Systems Analysis and Design in a Changing World, Fourth Edition

CHAPTER 12

DESIGNING DATABASES

Learning Objectives

- Describe the differences and similarities between relational and object-oriented database management systems
- Design a relational database schema based on an entity-relationship diagram
- Design an object database schema based on a class diagram

Learning Objectives (continued)

- Design a relational schema to implement a hybrid object-relational database
- Describe the different architectural models for distributed databases

Overview

- This chapter describes design of relational and OO data models
- Developers transform conceptual data models into detailed database models
  - Entity-relationship diagrams (ERDs) for traditional analysis
  - Class diagrams for object-oriented (OO) analysis
- Detailed database models are implemented with database management system (DBMS)

12

Databases and Database Management Systems

- Databases (DB) – integrated collections of stored data that are centrally managed and controlled
  - Entity or class attribute(eg. names, prices).
  - Relationships among the entities or classes (eg. which orders belong to which customers) .
  - Stores descriptive information about data, such as field names, restrictions on allowed data and access control to sensitive information.

12

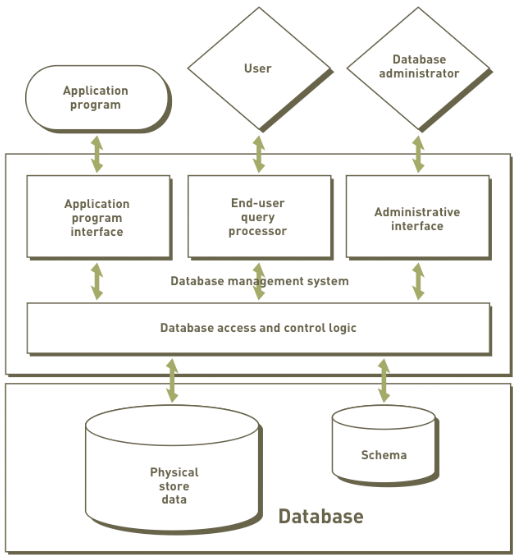
Databases and Database Management Systems (Cont.)

- The database (DB) consists of two related information store:
  - The physical data store: used by DBMS to store the raw bits and bytes of a database.
  - The schema: description of structure, content, and access controls of a physical data store or DB.
- Database management system (DBMS) – system software that manages and controls access to database (eg. Microsoft access, Oracle, DM2)

12

Components of a DB and DBMS

Figure 12-1
The components of a database and database management system and their interaction with application programs, users, and database administrators



12

Important DBMS Capabilities

- Simultaneous access by multiple users and applications.
- Access to data without writing application programs (via a query language).
- Organizational data management with uniform access and content controls

12

Database Models

- Impacted by technology changes since 1960s
- Model types
  - Hierarchical
  - Network
  - Relational
  - Object-oriented
- Most current systems use relational or object-oriented data models

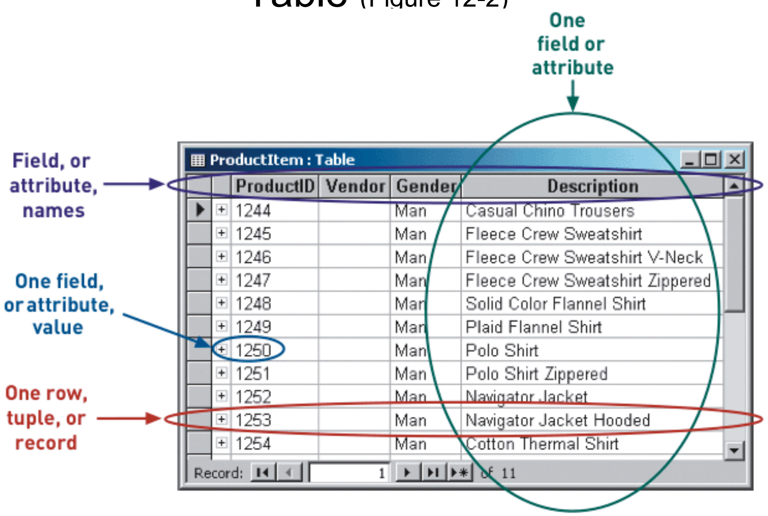
12

Relational Databases

- Relational database management system (RDBMS): organizes data into tables or relations
- Tables are two dimensional data structures
  - Tuples – **rows** or records
  - Fields – **columns** or attributes
- Tables have primary key field(s) that can be used to identify unique row of relational database table.
- Keys: A field that contains a value that is unique within each row of a RDBMS.

12

Partial Display of Relational Database Table (Figure 12-2)



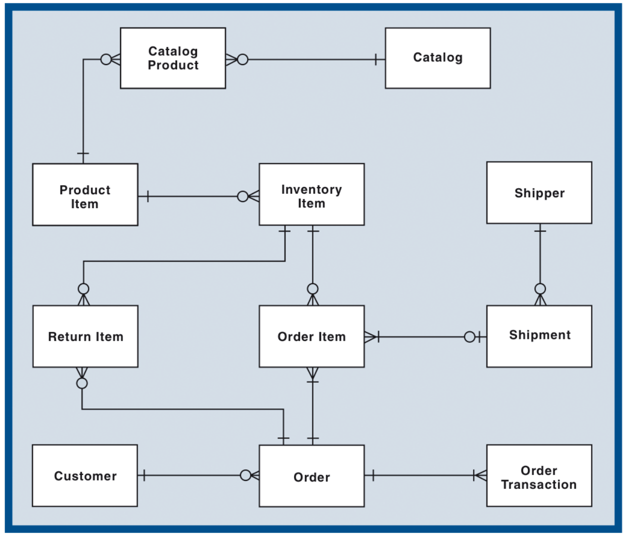
ProductID	Vendor	Gender	Description
1244		Man	Casual Chino Trousers
1245		Man	Fleece Crew Sweatshirt
1246		Man	Fleece Crew Sweatshirt V-Neck
1247		Man	Fleece Crew Sweatshirt Zippered
1248		Man	Solid Color Flannel Shirt
1249		Man	Plaid Flannel Shirt
1250		Man	Polo Shirt
1251		Man	Polo Shirt Zippered
1252		Man	Navigator Jacket
1253		Man	Navigator Jacket Hooded
1254		Man	Cotton Thermal Shirt

12

Designing Relational Databases

- Create table for each entity type
- Choose or invent primary key for each table
- Add foreign keys to represent one-to-many relationships
- Create new tables to represent many-to-many relationships
- Define referential integrity constraints
- Evaluate schema quality and make necessary improvements
- Choose appropriate data types and value restrictions (if necessary) for each field

RMO Entity-Relationship Diagram (Figure 12-5)



Entity Tables with Primary Keys (Figure 12-7)

Table	Attributes
Catalog	**CatalogID**, Season, Year, Description, EffectiveDate, EndDate
CatalogProduct	**CatalogProductID**, Price, SpecialPrice
Customer	**AccountNo**, Name, BillingAddress, ShippingAddress, DayTelephoneNumber, NightTelephoneNumber
InventoryItem	**InventoryID**, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
Order	**OrderID**, OrderDate, PriorityCode, ShippingAndHandling, Tax, GrandTotal, EmailAddress, ReplyMethod, PhoneClerk, CallStartTime, LengthOfCall, DateReceived, ProcessorClerk
OrderItem	**OrderItemID**, Quantity, Price, BackorderStatus
OrderTransaction	**OrderTransactionID**, Date, TransactionType, Amount, PaymentMethod
ProductItem	**ProductID**, Vendor, Gender, Description
ReturnItem	**ReturnItemID**, Quantity, Price, Reason, Condition, Disposal
Shipment	**TrackingNo**, DateSent, TimeSent, ShippingCost, DateArrived, TimeArrived
Shipper	**ShipperID**, Name, Address, ContactName, Telephone

Represent One-to-Many Relationships by Adding Foreign Keys (in italics) (Figure 12-8)

Table	Attributes
Catalog	**CatalogID**, Season, Year, Description, EffectiveDate, EndDate
CatalogProduct	**CatalogProductID**, Price, SpecialPrice
Customer	**AccountNo**, Name, BillingAddress, ShippingAddress, DayTelephoneNumber, NightTelephoneNumber
InventoryItem	**InventoryID**, *ProductID*, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
Order	**OrderID**, *AccountNo*, OrderDate, PriorityCode, ShippingAndHandling, Tax, GrandTotal, EmailAddress, ReplyMethod, PhoneClerk, CallStartTime, LengthOfCall, DateReceived, ProcessorClerk
OrderItem	**OrderItemID**, *OrderID*, *InventoryID*, *TrackingNo*, Quantity, Price, BackorderStatus
OrderTransaction	**OrderTransactionID**, *OrderID*, Date, TransactionType, Amount, PaymentMethod
ProductItem	**ProductID**, Vendor, Gender, Description
ReturnItem	**ReturnItemID**, *OrderID*, *InventoryID*, Quantity, Price, Reason, Condition, Disposal
Shipment	**TrackingNo**, *ShipperID*, DateSent, TimeSent, ShippingCost, DateArrived, TimeArrived
Shipper	**ShipperID**, Name, Address, ContactName, Telephone

Representing Relationships

- Relational databases use foreign keys to represent relationships
- One-to-many relationship
  - Add primary key field of "one" entity type as foreign key in table that represents "many" entity type
- Many-to-many relationship
  - Use the primary key field(s) of both entity types
  - Use (or create) an associative entity table to represent relationship (Figure 12-9 in text book)

Relationship Between Data in Two Tables

Figure 12-4
A relationship between data in two tables; the foreign key ProductID in the InventoryItem table refers to the primary key ProductID in the ProductItem table

ProductItem : Table

ProductID	Vendor	Gender	Description
1244		Man	Casual Chino Trousers
1245		Man	Fleece Crew Sweatshirt
1246		Man	Fleece Crew Sweatshirt V-Neck
1247		Man	Fleece Crew Sweatshirt Zippered
1248		Man	Solid Color Flannel Shirt
1249		Man	Plaid Flannel Shirt
1250		Man	Polo Shirt
1251		Man	Polo Shirt Zippered
1252		Man	Navigator Jacket
1253		Man	Navigator Jacket Hooded
1254		Man	Cotton Thermal Shirt

Record: 1 of 11

InventoryItem : Table

InventoryID	ProductID	Size	Color	Options	QuantityOnHand	Average Cost	RecorderQuantity
86779	1244	30/30	Khaki		45	$12.75	100
86780	1244	30/30	Slate		10	$12.75	100
86781	1244	30/30	LightTan		17	$12.75	100
86782	1244	30/31	Khaki		22	$12.75	100
86783	1244	30/31	Slate		6	$12.75	100
86784	1244	30/31	LightTan		31	$12.75	100
86785	1244	30/32	Khaki		120	$12.75	100
86786	1244	30/32	Slate		28	$12.75	100
86787	1244	30/32	LightTan		21	$12.75	100
86788	1244	30/33	Khaki		7	$12.75	100
86789	1244	30/33	Slate		41	$12.75	100
86790	1244	30/34	LightTan		35	$12.75	50

Record: 1 of 12

Enforcing Referential Integrity

- Referential Integrity: describes a consistent state among foreign key and primary key(eg. An order must be from a customer).
- Every foreign key value also exists as a primary key value
- DBMS enforces referential integrity automatically after schema designer identifies primary and foreign keys

DBMS Referential Integrity Enforcement

- When rows containing foreign keys are created
  - DBMS ensures that value also exists as a primary key in a related table
- When row is deleted
  - DBMS ensures no foreign keys in related tables have same value as primary key of deleted row
- When primary key value is changed
  - DBMS ensures no foreign key values in related tables contain the same value

Evaluating Schema Quality

- High-quality data model has
  - Uniqueness of table rows and primary keys
  - Ease of implementing future data model changes (flexibility and maintainability)
  - Lack of redundant data (database normalization)
- Database design is not objective or quantitatively measured; it is experience and judgment based

Database Normalization

12

- Normalization: Ensures relational database schema quality by minimizing data redundancy.
- Normal forms minimize data redundancy
  - First normal form (1NF) – no repeating fields or groups of fields.
  - Functional dependency – one-to-one relationship between the values of two fields.
  - The relationship is formally stated as follows:

  Field A is functionally depend on field B if for each value of B there is only one corresponding value of A

  - 2NF – in 1NF and if each non-key element is functionally dependent on entire primary key
  - 3NF – in 2NF and if no non-key element is functionally dependent on any other non-key element

Decomposition of 1NF Table into 2NF Tables

12

Figure 12-12
Decomposition of a first normal form table into two second normal form tables

IssueDate is determined by CatalogID alone, not by both CatalogID and ProductID

CatalogProduct1NF : Table

CatalogID	ProductID	Price	SpecialPrice	CatalogIssueDate
23	1244	$15.00	$12.00	8/1/2008
23	1245	$15.00	$12.00	8/1/2008
23	1246	$15.00	$13.00	8/1/2008
23	1247	$15.00	$13.00	8/1/2008
23	1248	$14.00	$11.20	8/1/2008
23	1249	$14.00	$11.20	8/1/2008
23	1252	$21.00	$16.80	8/1/2008
23	1253	$21.00	$16.40	8/1/2008
23	1254	$24.00	$19.20	8/1/2008
23	1257	$19.00	$15.20	8/1/2008

Record: 11 of 11

Convert to second normal form

CatalogProduct : Table

CatalogID	ProductID	Price	SpecialPrice
23	1244	$15.00	$12.00
23	1245	$15.00	$12.00
23	1246	$15.00	$13.00
23	1247	$15.00	$13.00
23	1248	$14.00	$11.20
23	1249	$14.00	$11.20
23	1252	$21.00	$16.80
23	1253	$21.00	$16.40
23	1254	$24.00	$19.20
23	1257	$19.00	$15.20

Record: 11 of 11

Catalog : Table

CatalogID	IssueDate
19	11/1/2007
20	2/1/2008
21	4/1/2008
22	6/1/2008
23	8/1/2008
24	9/15/2008

Record: 7

Conversion of 2NF Table into 3NF Tables

12

Figure 12-13
Converting a second normal form table into two third normal form tables

CustomerAddressUSA2NF : Table

AccountNo	Street Address	State	Zip Code
134425	123 Main Street	NM	87123
187763	456 Oak Street	MO	65701
214435	678 Poplar Avenue	UT	84697

Record: 1 of 3

ZipCode determines the value for State, and ZipCode is not the key to the table

Convert to third normal form

CustomerAddressUSA : Table

AccountNo	Street Address	Zip Code
134425	123 Main Street	87123
187763	456 Oak Street	65701
214435	678 Poplar Avenue	84697

Record: 1 of 3

StateZipCodeUSA : Table

ZipCode	State
65701	MO
84697	UT
87123	NM

Record: 1

Object-Oriented Databases

12

- Direct extension of OO design and programming paradigm
- ODBMS stores data as objects or class instances and to interface with OO programming languages
- Direct support for method storage, inheritance, nested objects, object linking, and programmer-defined data types
- Object Definition Language (ODL)
  - Standard object database description language for describing structure and content of an object database

12

Designing Object Databases

- Determine which classes require persistent storage
- Define persistent classes
- Represent relationships among persistent classes
- Choose appropriate data types and value restrictions (if necessary) for each field

12

Representing Classes

- There are two types of classes for purpose of DM
- Transient classes
  - Objects exist only during lifetime of program or process
  - Examples: view layer window, pop-up menu
- Persistent classes
  - Objects not destroyed when program or process ceases execution. State must be remembered.
  - Exist independently of program or process
  - Examples: problem domain(customer information, employee information).

12

Representing Relationships

- Object identifiers
  - Used to identify objects uniquely
  - Physical storage address or reference
  - Relate objects of one class to another
- ODBMS uses attributes containing object identifiers to find objects that are related to other objects
- Keyword *relationship* can be used to declare relationships between classes

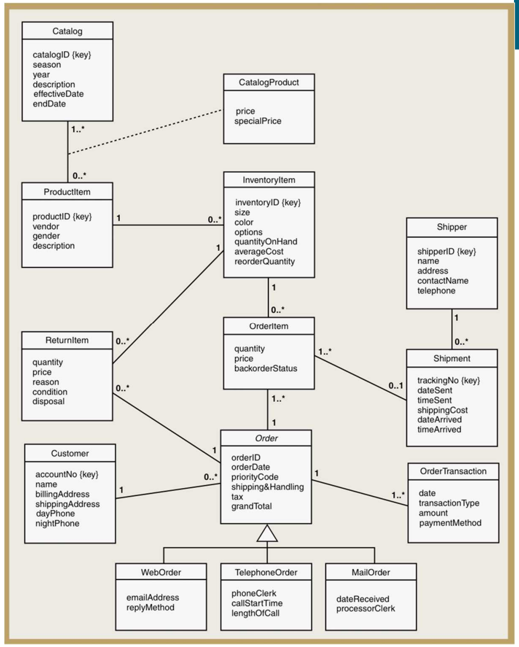
12

Representing Relationships (continued)

- Advantages include
  - ODBMS assumes responsibility for determining connection among objects
  - ODBMS assumes responsibility for maintaining referential integrity
- Type of relationships
  - 1:1, 1:M, M:M (one-to-one, one-to-many, many-to-many)
  - Association class used with M:M

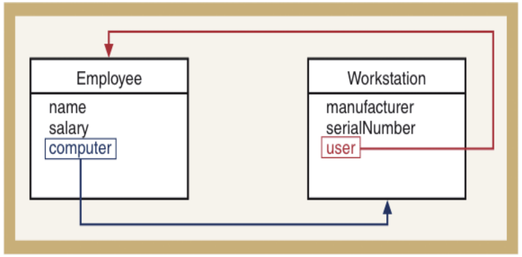
RMO Domain Model Class Diagram

(Figure 12-15)



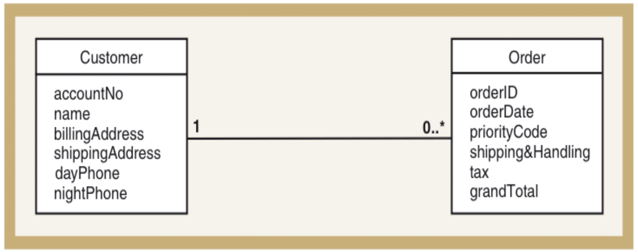
One-to-One Relationship Represented with Attributes Containing Object Identifiers

Figure 12-16
A one-to-one relationship represented with attributes (shown in color) containing object identifiers



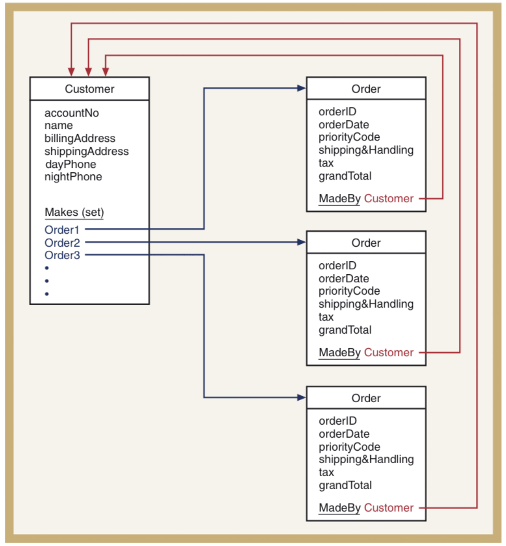
One-to-Many Relationship Between Customer and Order Classes

Figure 12-17
The one-to-many relationship between the Customer and Order classes



One-to-Many Relationship Represented with Attributes Containing Object Identifiers

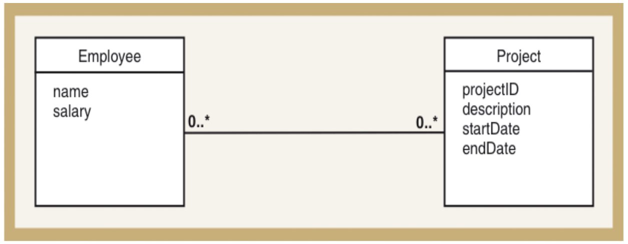
Figure 12-18
A one-to-many relationship represented with attributes containing object identifiers



Many-to-Many Relationship between Employee and Project Classes (Figure 12-19)

12

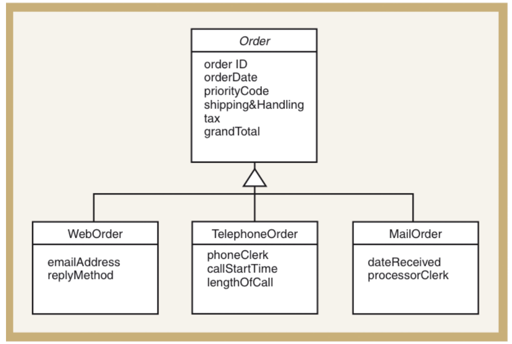
```
class Employee {
  attribute string name
  attribute string salary
  relationship set<Project> WorksOn
   inverse Project::Assigned
}
class Project {
  attribute string projectID
  attribute string description
  attribute string startDate
  attribute string endDate
  relationship set<Employee> Assigned
   inverse Employee::WorksOn
}
```



Generalization Hierarchy within the RMO Class Diagram (Figure 12-21)

12

```
class Order {
  attribute string orderID
  attribute string orderDate
  attribute string priorityCode
  attribute real shipping&Handling
  attribute real tax
  attribute real grandTotal
}
class WebOrder extends Order {
  attribute string emailAddress
  attribute string replyMethod
}
class TelephoneOrder extends Order {
  attribute string phoneClerk
  attribute string callStartTime
  attribute integer lengthOfCall
}
class MailOrder extends Order {
  attribute string dateReceived
  attribute string processorClerk
}
```



Hybrid Object-Relational Database Design

12

- RDBMS (hybrid DBMS) used to store object attributes and relationships
- Design complete relational schema and simultaneously design equivalent set of classes
- Mismatches between relational data and OO
  - Class methods cannot be directly stored or automatically executed
  - Relationships are restricted compared to ODBMS
  - ODBMS can represent wider range of data types

Classes and Attributes

12

- Designers store classes and object attributes in RDBMS by table definition
- For new system :relational schema can be designed based on class diagram
- Table is created for each class
- Fields of each table same as attributes of class
- Row holds attribute values of single object
- Key field is chosen for each table
- **อย่าลืมทำ EER Mapping (Class → Table)**
  - **Step 8 (เลือก Optionไหน 8A-8C)**
  - **Step 1-7**

Views of Stored Data

Figure 12-22
Correspondence among concepts in the object-oriented, entity-relationship, and relational database views of stored data

Object-oriented	Entity-relationship	Relational database
Class	Entity type	Table
Object	Entity instance	Row
Attribute	Attribute	Column

Relationships

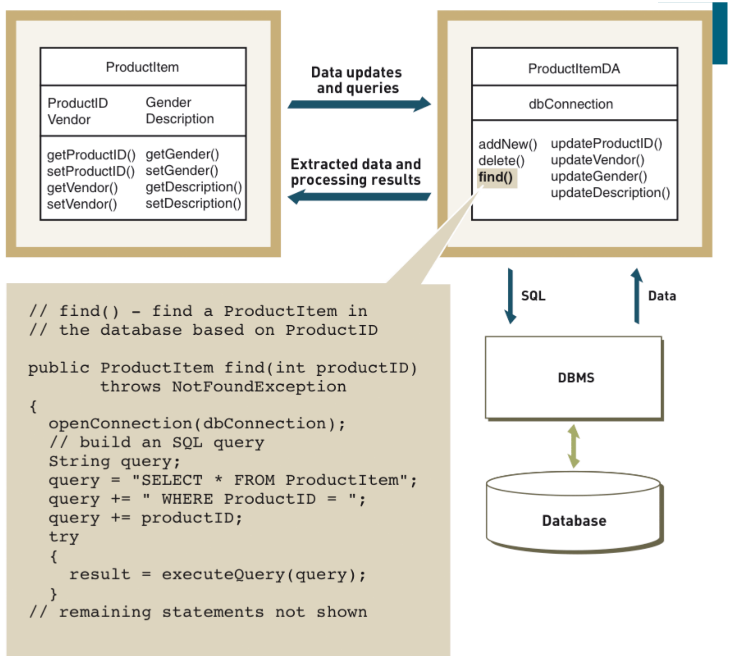
- Relationships are represented with foreign keys
- Foreign key values serve same purpose as object identifiers in ODBMS
- 1:M relationship – add primary key field of class on "one" side of the relationship to table representing class on "many" side
- M:M relationship – create new table that contains primary key fields of related class tables and attributes of the relationship itself

Data Access Classes

- OO design based on a three-layer architecture
- Data access classes are implementation bridge between data stored in program objects and data in relational database
- Methods add, update, find, and delete fields and rows in table or tables that represent the class
- Methods encapsulate logic needed to copy data values from problem domain class to database and vice versa

Interaction Among a Domain Class, a Data Access Class, and the DBMS

(Figure 12-25)



12

Data Types

- Storage format and allowable content of program variable, object state variable, or database field or attribute
- Primitive data types – directly implemented
  - Memory address (pointer), Boolean, integer, and so on
- Complex data types – user-defined
  - Dates, times, audio streams, video images, URLs

12

Relational DBMS Data Types

- Designer must choose appropriate data type for each field in relational database schema
- Choice for many fields is straightforward
  - Names and addresses use a set of fixed- or variable-length character arrays
  - Inventory quantities can use integers
  - Item prices can use real numbers
- Complex data types (DATE, LONG, LONGRAW)

12

Subset of Oracle RDBMS Data Types

Figure 12-26
A subset of the data types available in the Oracle relational DBMS

Type	Description
CHAR	Fixed-length character array
VARCHAR	Variable-length character array
NUMBER	Real number
DATE	Date and time with appropriate checks of validity
LONG	Variable-length character data up to 2 gigabytes
LONGRAW	Binary large object (BLOB) with no assumption about format or content
ROWID	Unique six-byte physical storage address

12

Object DBMS Data Types

- Use set of primitive and complex data types comparable to RDBMS data types
- Schema designer can create new data types and associated constraints
- Classes are complex user-defined data types that combine traditional concept of data with processes (methods) to manipulate data
- Flexibility to define new data types is one reason that OO tools are widely used

Distributed Databases

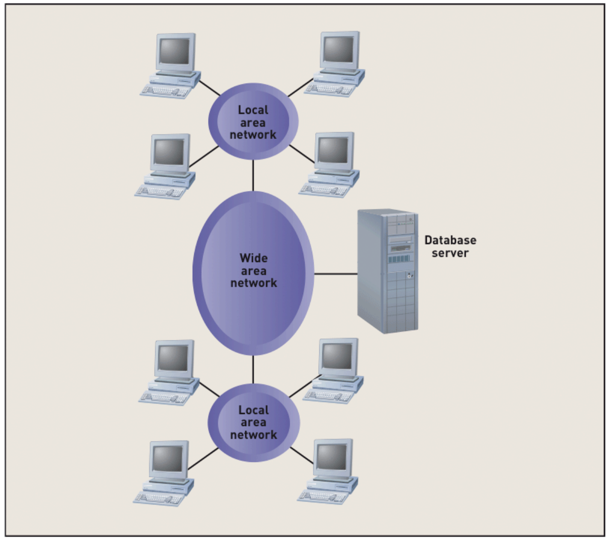
12

- Rare for all organizational data to be stored in a single database in one location
- Different information systems in an organization are developed at different times
- Parts of an organization's data may be owned and managed by different units
- System performance is improved when data is near primary applications

Single Database Server Architecture

(Figure 12-27)

12



Single Database Server Architecture (Cont.)

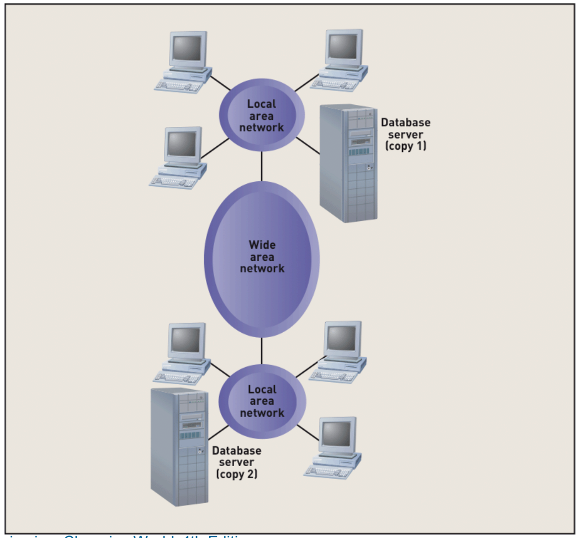
12

- Primary Advantages: Simplicity, because there are only one server to manage.
- Primary Disadvantages:
  - Server failure
  - Possible load of the server
  - No back up capabilities in the event of server failure.
- Poorly suited to applications that must be available on 24/7

Replicated Database Server Architecture

(Figure 12-28)

12

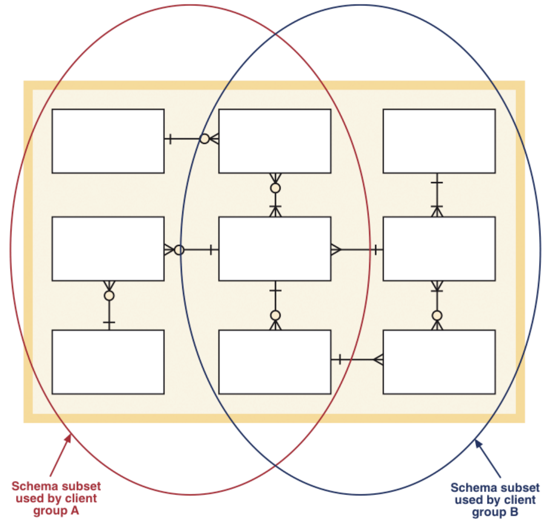


Replicated Database Server Architecture

- Designers can eliminate delay in accessing distance database server.
- More fault tolerance.
- Load balancing.
- Primary Disadvantage:
  - Data inconsistency
- Database Synchronization: The process of ensuring consistency among two or more database copies.

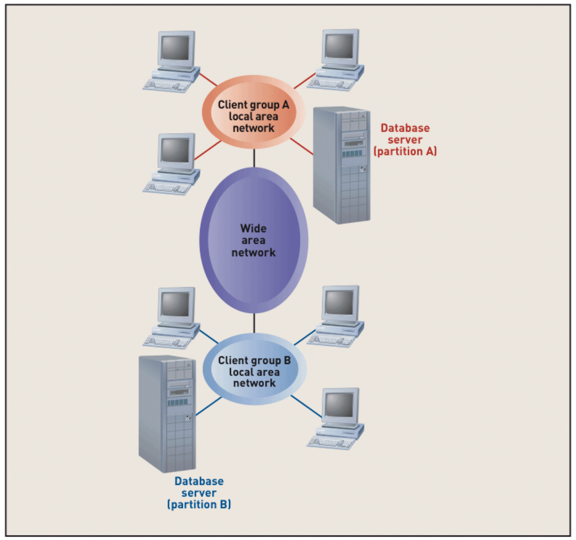
Partitioning Database Schema into Client Access Subsets

Figure 12-29
Partitioning a database schema into client access subsets



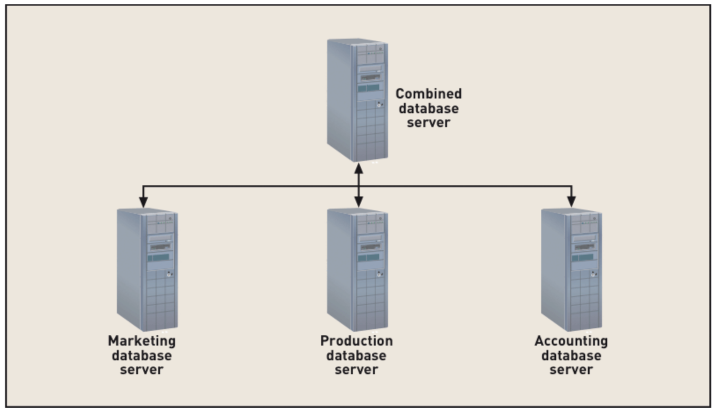
Partitioned Database Server Architecture

Figure 12-30
A partitioned database server architecture



Federated Database Server Architecture

Figure 12-31
A federated database server architecture



RMO Distributed Database Architecture

12

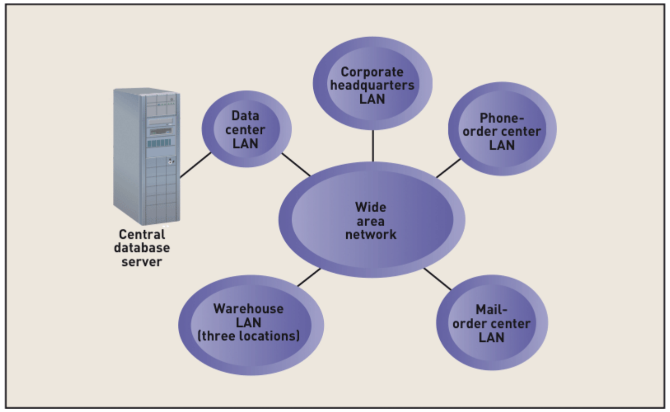
- Starting point for design was information about data needs of geographically dispersed users
- RMO gathered information during analysis phase
- RMO decided to manage database using Park City data center mainframe
- RMO is evaluating single-server vs. replicated and partitioned database server architectures
- Information on network traffic and costs needed

Single-Server Database Server Architecture for RMO

12

Figure 12-32

A single-server database architecture for RMO

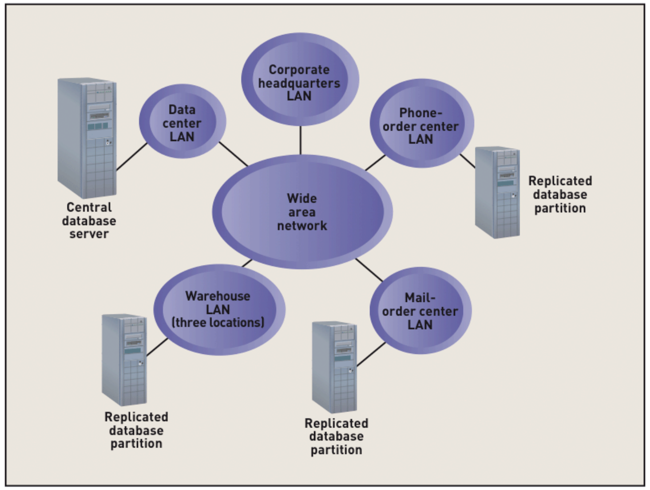


Replicated and Partitioned Database Server Architecture for RMO

12

Figure 12-33

A replicated and partitioned database server architecture for RMO



Summary

12

- Modern information systems store data in database and access and manage data using DBMS
- Relational DBMS is commonly used
- Object DBMS is increasing in popularity
- Key activity of systems design is developing relational or object database schema
- Relational database is collection of data stored in tables and is developed from entity-relationship diagram
- Object database stores data as collection of related objects and is developed from class diagram
- Objects can also be stored in RDBMS
  - RDBMS cannot store methods
  - RDBMS cannot directly represent inheritance
- Medium and larger information systems typically use multiple databases or database servers in various geographic locations