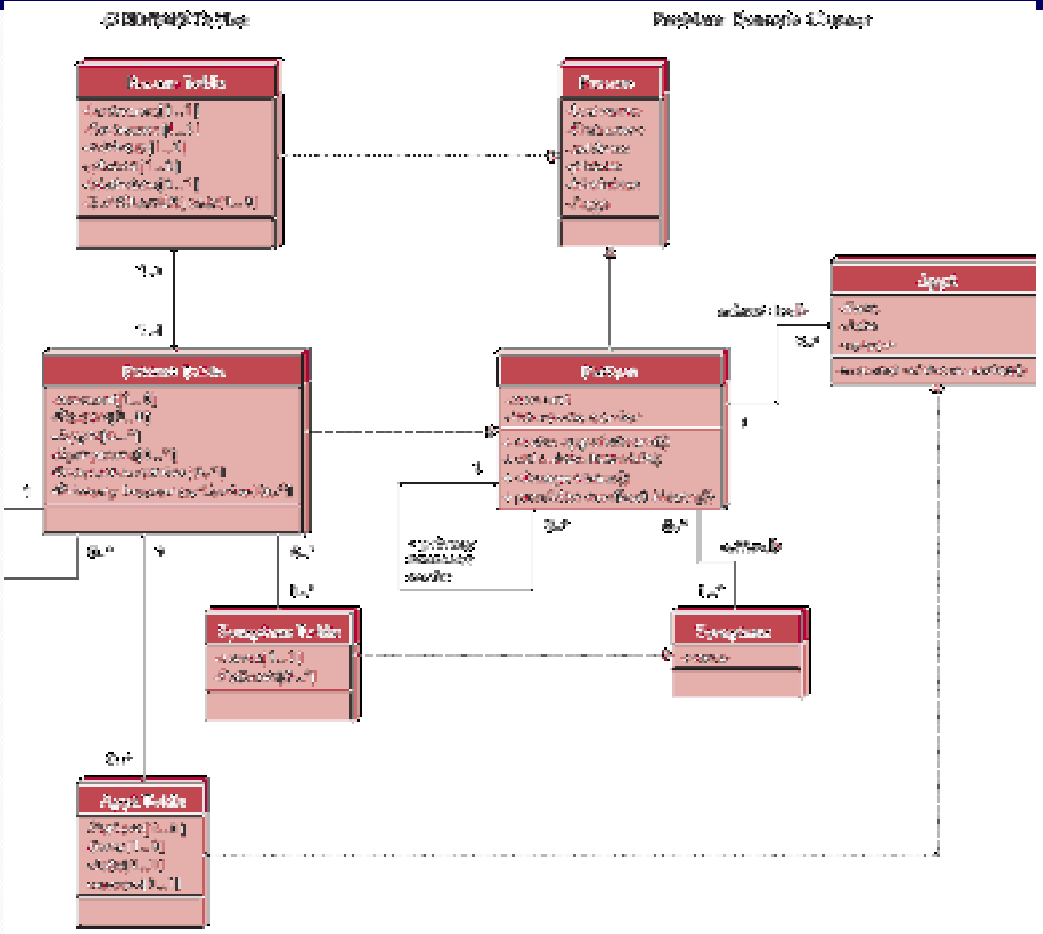


FIGURE 11-2 Mapping Problem-domain Objects to ORDBMS Schema Example

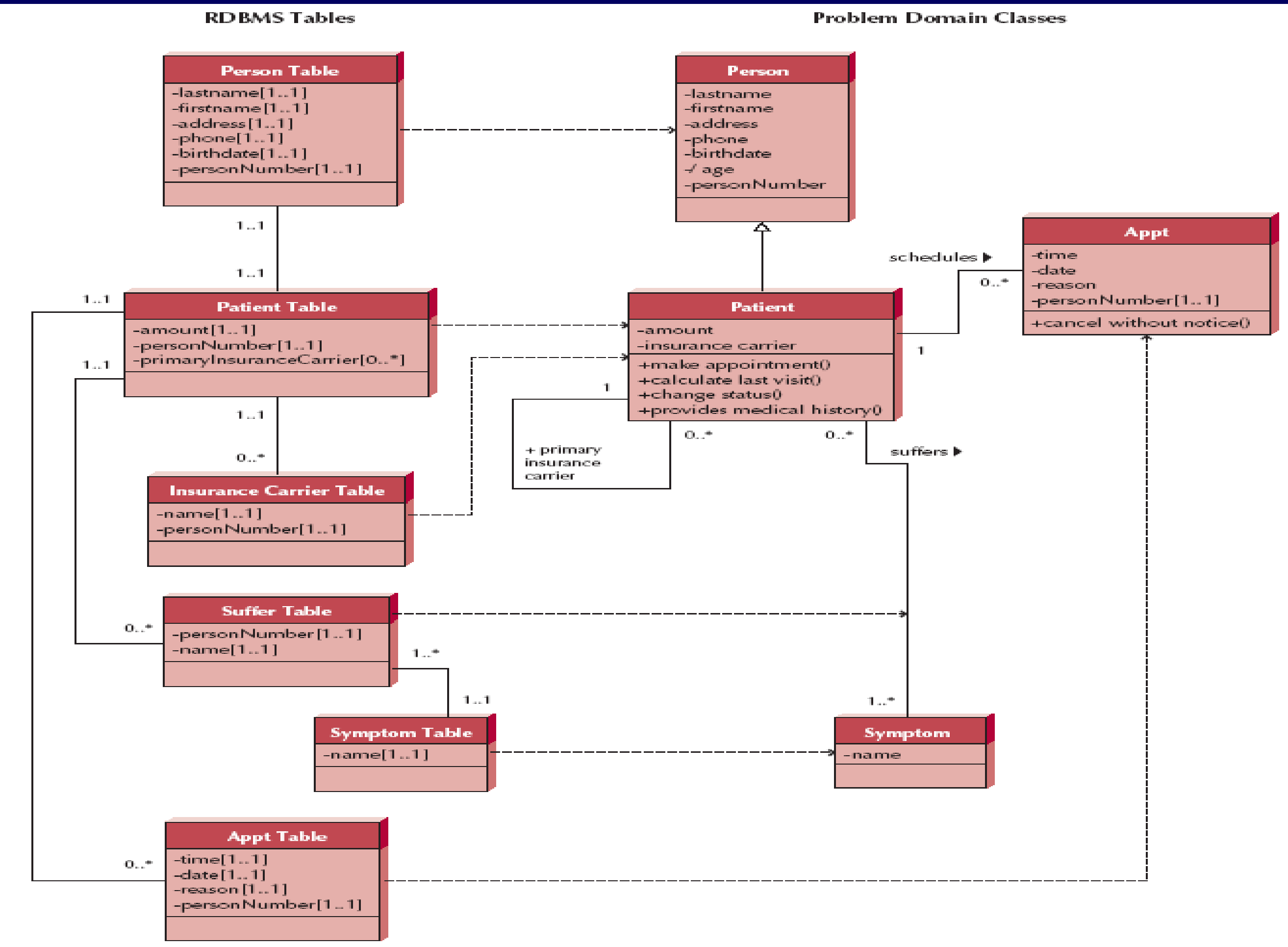
Mapping PD Objects to RDBMS Schema

- **Rule 1:** Map all concrete problem domain classes to the RDBMS tables.
- **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
- **Rule 5:** Map multi-valued 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 multi-valued 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
- **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 multi-valued side (1..* or 0..*) of the relationship that can store the key of
- the related table
- **Rule 8a:** Ensure that the primary key of the subclass instance is the same as the primary key of the superclass..

OR

- **Rule 8b:** Flatten the inheritance

Mapping RDBMS Tables to Problem Domain Classes



RDBMS Tables

Problem Domain Classes

Person Table
-lastname[1..1]
-firstname[1..1]
-address[1..1]
-phone[1..1]
-birthdate[1..1]
-personNumber[1..1]

Person
-lastname
-firstname
-address
-phone
-birthdate
-/ age
-personNumber

Patient Table
-amount[1..1]
-personNumber[1..1]
-primaryInsuranceCarrier[0..*]

Patient
-amount
-insurance carrier
+make appointment()
+calculate last visit()
+change status()
+provides medical history()

Appt
-time
-date
-reason
-personNumber[1..1]
+cancel without notice()

Insurance Carrier Table
-name[1..1]
-personNumber[1..1]

Suffer Table
-personNumber[1..1]
-name[1..1]

Symptom Table
-name[1..1]

Symptom
-name

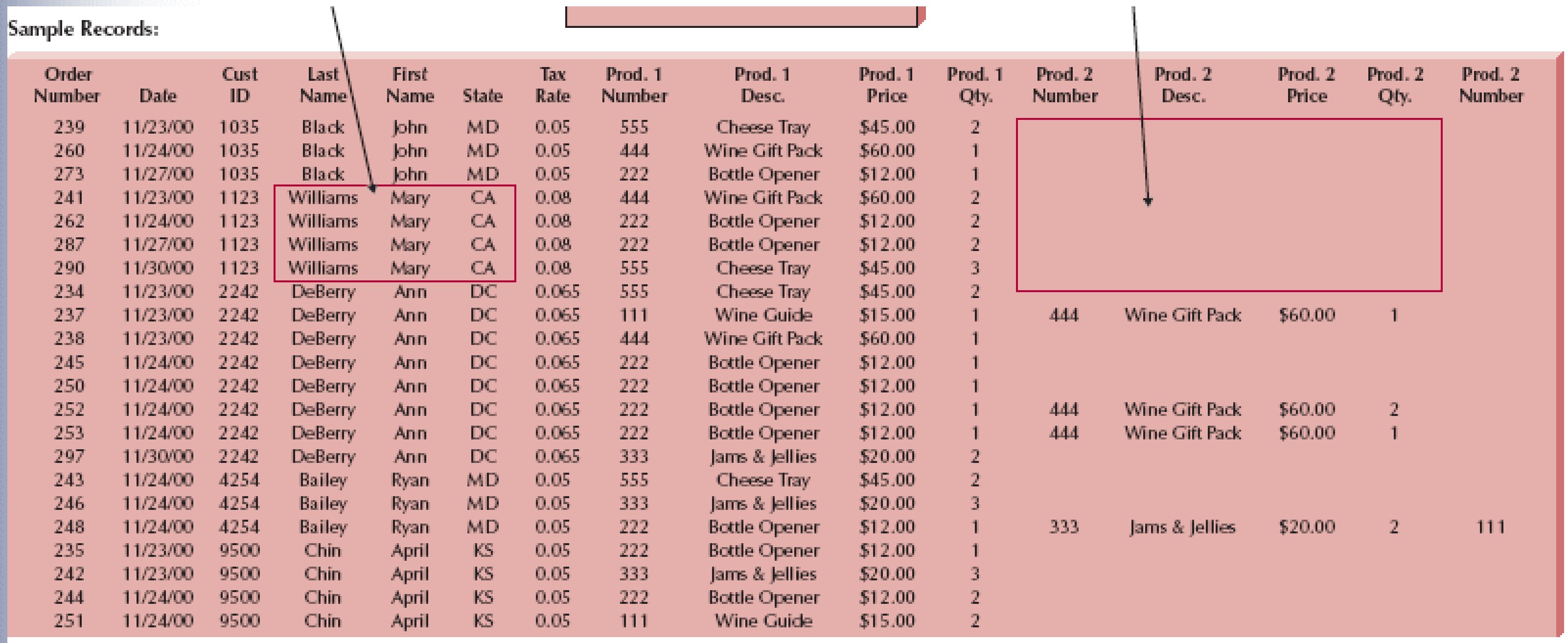
Appt Table
-time[1..1]
-date[1..1]
-reason [1..1]
-personNumber[1..1]

FIGURE 11-10 Mapping Problem Domain Objects to RDBMS Schema Example

Optimize RDBMS Object Storage

- No redundant data
  - Wastes space
  - Allow more room for error
- Few null values in tables
  - Difficult to interpret

Example of Non-normalized Data

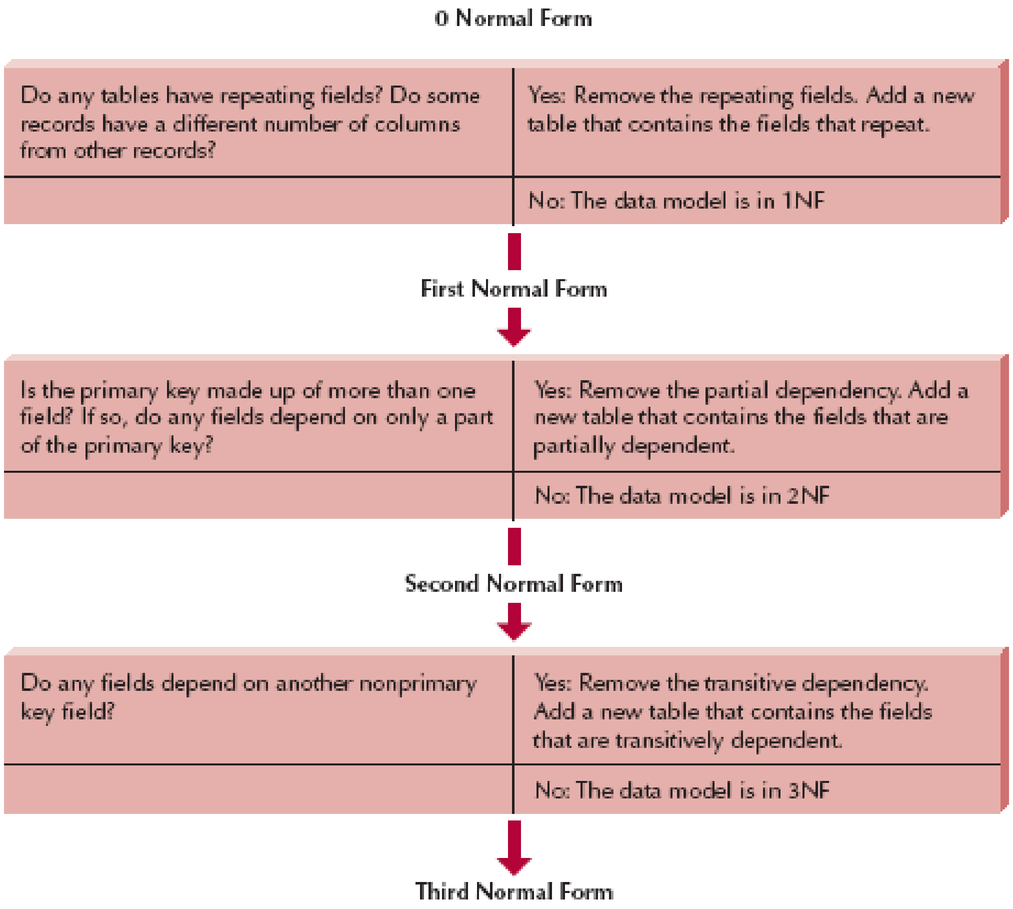


Sample Records:

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. 2 Number
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
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					

FIGURE 11-11 Optimizing Storage

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

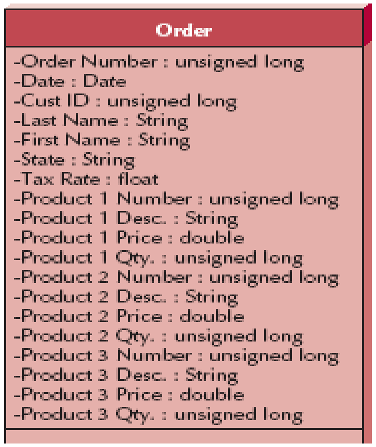
Do any fields depend on another nonprimary key field?	Yes: Remove the transitive dependency. Add a new table that contains the fields that are transitively dependent.
	No: The data model is in 3NF

Third Normal Form

FIGURE 11-12
The Steps of Normalization

Normalization Example

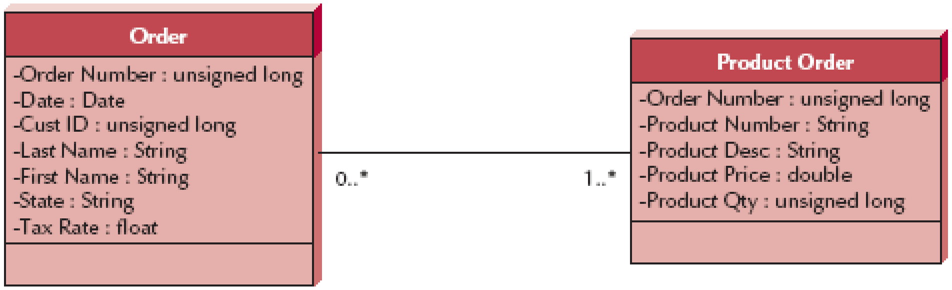
Original Model



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

Revised Model:



Order

-Order Number : unsigned long
-Date : Date
-Cust ID : unsigned long
-Last Name : String
-First Name : String
-State : String
-Tax Rate : float

Product Order

-Order Number : unsigned long
-Product Number : String
-Product Desc : String
-Product Price : double
-Product Qty : unsigned long

Note: Order Number will serve as part of the primary key of Order

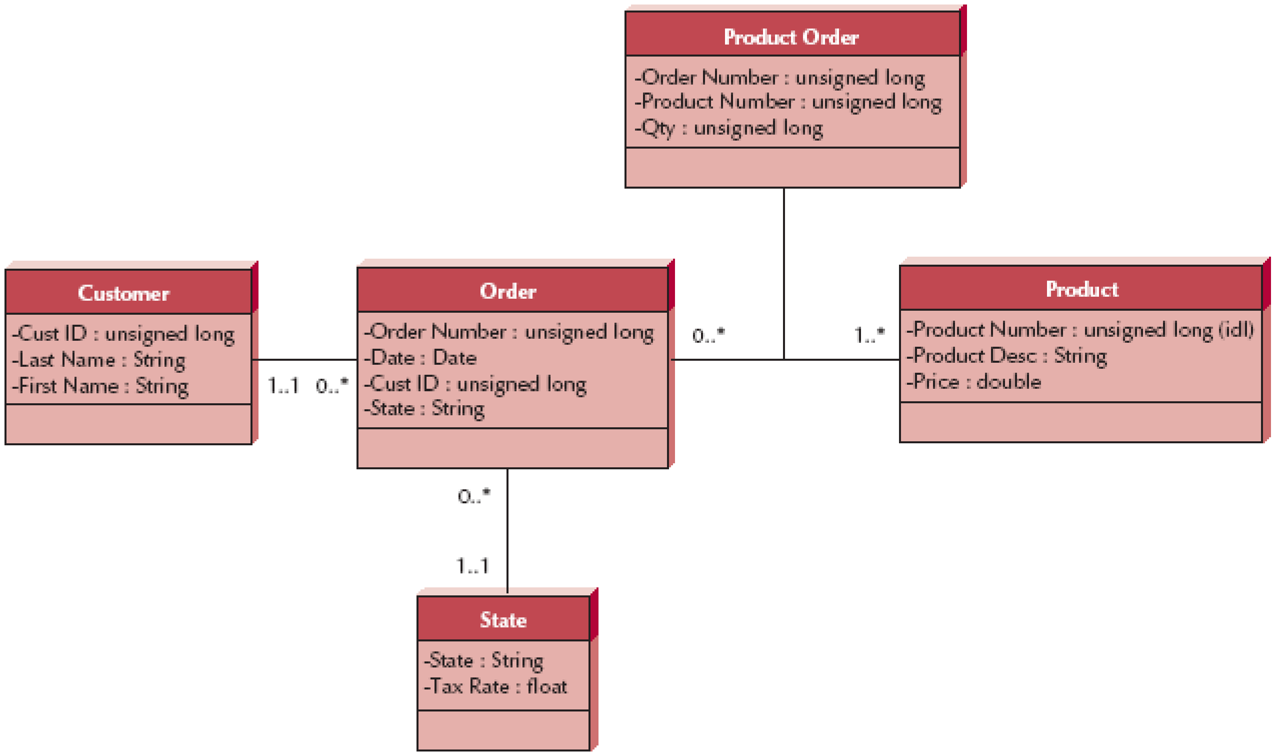
Note: Cust ID also will serve as part of the primary key of Order

Note: Order Number will serve as part of the primary key of Product Order

Note: Product Number will serve as part of the primary key of Product Order

Note: Order Number also will serve as a foreign key in Product Order

3NF Normalized Model



Product Order

- -Order Number : unsigned long
- -Product Number : unsigned long
- -Qty : unsigned long

Customer

- -Cust ID : unsigned long
- -Last Name : String
- -First Name : String

Order

- -Order Number : unsigned long
- -Date : Date
- -Cust ID : unsigned long
- -State : String

Product

- -Product Number : unsigned long (idl)
- -Product Desc : String
- -Price : double

State

- -State : String
- -Tax Rate : float

FIGURE 11-15 3NF Normalized Model

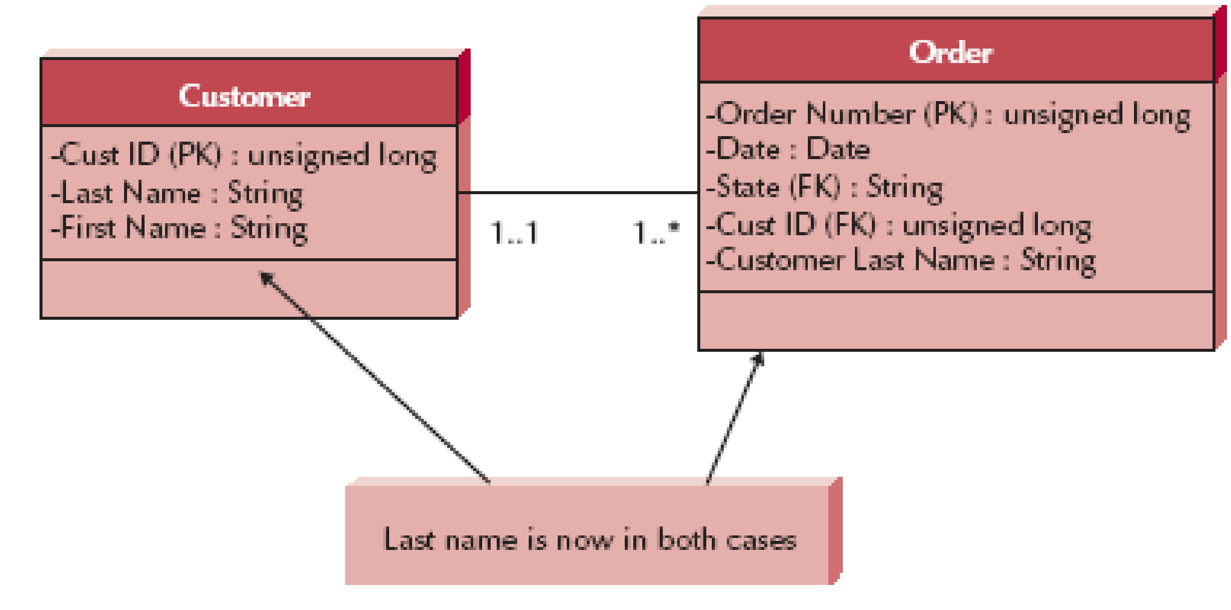
Problems with RDBMS

- To access data in multiple tables, the tables must be joined
- This can result in many database operations and lead to huge tables and slow processing

Speeding up access

- Denormalization – Adds data from one table to another in order to speed processing and eliminate a join operation
- Example: Add customer last name to order table to avoid joining order to customer to get just last name

Example



Customer

-Cust ID (PK) : unsigned long
-Last Name : String
-First Name : String

1..1

1..*

Order

-Order Number (PK) : unsigned long
-Date : Date
-State (FK) : String
-Cust ID (FK) : unsigned long
-Customer Last Name : String

Last name is now in both cases

Denormalization Candidates

- Lookup Tables
- one-to-one relationships
- include a parent entity's attributes in its child entity on the physical data model

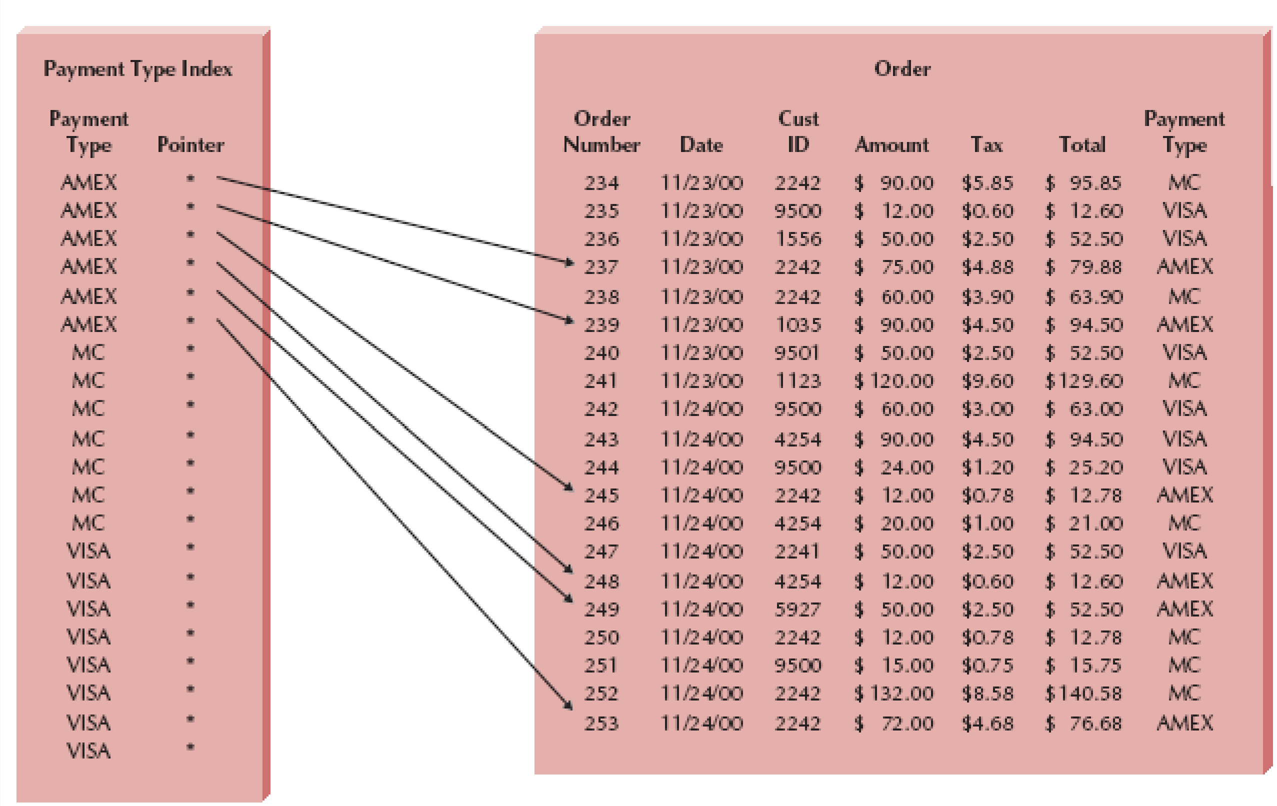
Clustering

- Interfile clustering
  - Arrange records on storage media so that similar records are stored close together
  - inter-file cluster would be similar to storing peanut butter, jelly, and bread next to each other in a grocery store since they are often purchased together.

Indexing

- An *index* in data storage is like an index in the back of a textbook;
- it is a mini table that contains values from one or more columns in a table and the location of the values within the table.
- A query can use an index to find the locations of only those records that are included in the query answer, and
- a table can have an unlimited number of indexes but too many can add overhead

Indexing Example



Payment Type Index

Payment Type	Pointer
AMEX	*
AMEX	*
AMEX	*
AMEX	*
AMEX	*
AMEX	*
MC	*
MC	*
MC	*
MC	*
MC	*
MC	*
MC	*
VISA	*
VISA	*
VISA	*
VISA	*
VISA	*
VISA	*
VISA	*
VISA	*

Order

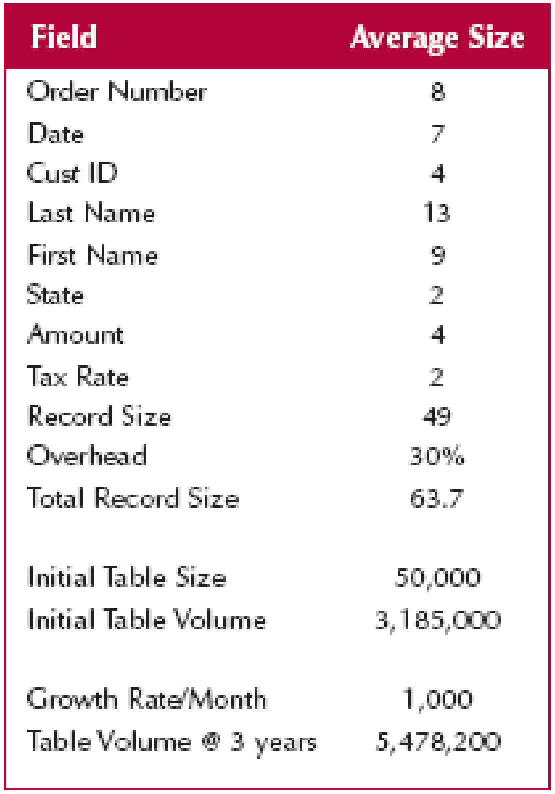
Order Number	Date	Cust ID	Amount	Tax	Total	Payment Type
234	11/23/00	2242	$ 90.00	$5.85	$ 95.85	MC
235	11/23/00	9500	$ 12.00	$0.60	$ 12.60	VISA
236	11/23/00	1556	$ 50.00	$2.50	$ 52.50	VISA
237	11/23/00	2242	$ 75.00	$4.88	$ 79.88	AMEX
238	11/23/00	2242	$ 60.00	$3.90	$ 63.90	MC
239	11/23/00	1035	$ 90.00	$4.50	$ 94.50	AMEX
240	11/23/00	9501	$ 50.00	$2.50	$ 52.50	VISA
241	11/23/00	1123	$ 120.00	$9.60	$129.60	MC
242	11/24/00	9500	$ 60.00	$3.00	$ 63.00	VISA
243	11/24/00	4254	$ 90.00	$4.50	$ 94.50	VISA
244	11/24/00	9500	$ 24.00	$1.20	$ 25.20	VISA
245	11/24/00	2242	$ 12.00	$0.78	$ 12.78	AMEX
246	11/24/00	4254	$ 20.00	$1.00	$ 21.00	MC
247	11/24/00	2241	$ 50.00	$2.50	$ 52.50	VISA
248	11/24/00	4254	$ 12.00	$0.60	$ 12.60	AMEX
249	11/24/00	5927	$ 50.00	$2.50	$ 52.50	AMEX
250	11/24/00	2242	$ 12.00	$0.78	$ 12.78	MC
251	11/24/00	9500	$ 15.00	$0.75	$ 15.75	MC
252	11/24/00	2242	$ 132.00	$8.58	$140.58	MC
253	11/24/00	2242	$ 72.00	$4.68	$ 76.68	AMEX

FIGURE 11-18 Payment Type Index

Estimating Data Storage Size

- sum the values of the average width of each column (field) to find total record size
- Add overhead (vendor may provide an estimate)
- Estimate the number of records you plan to have in the database

Data Sizing Example



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

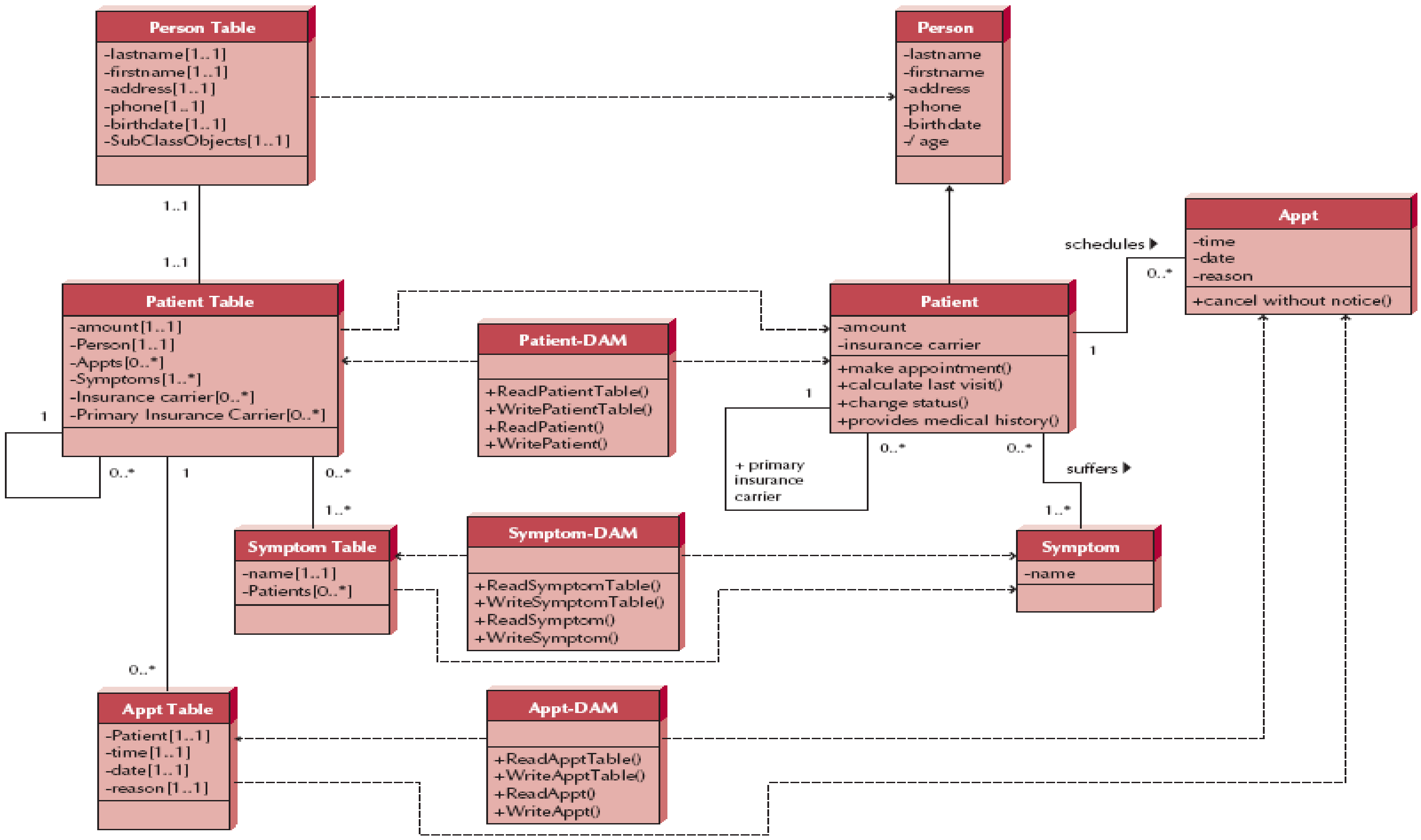
FIGURE 11-20
Calculating Volumetrics

DESIGNING DATA ACCESS AND MANIPULATION CLASSES

- Design data access and manipulation classes
- Prevent data management functionality from creeping into the problem domain classes

WILEY

Mapping PD Objects



Person Table
-lastname[1..1]
-firstname[1..1]
-address[1..1]
-phone[1..1]
-birthdate[1..1]
-SubClassObjects[1..1]

Person
-lastname
-firstname
-address
-phone
-birthdate
-/ age

Appt
-time
-date
-reason
+cancel without notice()

Patient Table
-amount[1..1]
-Person[1..1]
-Appts[0..*]
-Symptoms[1..*]
-Insurance carrier[0..*]
-Primary Insurance Carrier[0..*]

Patient-DAM
+ReadPatientTable()
+WritePatientTable()
+ReadPatient()
+WritePatient()

Patient
-amount
-insurance carrier
+make appointment()
+calculate last visit()
+change status()
+provides medical history()

Symptom Table
-name[1..1]
-Patients[0..*]

Symptom-DAM
+ReadSymptomTable()
+WriteSymptomTable()
+ReadSymptom()
+WriteSymptom()

Symptom
-name

Appt Table
-Patient[1..1]
-time[1..1]
-date[1..1]
-reason[1..1]

Appt-DAM
+ReadApptTable()
+WriteApptTable()
+ReadAppt()
+WriteAppt()

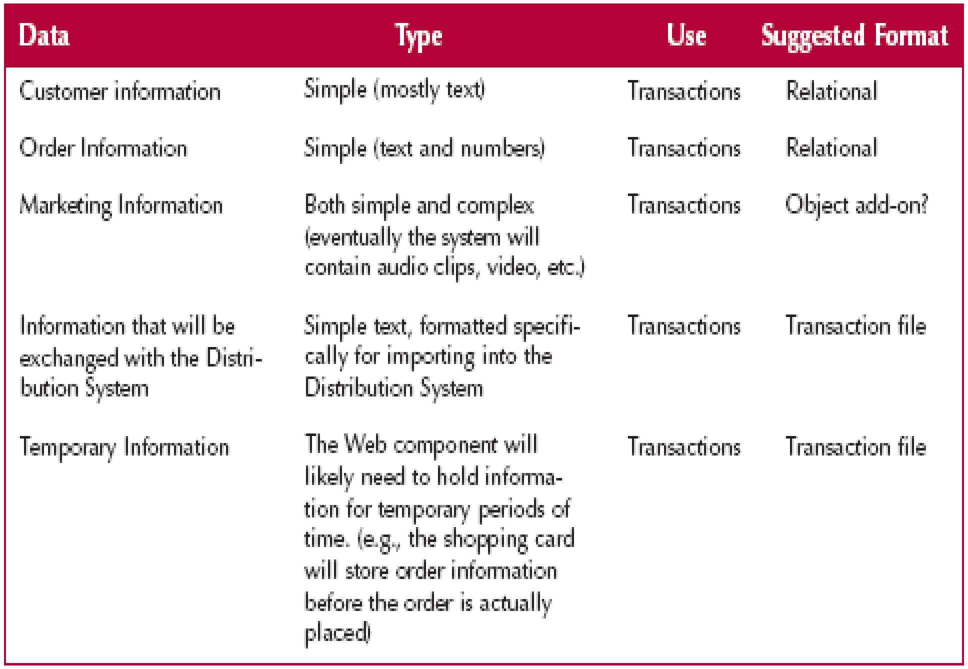
FIGURE 11-21 Mapping Problem Domain Objects to ORDBMS Using Data Access and Management Classes

CD Selections Example

- Most of the data would be text and numbers
- Thus a relational database would be able to handle the data effectively
- However, images for the catalog require complex data objects for sound and video

Looking at the Data Needs

FIGURE 11-23
Types of Data in Internet Sales System



Data	Type	Use	Suggested Format
Customer information	Simple (mostly text)	Transactions	Relational
Order Information	Simple (text and numbers)	Transactions	Relational
Marketing Information	Both simple and complex (eventually the system will contain audio clips, video, etc.)	Transactions	Object add-on?
Information that will be exchanged with the Distribution System	Simple text, formatted specifically for importing into the Distribution System	Transactions	Transaction file
Temporary Information	The Web component will likely need to hold information for temporary periods of time. (e.g., the shopping card will store order information before the order is actually placed)	Transactions	Transaction file

Object Persistent Design

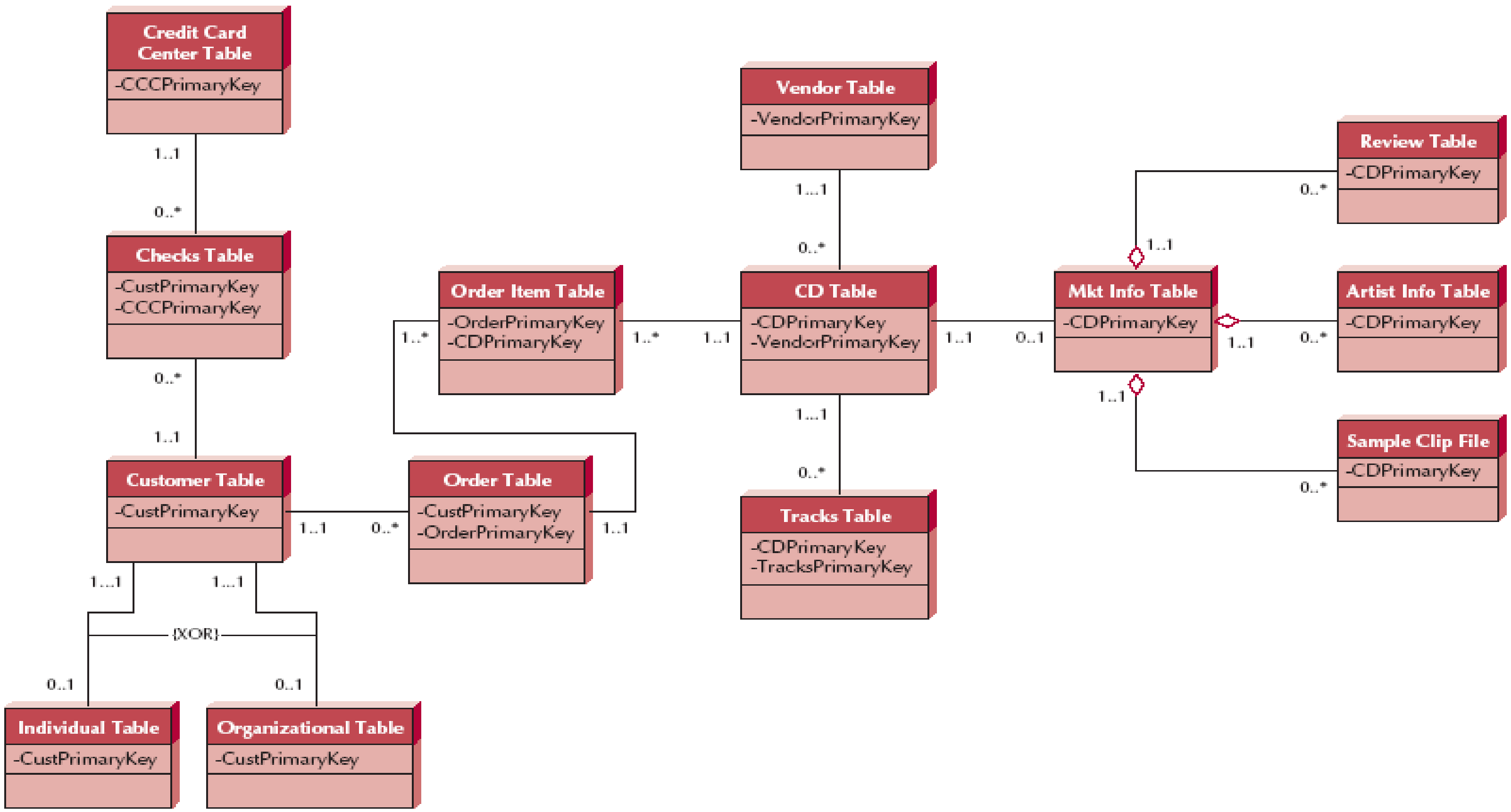
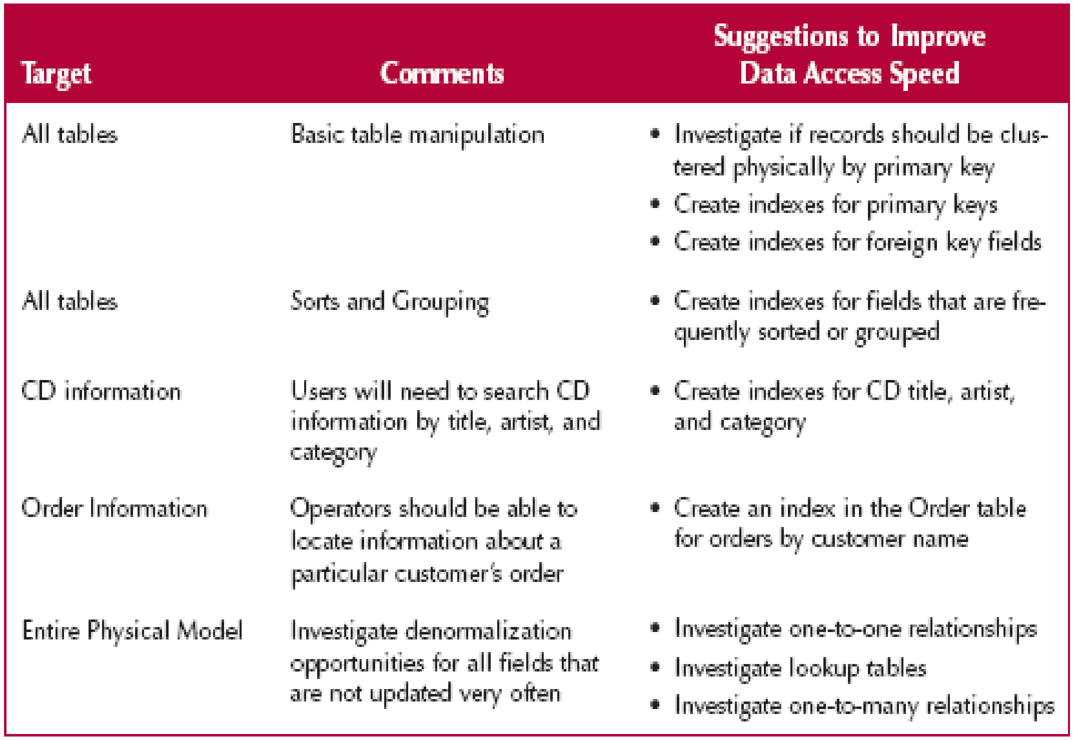


FIGURE 11-24 Internet Sales System Object Persistence Design

Optimizing Application



Target	Comments	Suggestions to Improve Data Access Speed
All tables	Basic table manipulation	• Investigate if records should be clustered physically by primary key • Create indexes for primary keys • Create indexes for foreign key fields
All tables	Sorts and Grouping	• Create indexes for fields that are frequently sorted or grouped
CD information	Users will need to search CD information by title, artist, and category	• Create indexes for CD title, artist, and category
Order Information	Operators should be able to locate information about a particular customer's order	• Create an index in the Order table for orders by customer name
Entire Physical Model	Investigate denormalization opportunities for all fields that are not updated very often	• Investigate one-to-one relationships • Investigate lookup tables • Investigate one-to-many relationships

FIGURE 11-26
Internet Sales System Performance

Problem Domain Layer

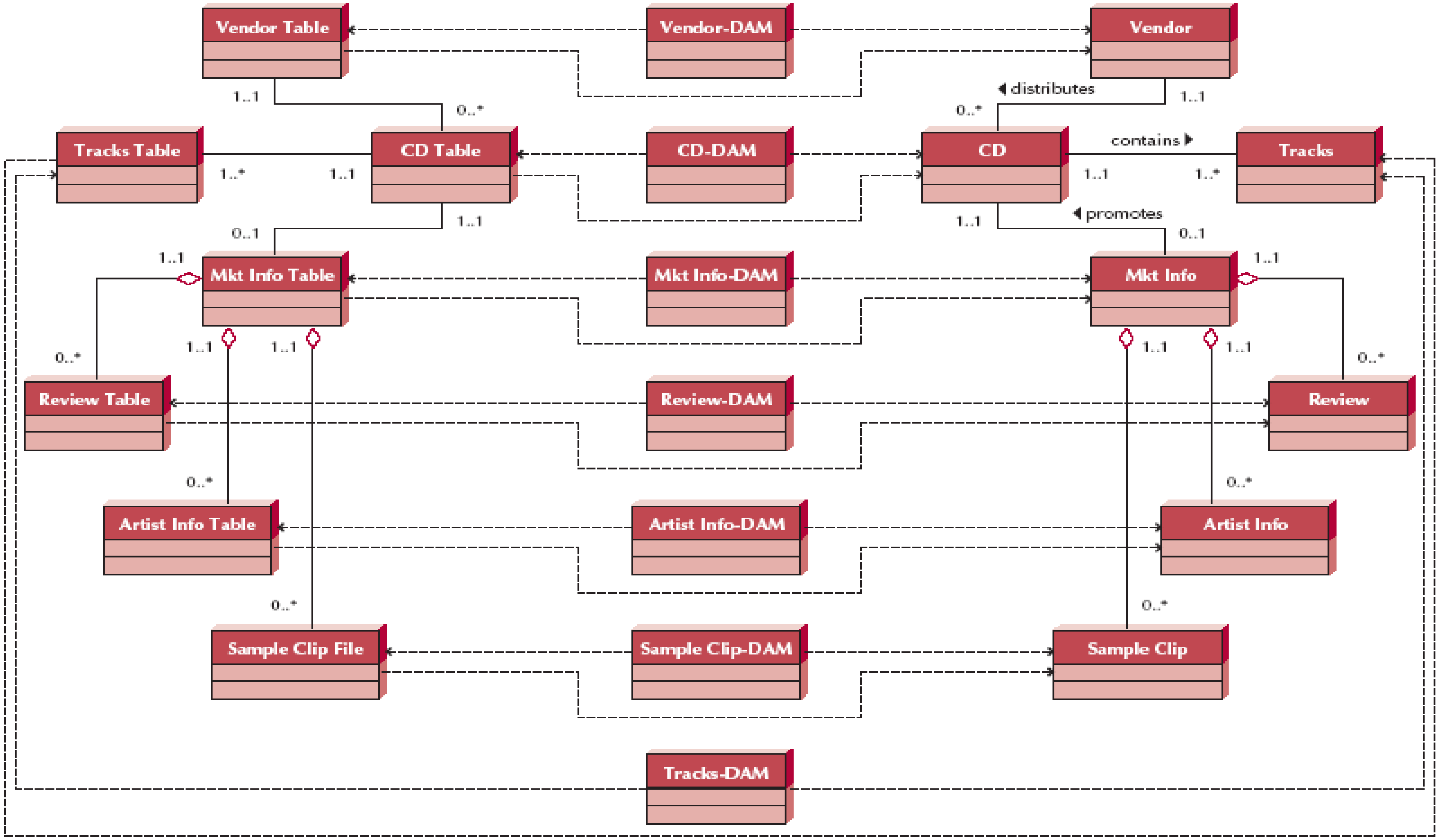


FIGURE 11-27 Data Management Layer and Problem Domain Layer Design for the CD Package of the Internet Sales System

Summary

- Choose an object-persistent format
  - Files (sequential or Random Access)
  - Databases (RDBMS, ORDBMS, OODBMS)
- Map problem domain objects to Data
- Optimizing object storage
  - Normalization
  - Denormalization, clustering, Indexes
- Design Data Access and Manipulation Classes