All rights reserved. Reproduction or translation of this work beyond that permitted in Section 117 of the 1976 United States Copyright Act without the express written permission of the copyright owner is unlawful.

Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.

The purchaser may make back-up copies for his/her own use only and not for redistribution or resale.

The Publisher assumes no responsibility for errors, omissions, or damages, caused by the use of these programs or from the use of the information contained herein.
Data Management Layer Design

Chapter 11
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

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

#### Customer

<table>
<thead>
<tr>
<th>Cust ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Prior Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2242</td>
<td>DeBerry</td>
<td>Ann</td>
<td>Y</td>
</tr>
<tr>
<td>9500</td>
<td>Chin</td>
<td>April</td>
<td>Y</td>
</tr>
<tr>
<td>1556</td>
<td>Fracken</td>
<td>Chris</td>
<td>N</td>
</tr>
<tr>
<td>1035</td>
<td>Black</td>
<td>John</td>
<td>Y</td>
</tr>
<tr>
<td>9501</td>
<td>Kaplan</td>
<td>Bruce</td>
<td>N</td>
</tr>
<tr>
<td>1123</td>
<td>Williams</td>
<td>Mary</td>
<td>N</td>
</tr>
<tr>
<td>4254</td>
<td>Bailey</td>
<td>Ryan</td>
<td>Y</td>
</tr>
<tr>
<td>2241</td>
<td>Jones</td>
<td>Chris</td>
<td>N</td>
</tr>
<tr>
<td>5927</td>
<td>Lee</td>
<td>Diane</td>
<td>N</td>
</tr>
</tbody>
</table>

#### Order

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

#### Payment Type

<table>
<thead>
<tr>
<th>Payment Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC</td>
<td>Mastercard</td>
</tr>
<tr>
<td>VISA</td>
<td>Visa</td>
</tr>
<tr>
<td>AMEX</td>
<td>American Express</td>
</tr>
</tbody>
</table>
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

Customer

<table>
<thead>
<tr>
<th>Cust ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Prior Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2342</td>
<td>DeBerry</td>
<td>Ann</td>
<td>Y</td>
</tr>
<tr>
<td>9500</td>
<td>Chin</td>
<td>April</td>
<td>Y</td>
</tr>
<tr>
<td>1556</td>
<td>Fracken</td>
<td>Chris</td>
<td>N</td>
</tr>
<tr>
<td>1935</td>
<td>Black</td>
<td>John</td>
<td>Y</td>
</tr>
<tr>
<td>9501</td>
<td>Kaplan</td>
<td>Bruce</td>
<td>N</td>
</tr>
<tr>
<td>1123</td>
<td>Williams</td>
<td>Mary</td>
<td>N</td>
</tr>
<tr>
<td>4254</td>
<td>Bailey</td>
<td>Ryan</td>
<td>Y</td>
</tr>
<tr>
<td>2241</td>
<td>Jones</td>
<td>Chris</td>
<td>N</td>
</tr>
<tr>
<td>5527</td>
<td>Lee</td>
<td>Diane</td>
<td>N</td>
</tr>
</tbody>
</table>

Order

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date</th>
<th>Cust ID</th>
<th>Amount</th>
<th>Tax</th>
<th>Total</th>
<th>Payment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>234</td>
<td>11/23/00</td>
<td>2242</td>
<td>$90.00</td>
<td>$5.85</td>
<td>$95.85</td>
<td>MC</td>
</tr>
<tr>
<td>235</td>
<td>11/23/00</td>
<td>9500</td>
<td>$12.00</td>
<td>$0.60</td>
<td>$12.60</td>
<td>VISA</td>
</tr>
<tr>
<td>250</td>
<td>11/24/00</td>
<td>2242</td>
<td>$15.00</td>
<td>$0.75</td>
<td>$15.75</td>
<td>MC</td>
</tr>
<tr>
<td>251</td>
<td>11/24/00</td>
<td>9500</td>
<td>$15.00</td>
<td>$0.75</td>
<td>$15.75</td>
<td>MC</td>
</tr>
<tr>
<td>252</td>
<td>11/24/00</td>
<td>2242</td>
<td>$132.00</td>
<td>$8.50</td>
<td>$140.50</td>
<td>AMEX</td>
</tr>
<tr>
<td>253</td>
<td>11/24/00</td>
<td>2242</td>
<td>$72.00</td>
<td>$4.68</td>
<td>$76.68</td>
<td>AMEX</td>
</tr>
</tbody>
</table>

Payment Type

- MC - Mastercard
- VISA
- AMEX - American Express

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, ...)

Slide 25
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
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 sake, 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

ORDBMS Tables

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

Patient Table
- amount
- person
- appointments
- insurance carriers
- primary insurance carrier

Symptom Table
- name
- patients

Appointment Table
- patient
- time
- date
- reason

Problem Domain Classes

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

Patient
- amount
- insurance carrier
- make appointment
- calculate bill
- change status
- provides medical history

Symptom
- name

CURE 11-8  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

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:

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Date</th>
<th>Cust ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>State</th>
<th>Tax Rate</th>
<th>Prod. 1 Number</th>
<th>Prod. 1 Desc.</th>
<th>Prod. 1 Price</th>
<th>Prod. 1 Qty.</th>
<th>Prod. 2 Number</th>
<th>Prod. 2 Desc.</th>
<th>Prod. 2 Price</th>
<th>Prod. 2 Qty.</th>
</tr>
</thead>
<tbody>
<tr>
<td>239</td>
<td>1/23/00</td>
<td>1035</td>
<td>Black</td>
<td>John</td>
<td>MD</td>
<td>0.05</td>
<td>555</td>
<td>Cheese Tray</td>
<td>$45.00</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>1/24/00</td>
<td>1035</td>
<td>Black</td>
<td>John</td>
<td>MD</td>
<td>0.05</td>
<td>444</td>
<td>Wine Gift Pack</td>
<td>$60.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>273</td>
<td>1/27/00</td>
<td>1035</td>
<td>Black</td>
<td>John</td>
<td>MD</td>
<td>0.05</td>
<td>222</td>
<td>Bottle Opener</td>
<td>$12.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>241</td>
<td>1/23/00</td>
<td>1123</td>
<td>Williams</td>
<td>Mary</td>
<td>CA</td>
<td>0.08</td>
<td>444</td>
<td>Wine Gift Pack</td>
<td>$60.00</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>262</td>
<td>1/24/00</td>
<td>1123</td>
<td>Williams</td>
<td>Mary</td>
<td>CA</td>
<td>0.08</td>
<td>222</td>
<td>Bottle Opener</td>
<td>$12.00</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>267</td>
<td>1/27/00</td>
<td>1123</td>
<td>Williams</td>
<td>Mary</td>
<td>CA</td>
<td>0.04</td>
<td>222</td>
<td>Bottle Opener</td>
<td>$12.00</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>290</td>
<td>1/30/00</td>
<td>1123</td>
<td>Williams</td>
<td>Mary</td>
<td>CA</td>
<td>0.08</td>
<td>555</td>
<td>Cheese Tray</td>
<td>$45.00</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>1/23/00</td>
<td>2242</td>
<td>DeBerry</td>
<td>Ann</td>
<td>DC</td>
<td>0.065</td>
<td>111</td>
<td>Wine Guide</td>
<td>$15.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>237</td>
<td>1/23/00</td>
<td>2242</td>
<td>DeBerry</td>
<td>Ann</td>
<td>DC</td>
<td>0.065</td>
<td>444</td>
<td>Wine Gift Pack</td>
<td>$60.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>238</td>
<td>1/24/00</td>
<td>2242</td>
<td>DeBerry</td>
<td>Ann</td>
<td>DC</td>
<td>0.065</td>
<td>222</td>
<td>Bottle Opener</td>
<td>$12.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>1/24/00</td>
<td>2242</td>
<td>DeBerry</td>
<td>Ann</td>
<td>DC</td>
<td>0.065</td>
<td>222</td>
<td>Bottle Opener</td>
<td>$12.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>252</td>
<td>1/24/00</td>
<td>2242</td>
<td>DeBerry</td>
<td>Ann</td>
<td>DC</td>
<td>0.065</td>
<td>222</td>
<td>Bottle Opener</td>
<td>$12.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>253</td>
<td>1/24/00</td>
<td>2242</td>
<td>DeBerry</td>
<td>Ann</td>
<td>DC</td>
<td>0.065</td>
<td>333</td>
<td>Jams &amp; Jellies</td>
<td>$20.00</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>291</td>
<td>1/30/00</td>
<td>2242</td>
<td>DeBerry</td>
<td>Ann</td>
<td>DL</td>
<td>0.065</td>
<td>444</td>
<td>Wine Gift Pack</td>
<td>$60.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>243</td>
<td>1/24/00</td>
<td>4254</td>
<td>Bailey</td>
<td>Ryan</td>
<td>MD</td>
<td>0.05</td>
<td>555</td>
<td>Cheese Tray</td>
<td>$45.00</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>246</td>
<td>1/24/00</td>
<td>4254</td>
<td>Bailey</td>
<td>Ryan</td>
<td>MD</td>
<td>0.05</td>
<td>333</td>
<td>Jams &amp; Jellies</td>
<td>$20.00</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>248</td>
<td>1/24/00</td>
<td>4254</td>
<td>Bailey</td>
<td>Ryan</td>
<td>MD</td>
<td>0.05</td>
<td>222</td>
<td>Bottle Opener</td>
<td>$12.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>1/23/00</td>
<td>9500</td>
<td>Chin</td>
<td>April</td>
<td>KS</td>
<td>0.05</td>
<td>222</td>
<td>Bottle Opener</td>
<td>$12.00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>244</td>
<td>1/23/00</td>
<td>9500</td>
<td>Chin</td>
<td>April</td>
<td>KS</td>
<td>0.05</td>
<td>333</td>
<td>Jams &amp; Jellies</td>
<td>$20.00</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>242</td>
<td>1/24/00</td>
<td>9500</td>
<td>Chin</td>
<td>April</td>
<td>KS</td>
<td>0.05</td>
<td>222</td>
<td>Bottle Opener</td>
<td>$12.00</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>251</td>
<td>1/24/00</td>
<td>9500</td>
<td>Chin</td>
<td>April</td>
<td>KS</td>
<td>0.05</td>
<td>111</td>
<td>Wine Guide</td>
<td>$15.00</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 11.11** Optimizing Storage
Normalization

FIGURE 11-12
The Steps of Normalization

1. **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

2. **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

3. **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

4. **Third Normal Form**
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

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
- Product Desc : String
- Price : double

State
- State : String
- Tax Rate : float

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

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

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

**FIGURE 11-18 Payment Type Index**

<table>
<thead>
<tr>
<th>Payment Type Index</th>
<th>Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMEX</td>
<td>*</td>
</tr>
<tr>
<td>AMEX</td>
<td>*</td>
</tr>
<tr>
<td>AMEX</td>
<td>*</td>
</tr>
<tr>
<td>AMEX</td>
<td>*</td>
</tr>
<tr>
<td>AMEX</td>
<td>*</td>
</tr>
<tr>
<td>AMEX</td>
<td>*</td>
</tr>
<tr>
<td>MC</td>
<td>*</td>
</tr>
<tr>
<td>MC</td>
<td>*</td>
</tr>
<tr>
<td>MC</td>
<td>*</td>
</tr>
<tr>
<td>MC</td>
<td>*</td>
</tr>
<tr>
<td>MC</td>
<td>*</td>
</tr>
<tr>
<td>MC</td>
<td>*</td>
</tr>
<tr>
<td>VISA</td>
<td>*</td>
</tr>
<tr>
<td>VISA</td>
<td>*</td>
</tr>
<tr>
<td>VISA</td>
<td>*</td>
</tr>
<tr>
<td>VISA</td>
<td>*</td>
</tr>
<tr>
<td>VISA</td>
<td>*</td>
</tr>
<tr>
<td>VISA</td>
<td>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order</th>
<th>Date</th>
<th>Cust ID</th>
<th>Amount</th>
<th>Tax</th>
<th>Total</th>
<th>Payment Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>234</td>
<td>11/23/00</td>
<td>2242</td>
<td>$ 90.00</td>
<td>$ 5.85</td>
<td>$ 95.85</td>
<td>MC</td>
</tr>
<tr>
<td>235</td>
<td>11/23/00</td>
<td>9500</td>
<td>$ 12.00</td>
<td>$ 0.60</td>
<td>$ 12.60</td>
<td>VISA</td>
</tr>
<tr>
<td>236</td>
<td>11/23/00</td>
<td>1556</td>
<td>$ 50.00</td>
<td>$ 2.50</td>
<td>$ 52.50</td>
<td>VISA</td>
</tr>
<tr>
<td>237</td>
<td>11/23/00</td>
<td>2242</td>
<td>$ 75.00</td>
<td>$ 4.88</td>
<td>$ 79.88</td>
<td>AMEX</td>
</tr>
<tr>
<td>238</td>
<td>11/23/00</td>
<td>2242</td>
<td>$ 60.00</td>
<td>$ 3.50</td>
<td>$ 63.50</td>
<td>MC</td>
</tr>
<tr>
<td>239</td>
<td>11/23/00</td>
<td>1035</td>
<td>$ 90.00</td>
<td>$ 4.50</td>
<td>$ 94.50</td>
<td>AMEX</td>
</tr>
<tr>
<td>240</td>
<td>11/23/00</td>
<td>5501</td>
<td>$ 50.00</td>
<td>$ 2.50</td>
<td>$ 52.50</td>
<td>VISA</td>
</tr>
<tr>
<td>241</td>
<td>11/23/00</td>
<td>1123</td>
<td>$ 120.00</td>
<td>$ 6.00</td>
<td>$ 126.00</td>
<td>MC</td>
</tr>
<tr>
<td>242</td>
<td>11/24/00</td>
<td>5500</td>
<td>$ 60.00</td>
<td>$ 3.00</td>
<td>$ 63.00</td>
<td>VISA</td>
</tr>
<tr>
<td>243</td>
<td>11/24/00</td>
<td>4254</td>
<td>$ 90.00</td>
<td>$ 4.50</td>
<td>$ 94.50</td>
<td>VISA</td>
</tr>
<tr>
<td>244</td>
<td>11/24/00</td>
<td>5500</td>
<td>$ 24.00</td>
<td>$ 1.20</td>
<td>$ 25.20</td>
<td>VISA</td>
</tr>
<tr>
<td>245</td>
<td>11/24/00</td>
<td>2242</td>
<td>$ 12.00</td>
<td>$ 0.78</td>
<td>$ 12.78</td>
<td>AMEX</td>
</tr>
<tr>
<td>246</td>
<td>11/24/00</td>
<td>4254</td>
<td>$ 20.00</td>
<td>$ 1.00</td>
<td>$ 21.00</td>
<td>MC</td>
</tr>
<tr>
<td>247</td>
<td>11/24/00</td>
<td>2241</td>
<td>$ 50.00</td>
<td>$ 2.50</td>
<td>$ 52.50</td>
<td>VISA</td>
</tr>
<tr>
<td>248</td>
<td>11/24/00</td>
<td>4254</td>
<td>$ 12.00</td>
<td>$ 0.60</td>
<td>$ 12.60</td>
<td>AMEX</td>
</tr>
<tr>
<td>249</td>
<td>11/24/00</td>
<td>5927</td>
<td>$ 50.00</td>
<td>$ 2.50</td>
<td>$ 52.50</td>
<td>AMEX</td>
</tr>
<tr>
<td>250</td>
<td>11/24/00</td>
<td>2242</td>
<td>$ 12.00</td>
<td>$ 0.78</td>
<td>$ 12.78</td>
<td>MC</td>
</tr>
<tr>
<td>251</td>
<td>11/24/00</td>
<td>5500</td>
<td>$ 15.00</td>
<td>$ 0.75</td>
<td>$ 15.75</td>
<td>MC</td>
</tr>
<tr>
<td>252</td>
<td>11/24/00</td>
<td>2242</td>
<td>$ 132.00</td>
<td>$ 6.50</td>
<td>$ 138.50</td>
<td>MC</td>
</tr>
<tr>
<td>253</td>
<td>11/24/00</td>
<td>2242</td>
<td>$ 72.00</td>
<td>$ 4.68</td>
<td>$ 76.68</td>
<td>AMEX</td>
</tr>
</tbody>
</table>
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

![Table: Calculating Volumetrics](image)

<table>
<thead>
<tr>
<th>Field</th>
<th>Average Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Number</td>
<td>8</td>
</tr>
<tr>
<td>Date</td>
<td>7</td>
</tr>
<tr>
<td>Cust ID</td>
<td>4</td>
</tr>
<tr>
<td>Last Name</td>
<td>13</td>
</tr>
<tr>
<td>First Name</td>
<td>9</td>
</tr>
<tr>
<td>State</td>
<td>2</td>
</tr>
<tr>
<td>Amount</td>
<td>4</td>
</tr>
<tr>
<td>Tax Rate</td>
<td>2</td>
</tr>
<tr>
<td>Record Size</td>
<td>49</td>
</tr>
<tr>
<td>Overhead</td>
<td>30%</td>
</tr>
<tr>
<td>Total Record Size</td>
<td>63.7</td>
</tr>
</tbody>
</table>

**FIGURE 11-20**  
Calculating Volumetrics

- Initial Table Size: 50,000
- Initial Table Volume: 3,185,000
- Growth Rate/Month: 1,000
- Table Volume @ 3 years: 5,478,200
DESIGNING DATA ACCESS AND MANIPULATION CLASSES

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

**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

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Use</th>
<th>Suggested Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Information</td>
<td>Simple (mostly text)</td>
<td>Transactions</td>
<td>Relational</td>
</tr>
<tr>
<td>Order Information</td>
<td>Simple (text and numbers)</td>
<td>Transactions</td>
<td>Relational</td>
</tr>
<tr>
<td>Marketing Information</td>
<td>Both simple and complex</td>
<td>Transactions</td>
<td>Object add-on?</td>
</tr>
<tr>
<td>(eventually the system will contain audio clips, video, etc.)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information that will be exchanged with the Distribution System</td>
<td>Simple text, formatted specifically for importing into the Distribution System</td>
<td>Transactions</td>
<td>Transaction file</td>
</tr>
<tr>
<td>Temporary Information</td>
<td>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)</td>
<td>Transactions</td>
<td>Transaction file</td>
</tr>
</tbody>
</table>
Object Persistent Design

FIGURE 11-24 Internet Sales System Object Persistence Design
## Optimizing Application

**FIGURE 11-26**

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

Slide 54

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