Chapter 13: Designing Databases

Systems Analysis and Design in a Changing World, 3rd Edition

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)

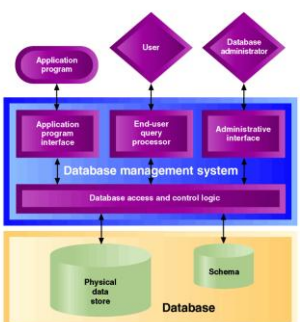
Databases and Database Management Systems

- Databases (DB) – integrated collections of stored data that are centrally managed and controlled
- Database management system (DBMS) – system software that manages and controls access to database
- Databases described by a schema: description of structure, content, and access controls

Components of a DB and DBMS

FIGURE 13-1

The components of a database and database management system and their interaction with application programs, users, and database administrators.



DBMS Important Capabilities

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

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

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) which can be used to identify unique records
- Keys relate tables to each other

Partial Display of Relational Database Table

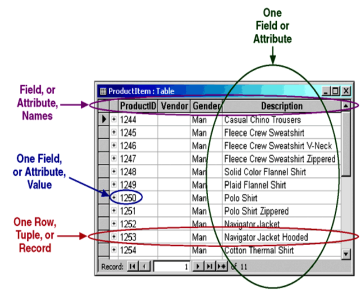


FIGURE 13-2
A partial display of a relational database table.

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

Designing Relational Databases (continued)

- Define referential integrity constraints
- Evaluate schema quality and make necessary improvements
- Choose appropriate data types and value restrictions (if necessary) for each field

13

Relationship Between Data in Two Tables

FIGURE 13-4

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

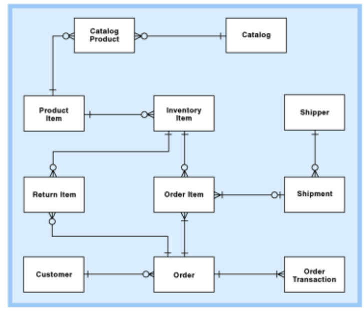
Record: 1 of 12

13

RMO Entity-Relationship Diagram

FIGURE 13-5

The RMO entity-relationship diagram.



13

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 associate entity table to represent relationship

13

Entity Tables with Primary Keys

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

FIGURE 13-7

Entity tables with the primary keys identified in bold.

13

Represent One-to-Many Relationships

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

FIGURE 13-8

Represent one-to-many relationships by adding foreign key attributes (shown in italics).

13

Enforcing Referential Integrity

- Consistent relational database state
- Every foreign key also exists as a primary key value
- DBMS enforces referential integrity automatically once 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 key 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

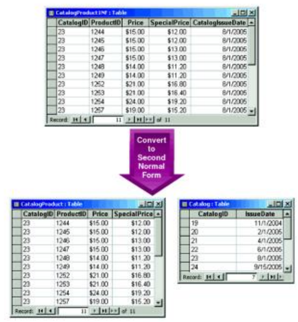
- High quality data model has:
  - Uniqueness of table rows and primary keys
  - Ease of implementing future data model changes (flexibility and maintainability)
  - Lack of redundant data (database normalization)
- Database design is not objective or quantitatively measured; it is experience and judgment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

Database Normalization

- 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
  - 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

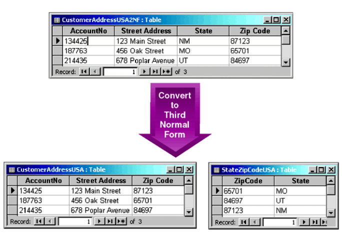
Decomposition of 1NF Table into 2NF Tables

FIGURE 13-12
Decomposition of a first normal form table into two second normal form tables.



Conversion of 2NF Table into 3NF Tables

FIGURE 13-13
Converting a second normal form table into two third normal form tables.



Object-Oriented Databases

- Direct extension of OO design and programming paradigm
- ODBMS stores data as objects or classes
- Direct support for method storage, inheritance, nested objects, object linking, and programmer-defined data types
- Object definition language (ODL)
  - Standard language for describing structure and content of an object database

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

Representing Classes

- Transient object
  - Exist only during lifetime of program or process
  - Examples: view layer window, pop-up menu
- Persistent object
  - Not destroyed when program or process ceases execution
  - Exist independently of program or process
  - Examples: customer information, employee information

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

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 Class Diagram

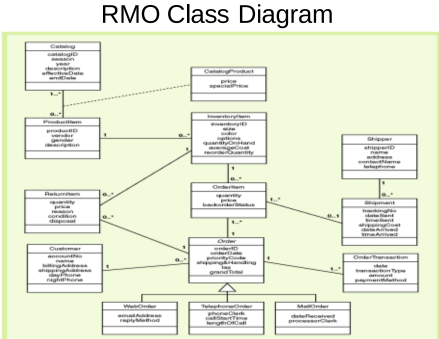
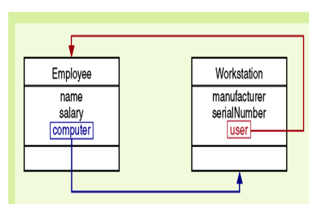


FIGURE 13-15
The RMO class diagram.

1:1 Relationship Represented with Attributes Containing Object Identifiers

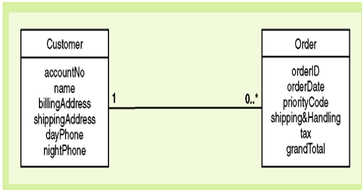
FIGURE 13-16
A one-to-one relationship represented with attributes (shown in color) containing object identifiers.



1:M Relationship Between Customer and Order Classes

FIGURE 13-17

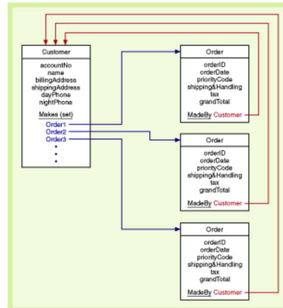
The one-to-many relationship between the Customer and Order classes.



1:M Represented with Attributes Containing Object Identifiers

FIGURE 13-18

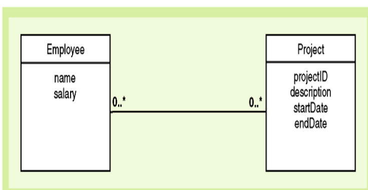
A one-to-many relationship represented with attributes containing object identifiers.



M:M Relationship between Employee and Project Classes

FIGURE 13-19

A many-to-many relationship between the Employee and Project classes.



M:M Relationship Represented with two 1:M Relationship

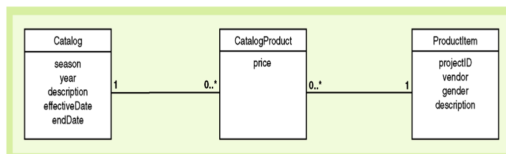


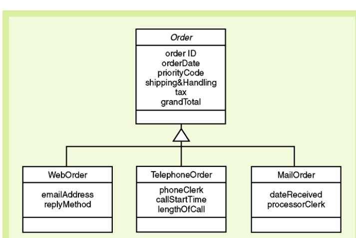
FIGURE 13-20

A many-to-many relationship represented with two one-to-many relationships.

Generalization Hierarchy within the RMO Class Diagram

FIGURE 13-21

A generalization hierarchy within the RMO class diagram.



Hybrid Object-Relational Database Design

- 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

- Designers store classes and object attributes in RDBMS by table definition
- 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

Views of Stored Data

FIGURE 13-22
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 Between Classes

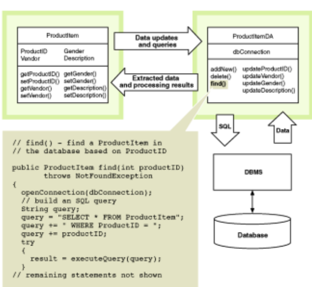


FIGURE 13-25
Interaction among a problem domain class, a data access class, and the DBMS.

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, etc.
- Complex data types: user-defined
  - Dates, times, audio streams, video images, URLs

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)

Subset of Oracle RDBMS Data Types

FIGURE 13-26

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

Object DBMS Data Types

- Uses 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 combines 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

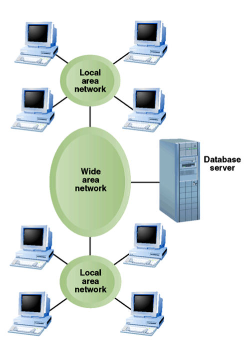
Distributed Databases

- Rare for all organizational data to be stored in one location in a single database
- 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 13-27

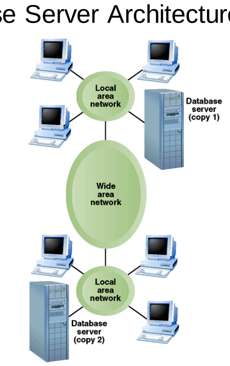
A single database server architecture.



Replicated Database Server Architecture

FIGURE 13-28

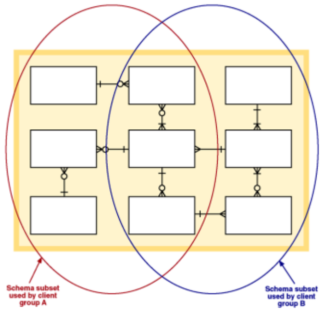
A replicated database server architecture.



Partitioning Database Schema into Client Access Subsets

FIGURE 13-29

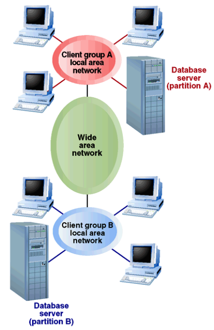
Partitioning a database schema into client access subsets.



Partitioned Database Server Architecture

FIGURE 13-30

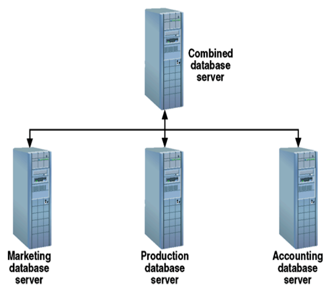
A partitioned database server architecture.



Federated Database Server Architecture

FIGURE 13-31

A federated database server architecture.



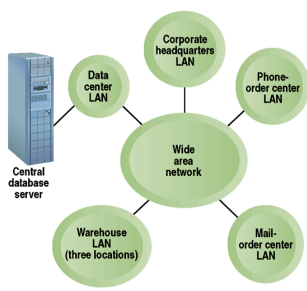
RMO Distributed Database Architecture

- Starting point for design is 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

FIGURE 13-32

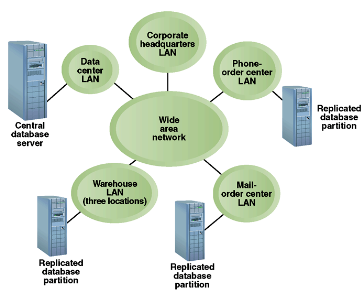
A single-server database architecture for RMO.



Replicated and Partitioned Database Server Architecture for RMO

FIGURE 13-33

A replicated and partitioned database server architecture for RMO.



Summary

- Modern information systems store data in database, 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

Summary (continued)

- Object database stores data as collection of related objects and is developed from class diagram
- Objects can also be stored within 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