Chapter 9 : Designing the Database

Tannaz R.Damavandi
Cal Poly Pomona

Outline

- Databases and Database Management Systems
- Database Design and Administration
- Relational Databases
- Distributed Database Architectures
- Protecting the Database

Overview

- Databases and database management systems are important components of a modern information system
- Database design transforms the domain model class diagram into a detailed database model for the system
- A database management system is used to implement and interact with the database

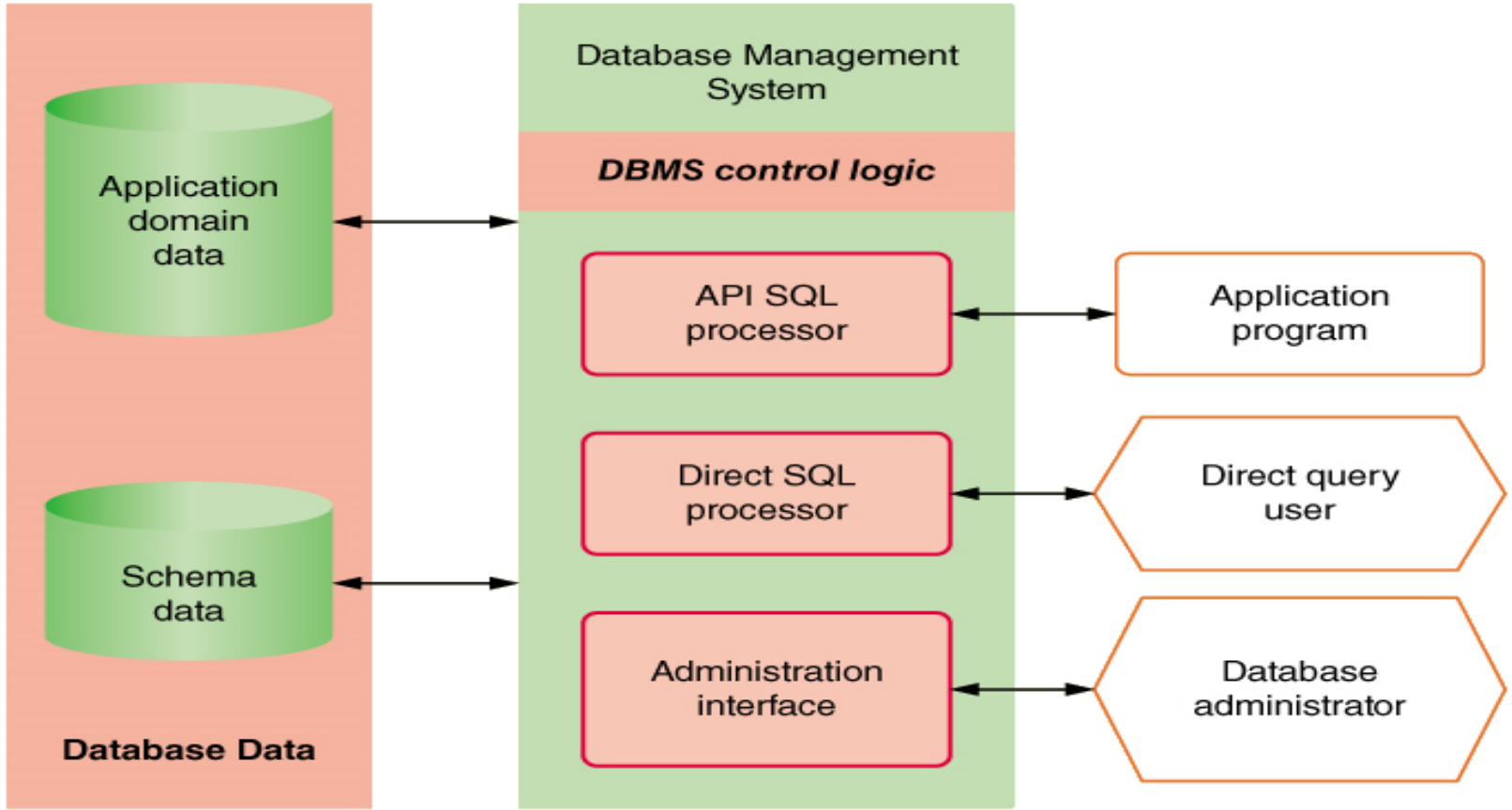
Databases and DBMSs

- Database (DB) -- an integrated collection of stored data that is centrally managed and controlled
- Database management system (DBMS) -- a system software component that manages and controls one or more databases
- Schema -- database component that contains descriptive information about the data stored in the physical data store (sometimes called *metadata*)
- Structured Query Language -- the standard query language to access and update data in a relational DBMS

Database Schema

- Organization of individual stored data items into higher level groups, such as tables
- Associations among tables or classes
- Details of physical data store organization, including types, lengths, locations, and indexing of data items
- Access and content controls, including allowable values for specific data items, value dependencies among multiple data items, and lists of users allowed to read or update data items

DBMS Components



Characteristics of a DBMS

- Simultaneous access by many users and many applications
- Direct access to data with a data interface
- Uniform and consistent access
- Integration and distribution of data across multiple servers

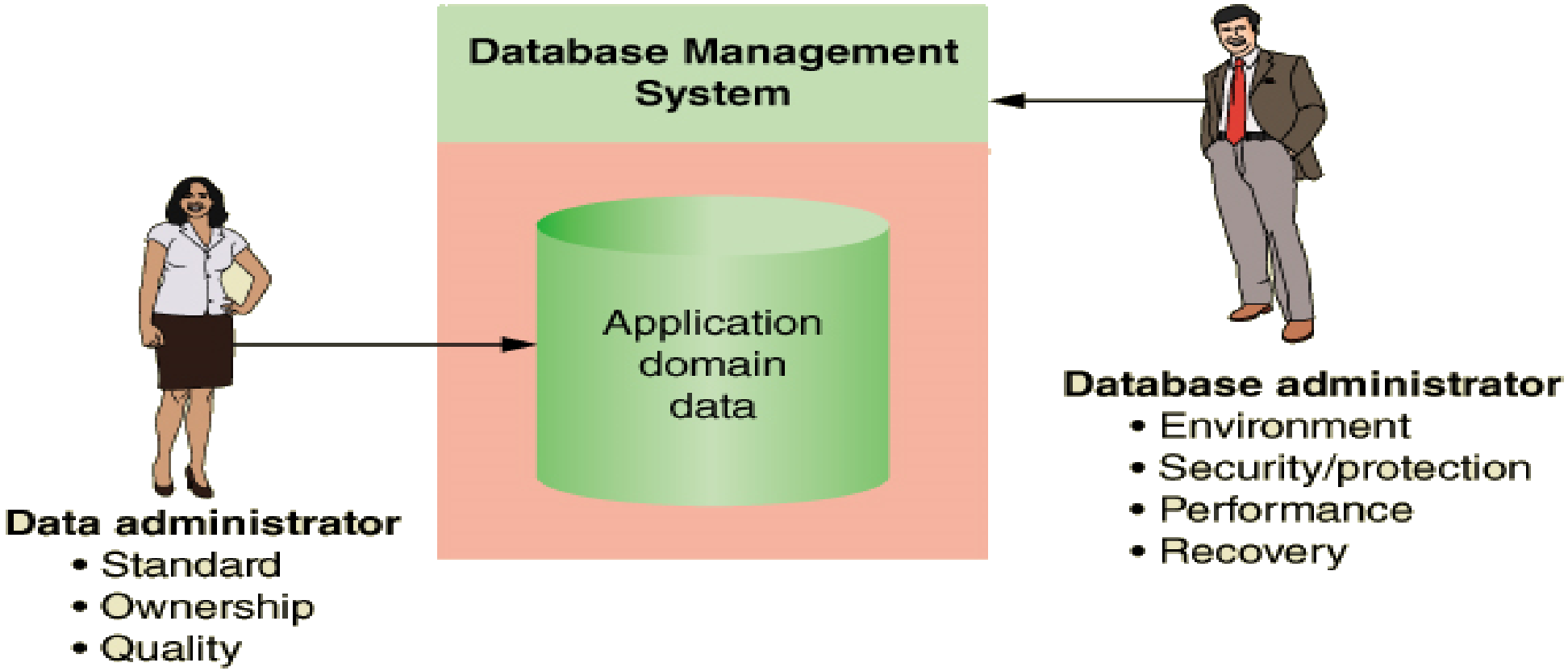
Database Design and Administration

- How does database design integrate within the project plan?
  - Water-fall development – design and implement database first
  - Iterative development – database is foundational, early iterations need to focus on data and key portions of the database
    - Important to consider database impacts of all subsystems

Database Design and Administration

- Who is involved in database design?
- Data Administrator (DA) – person in charge of structure and integrity of the data
  - Data standards – naming, definition, data typing
  - Data use – ownership, accessibility, confidentiality
  - Data quality – validation rules, completeness, currency
- Database Administrator (DBA) – person in charge of safety and the operation of the database
  - Manage multiple DBMS environment
  - Protect the database and data – authentication
  - Maintain high-performance level
  - Backup data and define recovery procedures

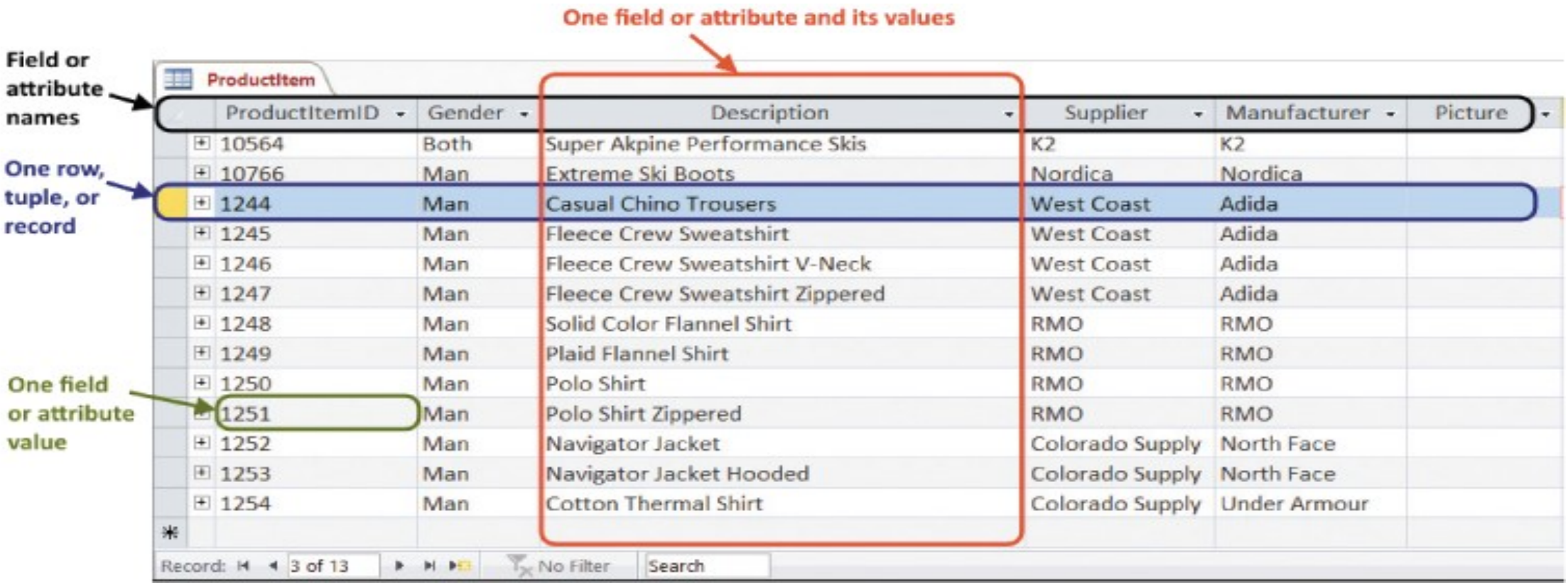
DA and DBA Responsibilities



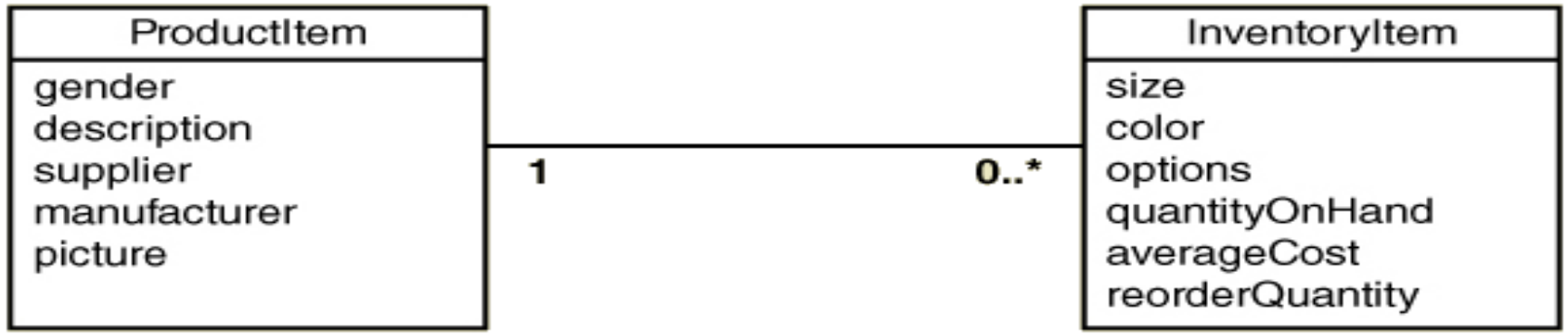
Relational Database

- Relational database management system (RDBMS) -- a DBMS that organizes data in tables (relations)
- Table – a two-dimensional data structure of columns and rows
- Row – one horizontal group of data attribute values
- Attribute – one vertical group of data attribute values
- Attribute value – the value held in a single table cell
- Key – an attribute or set of attributes, the values of which occur only once in all the rows of the table
- Candidate Key – an attribute or set of attributes that could server as the primary key
- Primary key – the key chosen by a database designer to represent relationships among rows in different tables
- Foreign key – an attribute that duplicates the primary key of a different (or foreign) table

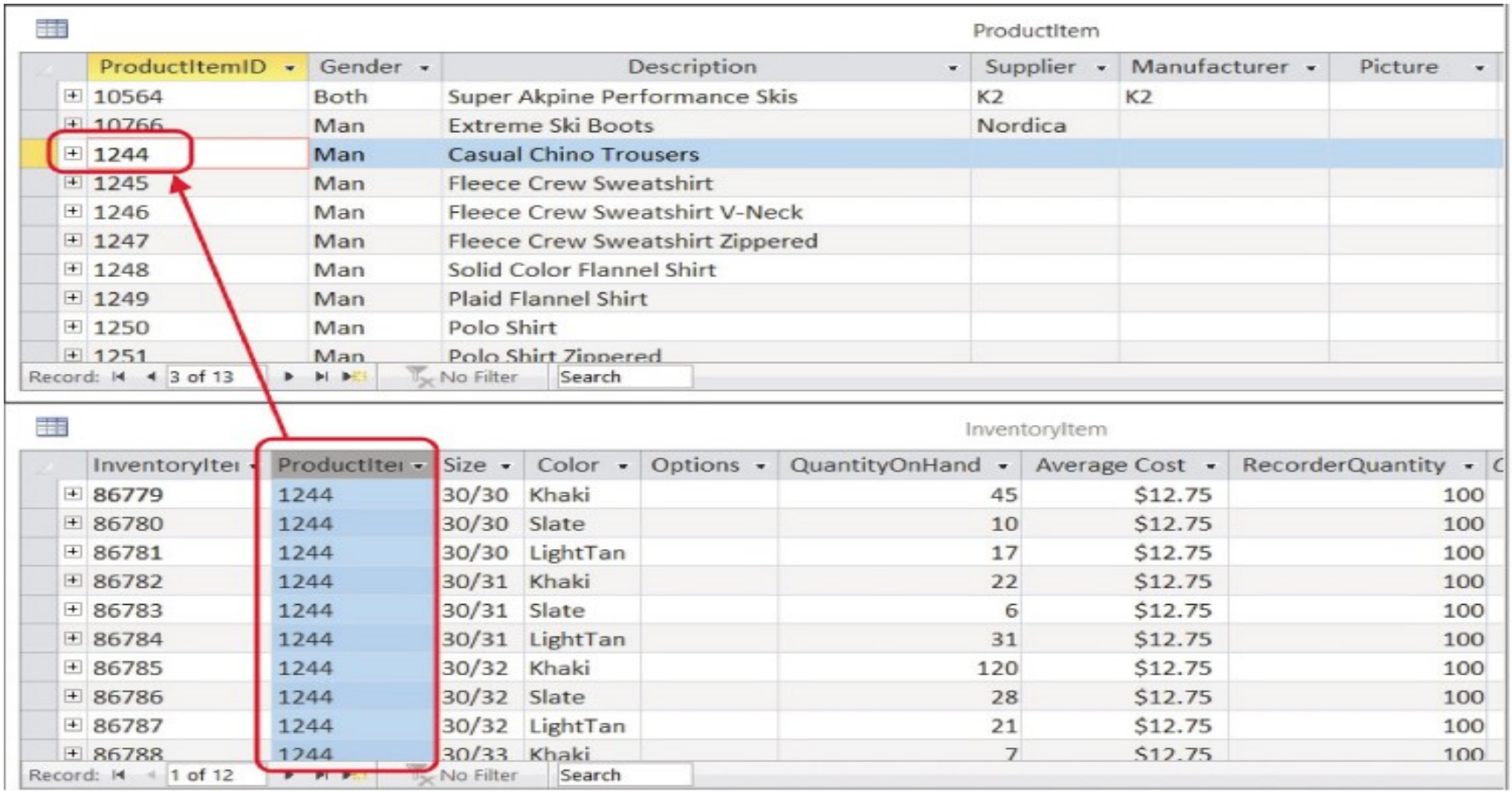
Partial Display of a Relational Database Table



An Association Between Two Classes



An Association Between Rows in Two Tables (Key and Foreign key)



ProductItem

ProductItemID	Gender	Description	Supplier	Manufacturer	Picture
10564	Both	Super Akpine Performance Skis	K2	K2	
10766	Man	Extreme Ski Boots	Nordica		
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			

Record: 3 of 13 No Filter Search

InventoryItem

InventoryItei	ProductItei	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

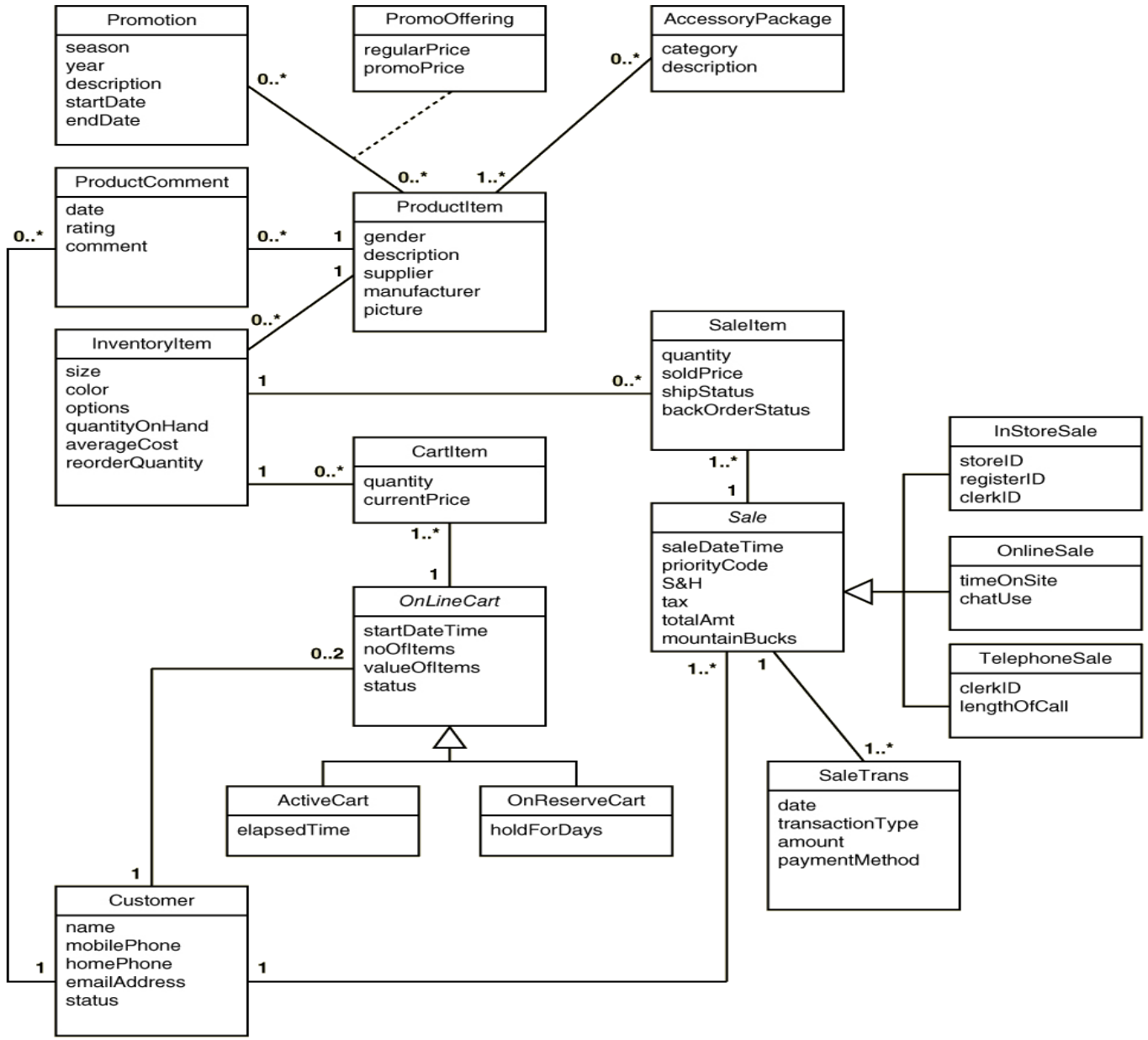
Record: 1 of 12 No Filter Search

Designing Relational Databases

Based on the Domain Model Class Diagram

1. Create a table for each class
2. Choose a primary key for each table (invent one, if necessary)
3. Add foreign keys to represent one-to-many associations
4. Create new tables to represent many-to-many associations
5. Represent classification hierarchies
6. Define referential integrity constraints
7. Evaluate schema quality and make necessary improvements
8. Choose appropriate data types
9. Incorporate integrity and security controls

RMO Classes


Promotion
season
year
description
startDate
endDate
PromoOffering
regularPrice
promoPrice
AccessoryPackage
category
description
ProductComment
date
rating
comment
ProductItem
gender
description
supplier
manufacturer
picture
InventoryItem
size
color
options
quantityOnHand
averageCost
reorderQuantity
SaleItem
quantity
soldPrice
shipStatus
backOrderStatus
CartItem
quantity
currentPrice
Sale
saleDateTime
priorityCode
S&H
tax
totalAmt
mountainBucks
InStoreSale
storeID
registerID
clerkID
OnlineSale
timeOnSite
chatUse
TelephoneSale
clerkID
lengthOfCall
OnLineCart
startDateTime
noOfItems
valueOfItems
status
ActiveCart
elapsedTime
OnReserveCart
holdForDays
SaleTrans
date
transactionType
amount
paymentMethod
Customer
name
mobilePhone
homePhone
emailAddress
status
0..*
1
1..*
0..2

Initial Set of Tables Based on RMO Domain Classes

Table	Attributes
AccessoryPackage	Category, Description
CartItem	Quantity, CurrentPrice
Customer	Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	StartDateTime, NumberOfItems, ValueOfItems, Status, ElapsedTime, HoldForDays
ProductComment	Date, Rating, Comment
ProductItem	Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	RegularPrice, PromoPrice
Promotion	Season, Year, Description, StartDate, EndDate
Sale	SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks, StoreID, RegisterID, ClerkID, TimeOnSite, ChatUse, LengthOfCall
SaleItem	Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	Date, TransactionType, Amount, PaymentMethod

Initial Set of Tables Based on RMO Domain Classes

- Choose a primary key for each table (invent one, if necessary)

Table	Attributes
AccessoryPackage	**AccessoryPackageID**, Category, Description
CartItem	**CartItemID**, Quantity, CurrentPrice
Customer	**AccountNumber**, Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	**InventoryItemID**, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	**OnlineCartID**, StartDateTime, NumberOfItems, ValueOfItems, Status, ElapsedTime, HoldForDays
ProductComment	**ProductCommentID**, Date, Rating, Comment
ProductItem	**ProductItemID**, Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	**PromoOfferingID**, RegularPrice, PromoPrice
Promotion	**PromotionID**, Season, Year, Description, StartDate, EndDate
Sale	**SaleID**, SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks, StoreID, RegisterID, ClerkID, TimeOnSite, ChatUse, LengthOfCall
SaleItem	**SaleItemID**, Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	**SaleTransactionID**, Date, TransactionType, Amount, PaymentMethod

Representing Associations

- One-to-Many – Add primary key attribute of the “one” class to the “many” class as a foreign key
- Many-to-Many –
  - With an Association Class – Add primary keys of endpoint classes as foreign keys and as candidate keys. May also become primary key
  - Without an Association Class – Create new table. Add primary keys of endpoint classes as foreign keys and as candidate keys.

Initial Set of Tables Based on RMO Domain Classes

- With Foreign keys added (in *italics*)

Table	Attributes
AccessoryPackage	**AccessoryPackageID**, Category, Description
CartItem	**CartItemID**, *InventoryItemID*, *OnlineCartID*, Quantity, CurrentPrice
Customer	**AccountNumber**, Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	**InventoryItemID**, *ProductItemID*, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	**OnlineCartID**, *CustomerAccountNumber*, StartDateTime, NumberOfItems, ValueOfItems, Status, ElapsedTime, HoldForDays
ProductComment	**ProductCommentID**, *ProductItemID*, *CustomerAccountNumber*, Date, Rating, Comment
ProductItem	**ProductItemID**, Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	**PromoOfferingID**, RegularPrice, PromoPrice
Promotion	**PromotionID**, Season, Year, Description, StartDate, EndDate
Sale	**SaleID**, *CustomerAccountNumber*, SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks, StoreID, RegisterID, ClerkID, TimeOnSite, ChatUse, LengthOfCall
SaleItem	**SaleItemID**, *InventoryItemID*, *SaleID*, Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	**SaleTransactionID**, *SaleID*, Date, TransactionType, Amount, PaymentMethod

Association Class

- PromoOfferingID ,CartItemID and SaleItemID can be discarded. **Why?**

Table	Attributes
AccessoryPackage	**AccessoryPackageID,** AccessoryCategory, Description
AccessoryPackageContents	***AccessoryPackageID, ProductItemID***
CartItem	***InventoryItemID, OnlineCartID,*** Quantity, CurrentPrice
Customer	**AccountNumber**, Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	**InventoryItemID**, *ProductItemID*, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	**OnlineCartID**, *CustomerAccountID*, StartDateTime, NumberOfItems, ValueOfItems, Status, ElapsedTime, HoldForDays
ProductComment	**ProductCommentID**, *ProductItemID*, *CustomerAccountNumber*, Date, Rating, Comment
ProductItem	**ProductItemID**, Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	***PromotionID, ProductItemID***, RegularPrice, PromoPrice
Promotion	**PromotionID**, Season, Year, Description, StartDate, EndDate
Sale	**SaleID**, *CustomerAccountNumber*, SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks, StoreID, RegisterID, ClerkID, TimeOnSite, ChatUse, LengthOfCall
SaleItem	***InventoryItemID, SaleID***, Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	**SaleTransactionID**, *SaleID*, Date, TransactionType, Amount, PaymentMethod

Representing Classification Hierarchies

- Just as a specialized class inherits the data and methods of a generalized class, a table representing a specialized class inherits some or all of its data from the table representing its generalized class. This inheritance can be represented in multiple ways, including the following:
  - Combining all the tables into a single table containing the superset of all classes
  - Using separate tables to represent the child classes, and using the primary key of the parent class table as the primary key of the child class tables
  - Some combination of the previous two methods

Final Tables

- Specialized subclasses included within OnlineCart and Sale tables

Table	Attributes
AccessoryPackage	**AccessoryPackageID**, AccessoryCategory, Description
AccessoryPackageContents	***AccessoryPackageID, ProductItemID***
CartItem	***InventoryItemID, OnlineCartID,*** Quantity, CurrentPrice
Customer	**AccountNumber**, Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	**InventoryItemID**, *ProductItemID*, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	**OnlineCartID**, *CustomerAccountID*, StartDateTime, NumberOfItems, ValueOfItems, Status, ElapsedTime, HoldForDays
ProductComment	**ProductCommentID**, *ProductItemID*, *CustomerAccountNumber*, Date, Rating, Comment
ProductItem	**ProductItemID**, Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	***PromotionID***, ***ProductItemID***, RegularPrice, PromoPrice
Promotion	**PromotionID**, Season, Year, Description, StartDate, EndDate
Sale	**SaleID**, *CustomerAccountNumber*, SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks, StoreID, RegisterID, ClerkID, TimeOnSite, ChatUse, LengthOfCall
SaleItem	***InventoryItemID***, ***SaleID***, Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	**SaleTransactionID**, *SaleID*, Date, TransactionType, Amount, PaymentMethod

Final Tables

- Specialized subclasses as separate tables

Table	Attributes
AccessoryPackage	**AccessoryPackageID,** AccessoryCategory, Description
AccessoryPackageContents	***AccessoryPackageID***, ***ProductItemID***
CartItem	***InventoryItemID***, ***OnlineCartID***, Quantity, CurrentPrice
Customer	**AccountNumber**, Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	**InventoryItemID**, *ProductItemID*, Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	**OnlineCartID**, *CustomerAccountID*, StartDateTime, NumberOfItems, ValueOfItems, Status, ElapsedTime, HoldForDays
ActiveCart	***OnlineCartID***, ElapsedTime
OnReserveCart	***OnlineCartID***, HoldForDays
ProductComment	**ProductCommentID**, ProductItemID,*CustomerAccountNumber*, Date, Rating, Comment
ProductItem	**ProductItemID**, Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	***PromotionID***, ***ProductItemID***, RegularPrice, PromoPrice
Promotion	**PromotionID**, Season, Year, Description, StartDate, EndDate
Sale	**SaleID**, *CustomerAccountNumber*, SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks
InStoreSale	***SaleID***, StoreID, RegisterID, ClerkID
OnlineSale	***SaleID***, TimeOnSite, ChatUse
TelephoneSale	***SaleID***, ClerkID, LengthOfCall
SaleItem	***InventoryItemID***, ***SaleID***, Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	**SaleTransactionID**, *SaleID*, Date, TransactionType, Amount, PaymentMethod

Designing Relational Databases- Referential Integrity and Schema Quality

- Referential integrity - a consistent state among foreign key and primary key values
- Referential integrity constraint - a constraint, stored in the schema, that the DBMS uses to automatically enforce referential integrity

Designing Relational Databases- Referential Integrity and Normalization

- A normalized relational database schema has these features:
  - Flexibility or ease of implementing future data model changes
  - Lack of redundant data
  - Protects against insertion, deletion and update anomalies
- Normalization - a formal technique for evaluating and improving the quality of a relational database schema
  - First Normal Form –
  - Second Normal Form –
  - Third Normal Form –

First Normal Form

- A table is in first normal form if every field contains only one value. All attributes values must be atomic.
  - Not multiple values in an attribute

SSN	Name	Department	Salary	Dependents
111-22-3333	Mary Smith	Accounting	40,000	John, Alice, Dave
222-33-4444	Jose Pena	Marketing	50,000	---
333-44-5555	Frank Collins	Production	35,000	Jan, Julia

  - Not varying number of columns

SSN	Name	Department	Salary	Dependent	Dependent	Dependent
111-22-3333	Mary Smith	Accounting	40,000	John	Alice	Dave
222-33-4444	Jose Pena	Marketing	50,000			
333-44-5555	Frank Collins	Production	35,000	Jan	Julia	

In-Class Activity 1 - First Normal Form Solution

– Solution is to put multivalued attribute in a separate table.

SSN	Name	Department	Salary
111-22-3333	Mary Smith	Accounting	40,000
222-33-4444	Jose Pena	Marketing	50,000
333-44-5555	Frank Collins	Production	35,000

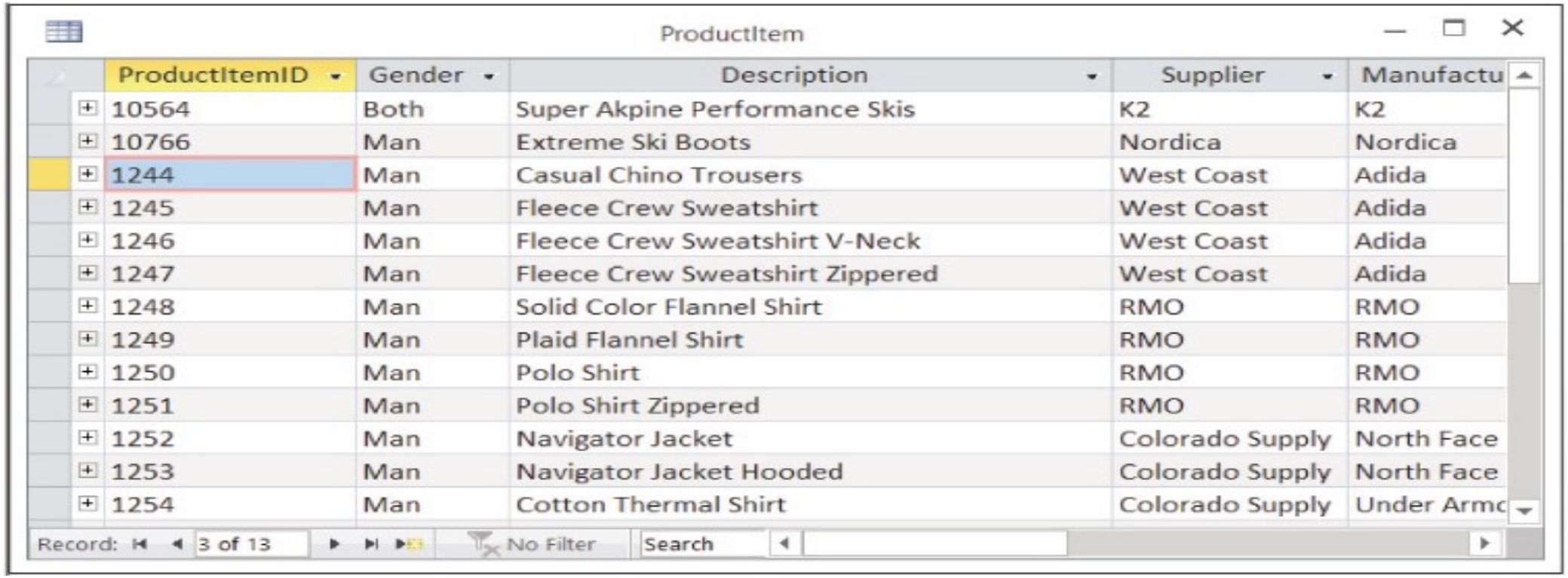
RecordID	SSN	Dependent
1	111-22-3333	John
2	111-22-3333	Alice
3	111-22-3333	Dave
4	333-44-5555	Jan
5	333-44-5555	Julia

Functional Dependency

- A relationship between attributes such that the values in the first attribute (or set) always determine the values in the second attribute (or set)
- *Attribute B is functionally **dependent** on attribute A if for each value of attribute A there is only one corresponding value of attribute B.*
  - *Written as FD: A → B.*
  - *Also stated as A functionally **determines** B*

Functional Dependency Example

- ProductID → Supplier
- But **NOT** Supplier → ProductID



ProductItem

ProductItemID	Gender	Description	Supplier	Manufactu
10564	Both	Super Akpine Performance Skis	K2	K2
10766	Man	Extreme Ski Boots	Nordica	Nordica
1244	Man	Casual Chino Trousers	West Coast	Adida
1245	Man	Fleece Crew Sweatshirt	West Coast	Adida
1246	Man	Fleece Crew Sweatshirt V-Neck	West Coast	Adida
1247	Man	Fleece Crew Sweatshirt Zippered	West Coast	Adida
1248	Man	Solid Color Flannel Shirt	RMO	RMO
1249	Man	Plaid Flannel Shirt	RMO	RMO
1250	Man	Polo Shirt	RMO	RMO
1251	Man	Polo Shirt Zippered	RMO	RMO
1252	Man	Navigator Jacket	Colorado Supply	North Face
1253	Man	Navigator Jacket Hooded	Colorado Supply	North Face
1254	Man	Cotton Thermal Shirt	Colorado Supply	Under Armc

Record: 3 of 13 No Filter Search

Second Normal Form

- A table is in Second Normal Form if it is First Normal Form and each non-key attribute is only functionally dependent on the entire primary key.
  - A table violates 2NF when a non-key attribute is functionally dependent on only part of the primary key. This situation only arises with tables that have multiple attribute keys.

Example

- PromoOffering table is **NOT** in 2NF
  - PromotionID, ProductItemID → PromoPrice
  - ProductItemID → RegularPrice -- Violation of 2NF

PromoOffering1NF

PromotionID	ProductItemID	RegularPrice	PromoPrice	P
1	10564	$599.99	$529.99	
1	10766	$399.99	$339.99	
2	10564	$599.99	$449.00	
2	10766	$399.99	$299.00	
2	1250	$49.99	$29.00	
2	1251	$49.99	$29.00	

Record: 6 of 6 No Filter Search

NOTE :A product can be part of multiple promotions at the same time. Although a product's promotional price can be different in different promotions, its regular price is the same whether it participates in one promotion, three promotions, or none.

- **Solution ?**

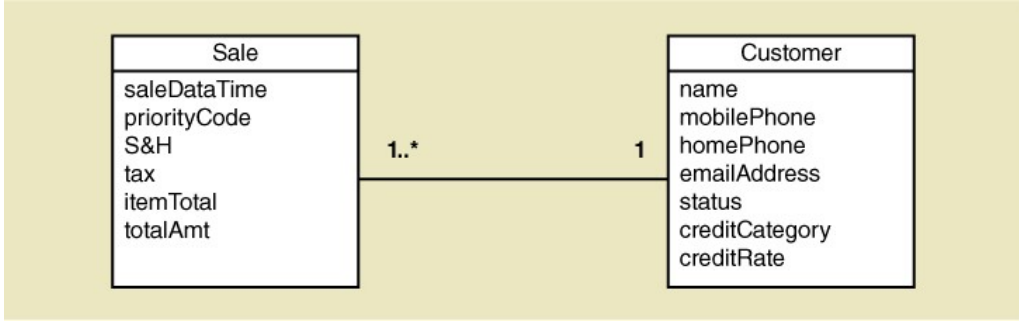
  To remove RegularPrice from this table and place it in another table, in this case ProductItem table.

Third Normal Form

- A table is in Third Normal Form if it is in 2NF and NO non-key attribute (or set) is functionally dependent on any other non-key attribute (or set)
  - In other words, no FDs among any non-key attributes

Example

- This version of Sale table **violates** 3NF
  - Shipping + Tax +Item Total = TotalAmt
  - i.e., FD: Shipping, Tax, ItemTotal → TotalAmt



SaleModified

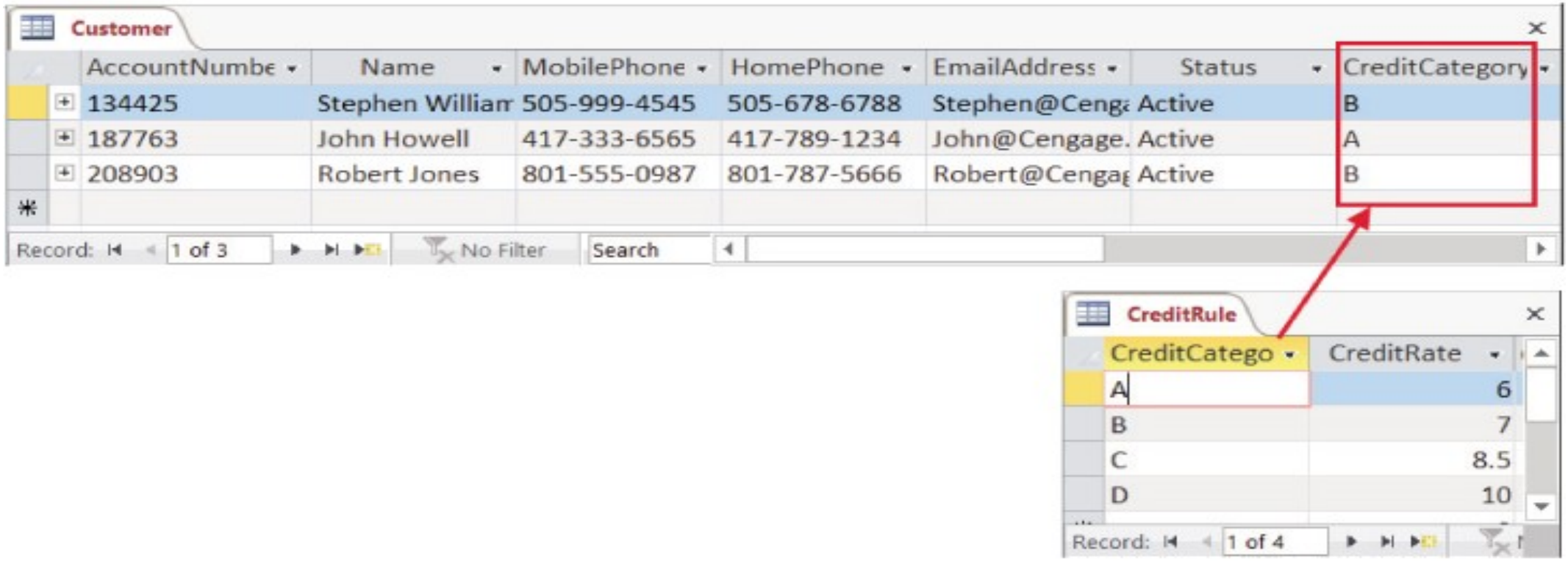
SaleID	SaleDate1	PriorityC	Shipping	Tax	ItemTotal	TotalAmt	CustomerAccountN
841152	9/1/2012		$8.50	$0.00	$91.35	$99.85	134425
841153	9/2/2012		$6.00	$0.00	$28.00	$34.00	187763
*			$0.00	$0.00	$0.00		

Record: 1 of 2 No Filter Search

- Solution is to remove TotalAmt. It is not needed

Third Normal Form

- Another solution is to move offending attribute to a ***new table***.
  - Violation = Customer table had CreditCategory and CreditRate
  - Solution = Make new table of CreditRule with CreditRate



In-Class Activity 2

Given the database table of employees and their employment, normalize the table so that it is in third normal form (3NF).

Hint: Look for functional dependencies.

Emp#	Employee Name	Job Title	Wage Range	Date Promoted	Supervisor Emp#	Supervisor Name
876	W. Johnson	Machinist Pipe Fitter Worker	25.00–45.00 18.00–30.00 12.00–25.00	June 1, 2011 May 15, 2006 July 1, 2002	450	B. Noch
651	A. Hansen	Welder Worker	20.00–35.00 12.00–25.00	Aug 1, 2012 June 15, 2009	335	P. Williams

In-Class Activity 2- Solution

- Note: The data in the table is not sufficient to analyze for all the functional dependencies that exist. However, using our experience (we are domain experts), we can assume the following.

EMP#	Employee Name	Job Title	Wage Range	Date Promoted	Supervisor Emp#	Supervisor Name
876	W. Johnson	Machinist	25.00-45.00	June 1, 2011	450	B. Noch
875	W. Johnson	Pipe Fitter	18.00-30.00	May 1, 2006	450	B. Noch
876	W. Johnson	Worker	12.00-25.00	July 1, 2002	450	B. Noch
651	A. Hansen	Welder	20.00-35.00	Aug 1, 2012	335	P. Williams
651	A. Hansen	Worker	12.00-25.00	June 15, 2009	335	P. Williams

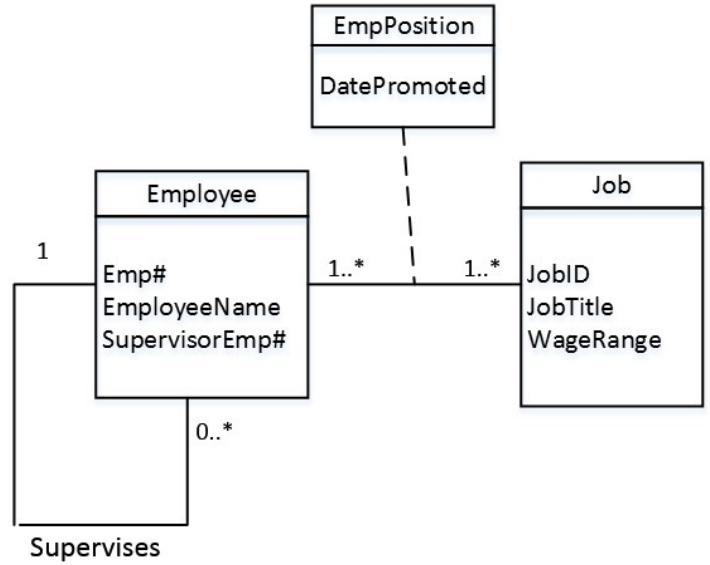
Key to this table is Emp#
EmployeeName is FD on the key.
JobTitle appears to be multiply occurring. Violates 1NF
WageRange appears to be FD on JobTitle. Violates 3NF
DatePromoted appears to be FD on EmployeeName and JobTitle. Violates 3NF
SupervisorEmp# is FD on key.
SupervisorName is FD on SupervisorEmp#. Violates 3NF

In-Class Activity 2- Solution (Cont.)

Note: The data in the table is not sufficient to analyze for all the functional dependencies that exist. However, using our experience (we are domain experts), we can assume the following.

- Key to this table is Emp#
- EmployeeName is FD on the key.
- JobTitle appears to be multiply occurring. Violates 1NF
- WageRange appears to be FD on JobTitle. Violates 3NF
- DatePromoted appears to be FD on EmployeeName and JobTitle. Violates 3NF
- SupervisorEmp# is FD on key.
- SupervisorName is FD on SupervisorEmp#. Violates 3NF

Table	Attributes	Comments
Supervisor	SupervisorEmp#, Name	This table is redundant to Employee table. Can be represented by a unary relationship.
Job	JobID, JobTitle, WageRange	Added JobID as key
Employee	Emp#, EmployeeName, SupervisorEmp#	This is all that is required.
EmpPosition	Emp#, JobID, DatePromoted	This is an association class.



Data Anomalies

If a database is designed poorly and not normalized properly, anomalies can happen.

- Insertion anomaly
  - Occurs when certain attribute value cannot be inserted into a database without the presence of other attributes.
- Deletion anomaly
  - Occurs when a certain value (s) are lost due to deletion of other attributes.
- Update anomaly
  - Occurs when one or multiple instances of duplicated data is updated but not all of them.

Data Types

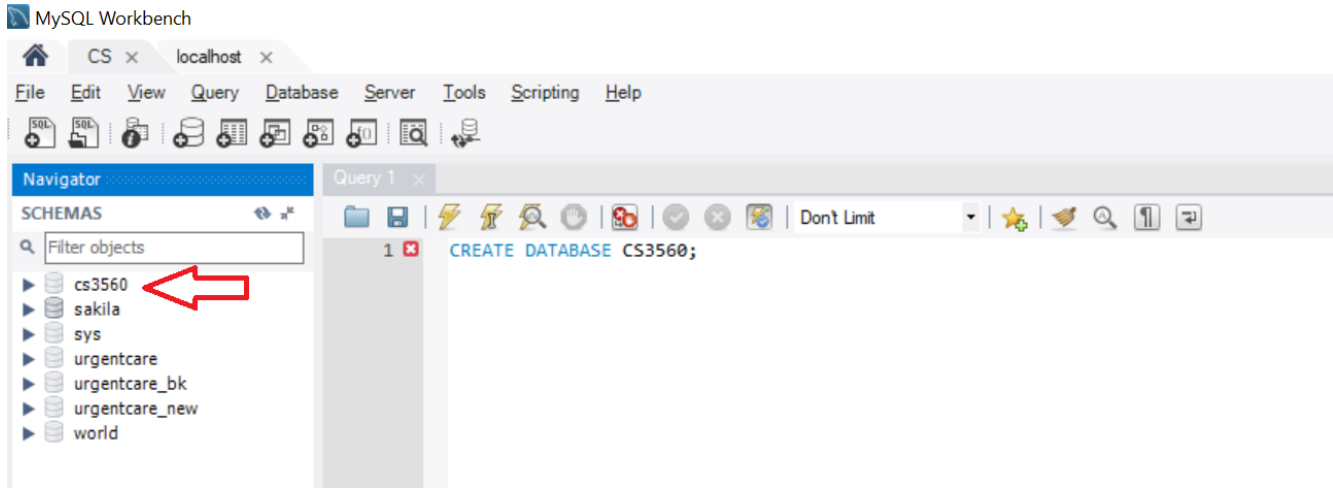
- The data type defines the storage format and allowable content of an attribute (field)
- Primitive data types – are directly supported by computer hardware and programming languages
  - i.e : integers, single characters, and real numbers (floating- point numbers).
- Complex data types – combinations or compositions of primitive data types supported by programming languages, operating systems, and DBMSs.
  - i.e: arrays and tables, strings (character arrays), dates, times, currency (money), audio streams, still images, motion video streams, and Uniform Resource Locators (URLs or Web links).

Example of Data Types –available in MSSQL Server RDBMS

Type(s)	Description
datetimeoffset	Date, time, and time zone
int, small int, and bigint	Whole numeric values
float and real	Numeric values with fractional quantities
money	Currency values and related symbols (e.g., $ and €)
nchar and nvarchar	Fixed- and variable-length Unicode string
varbinary	Variable-length byte sequence up to 2GB
xml	XML document up to 2GB

Basic SQL Operation

- Create a Database
  - `CREATE DATABASE CS3560;`



- Use a Database
  - `USE CS3560;`

- Drop a Database
  - `DROP DATABASE CS3560;`

Basic SQL Operation

- Create a table
  - ```
    CREATE TABLE table_name (
        column1 datatype,
        column2 datatype,
        column3 datatype,
       ....
    );
    ```

Example:

  - ```
    Create Table Student(
     student_ID INT (6),
     f_name VARCHAR (50),
     l_name VARCHAR (50),
     major VARCHAR (20),
     PRIMARY KEY(student_ID)
    );
    ```

- Drop a table
  - ```
    DROP TABLE table_name;    i.e. DROP TABLE student;
    ```
- Alter a table (Add, Modify or Drop a column)

Add a new column :

```
ALTER TABLE table_name ADD column_name datatype; i.e.  ALTER TABLE student ADD DOB VARCHAR(10);
```

Modify a column data type:

```
ALTER TABLE table_name  MODIFY COLUMN column_name datatype;
```

i.e. ALTER TABLE student MODIFY COLUMN DOB INT(8);

Drop a column data type:

```
ALTER TABLE table_name  DROP COLUMN column_name;
```

i.e. ALTER TABLE student DROP COLUMN DOB;
```

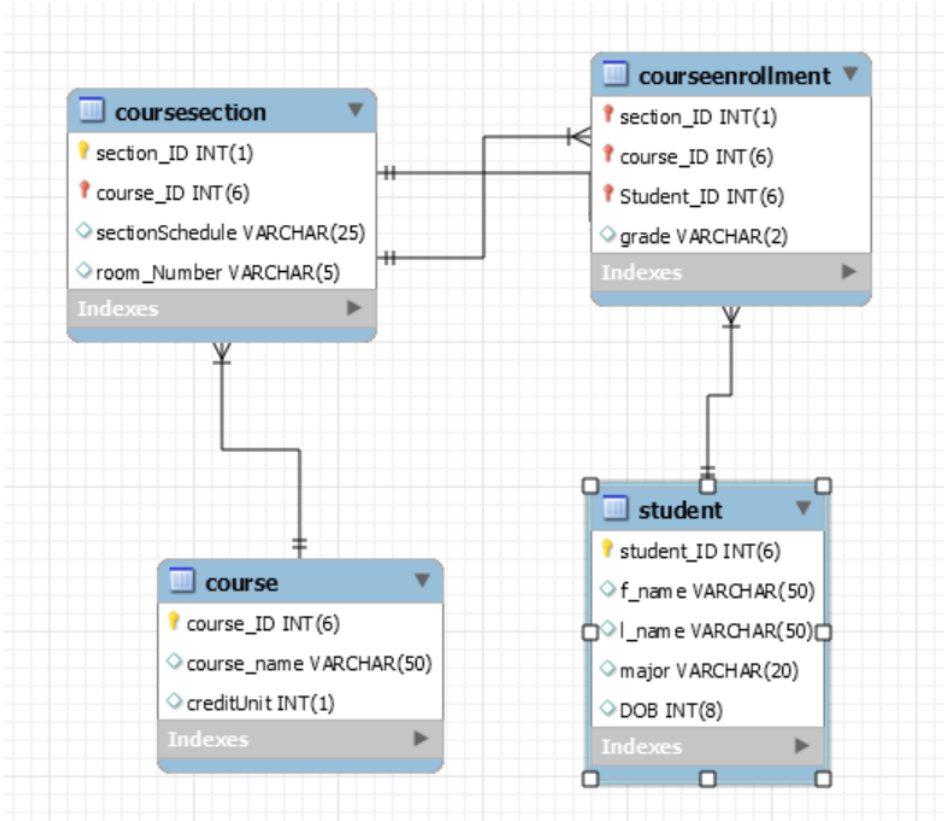
Basic SQL Operation

- Example of creating P.K and F.K. (Course and CourseSection tables)

```
Create Table Course (
course_ID VARCHAR(6),
PRIMRY KEY (course_ID),
Course_name VARCHAR(50),
creditUnit INT(1)
);

Create Table CourseSection(
section_ID INT (1),
course_ID VARCHAR(6),
sectionSchedule VARCHAR (25),
room_Number VARCHAR(5),
FOREIGN KEY (course_ID) REFERENCES Course(course_ID),
PRIMARY KEY (section_ID , course_ID));

Create Table CourseEnrollment(
section_ID INT (1),
course_ID VARCHAR(6),
Student_ID INT (6),
grade VARCHAR(2),
FOREIGN KEY (student_ID) REFERENCES student(student_ID),
FOREIGN KEY (course_ID) REFERENCES CourseSection(course_ID),
FOREIGN KEY (section_ID) REFERENCES courseSection(section_ID),
PRIMARY KEY (section_ID , course_ID, student_ID));
```



Basic SQL Operation - CRUD

- Create (Insert)
  - `INSERT INTO table_name (Att1, Att 2, ....) VALUES (value1, value2, ...);`
    - No need to specify attributes' names if all attribute values are specified based on table columns.
    - Multiple rows can be added by 1 `INSERT` query .
      - » `INSERT INTO table_name (Att1, Att2, ....) VALUES (value1, value2, ...), (value1, value2, ...), (value1, value2, ...),…. ;`

        i.e : `INSERT INTO Sale VALUES (123456,20210101, 3.52,0.00,12.57), (234567,20210201, 25.00, 10.25,110.18), (345678,20210301, 19.99,6.54,65.00);`
- Report (Select)
  - `SELECT [ DISTINCT ]{ select-item-list | * } FROM table-or-view-list [ WHERE search-condition ]`

    `[ GROUP BY column-reference-list ]`

    `[ HAVING search-condition ]`

    `[ ORDER BY sort-item-list ]`
- Update (Update)
  - `UPDATE table_name`
    `SET att1 = value1, att2 = value2, ...`
    `WHERE condition;`
- Delete (Delete)
  - `DELETE FROM table_name WHERE condition;`

Basic SQL Operation - View

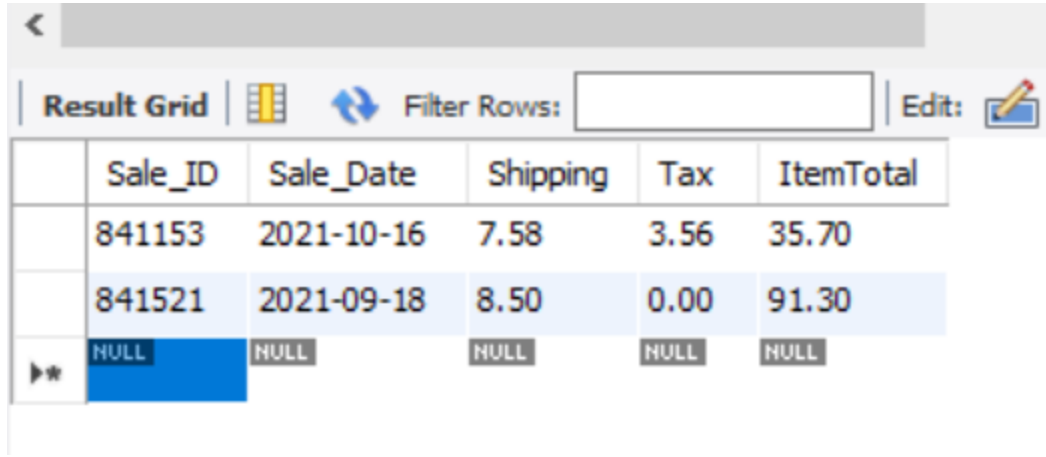
- View is a virtual subset of a table.
- It appears like a real table with a set of rows and columns.
- Can be queried like tables.
  - More complex views are restricted to “Select” only.
- Advantages
  - Simplicity
    - Structural
    - Query
  - Security
  - Independency
  - Data integrity
  - Consistency
- Disadvantages
  - Performance
  - Update restriction

Basic SQL Operation - View

Create a View

Assume there is a table “Sale”:

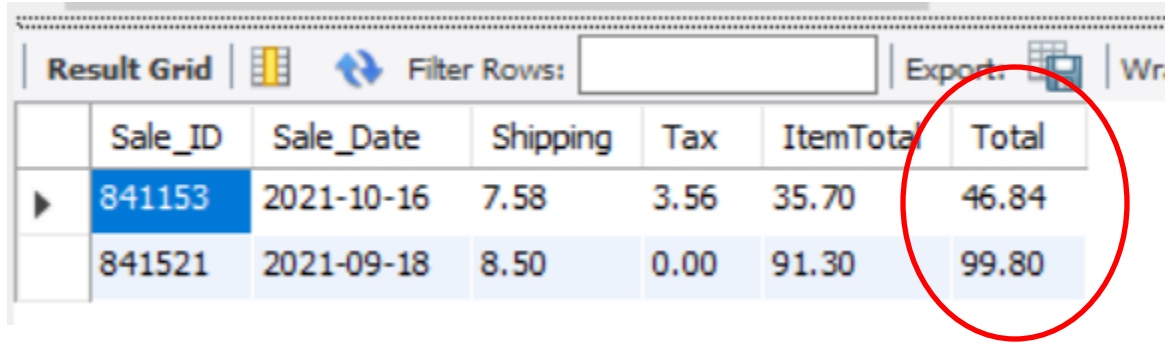
```
Create Table Sale(
Sale_ID INT (6),
Sale_Date DATE,
Shipping DECIMAL (5,2),
Tax DECIMAL(5,2),
ItemTotal DECIMAL (5, 2),
primary key (Sale_ID));
```



Sale_ID	Sale_Date	Shipping	Tax	ItemTotal
841153	2021-10-16	7.58	3.56	35.70
841521	2021-09-18	8.50	0.00	91.30
NULL	NULL	NULL	NULL	NULL

A view is created to calculate and show the total

```
Create VIEW SaleTotal AS Select
        *, Shipping+Tax+ItemTotal AS Total from Sale;
```



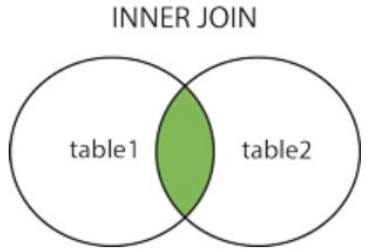
Sale_ID	Sale_Date	Shipping	Tax	ItemTotal	Total
841153	2021-10-16	7.58	3.56	35.70	46.84
841521	2021-09-18	8.50	0.00	91.30	99.80

Basic SQL Operation - JOIN

INNER JOIN

- The `INNER JOIN` keyword selects records that have matching values in both tables.

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```



INNER JOIN two tables - Example:

```
SELECT course.course_name , coursesection.section_ID, coursesection.sectionSchedule
FROM course INNER JOIN coursesection
ON course.course_ID = coursesection.course_ID;
```

INNER JOIN more than 2 tables - Example:

```
SELECT  concat(student.f_name , ' ' , student.l_name) AS name, coursesection.course_ID, course.course_name,
courseenrollment.grade  FROM student
INNER JOIN courseenrollment ON student.student_ID = courseenrollment.Student_ID
INNER JOIN coursesection ON coursesection.course_ID = courseenrollment.course_ID AND coursesection.section_ID =
courseenrollment.section_ID
INNER JOIN course ON course.course_ID = coursesection.course_ID;
```

Source:https://www.w3schools.com/MySQL/mysql_join_inner.asp

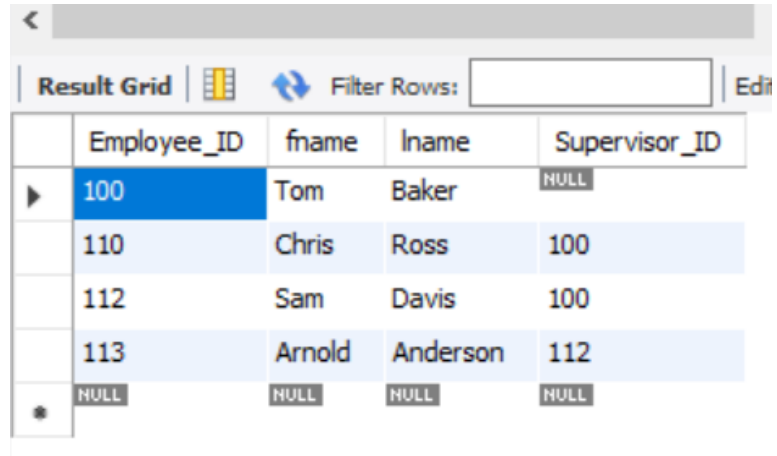
Basic SQL Operation - Self Join

- A self Join is a regular join, but the table is joined with itself..

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

Example: Let's create a table for Employee with some mock data as follow :

```
Create Table Employee(Employee_ID INT (3),
PRIMARY KEY (Employee_ID),
fname VARCHAR (50) NOT NULL,
lname VARCHAR (50) NOT NULL,
Supervisor_ID INT (3)
);
```

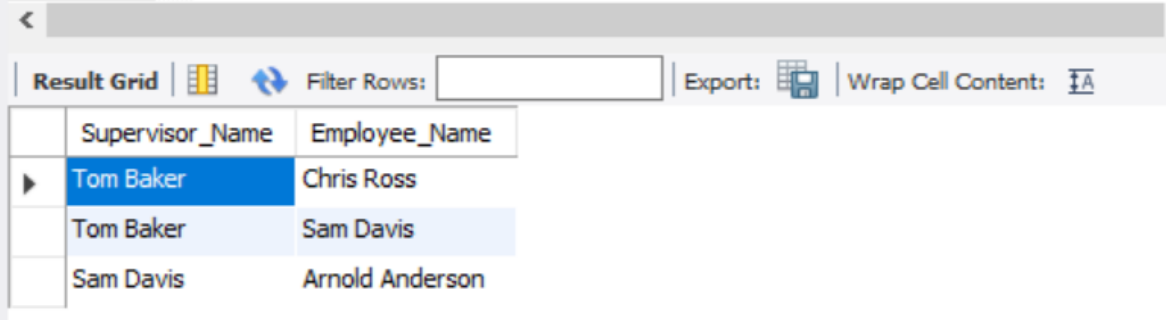


Employee_ID	fname	lname	Supervisor_ID
100	Tom	Baker	NULL
110	Chris	Ross	100
112	Sam	Davis	100
113	Arnold	Anderson	112
NULL	NULL	NULL	NULL

Self join :

```
SELECT concat(A.fname , ' ', A.lname)
AS Supervisor_Name,
concat(B.fname , ' ', B.lname) AS Employee_Name
FROM Employee A, Employee B
WHERE A.Employee_ID =  B.Supervisor_ID;
```

```
121  select * from Employee;
122
123  SELECT concat(A.fname , ' ', A.lname)
124  AS Supervisor_Name, concat(B.fname , ' ', B.lname)
125  AS Employee_Name
126  FROM Employee A, Employee B
127  WHERE A.Employee_ID =  B.Supervisor_ID;
128
```



Supervisor_Name	Employee_Name
Tom Baker	Chris Ross
Tom Baker	Sam Davis
Sam Davis	Arnold Anderson

Source:https://www.w3schools.com/MySQL

Some Useful Functions

COUNT(), AVG(), SUM()

```
SELECT COUNT/AVG/SUM(column_name)
FROM table_name
WHERE condition;
```

i.e:
```
SELECT course_ID, COUNT(section_ID) as NumberEnrolled
    FROM courseenrollment GROUP BY course_ID;
```

MAX() , MIN ()

```
SELECT MIN/MAX (column_name)
FROM table_name
WHERE condition;
```

i.e:
```
SELECT MAX(creditUnit), course_ID, course_name  FROM course;
```

LIKE and Wildcard characters:

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
- There are two wildcards often used in conjunction with the LIKE operator:
- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

i.e:
```
SELECT course_ID, course_name  FROM course WHERE course_name LIKE '%ign%';
SELECT course_ID, course_name  FROM course WHERE course_name NOT LIKE '_is%';
```

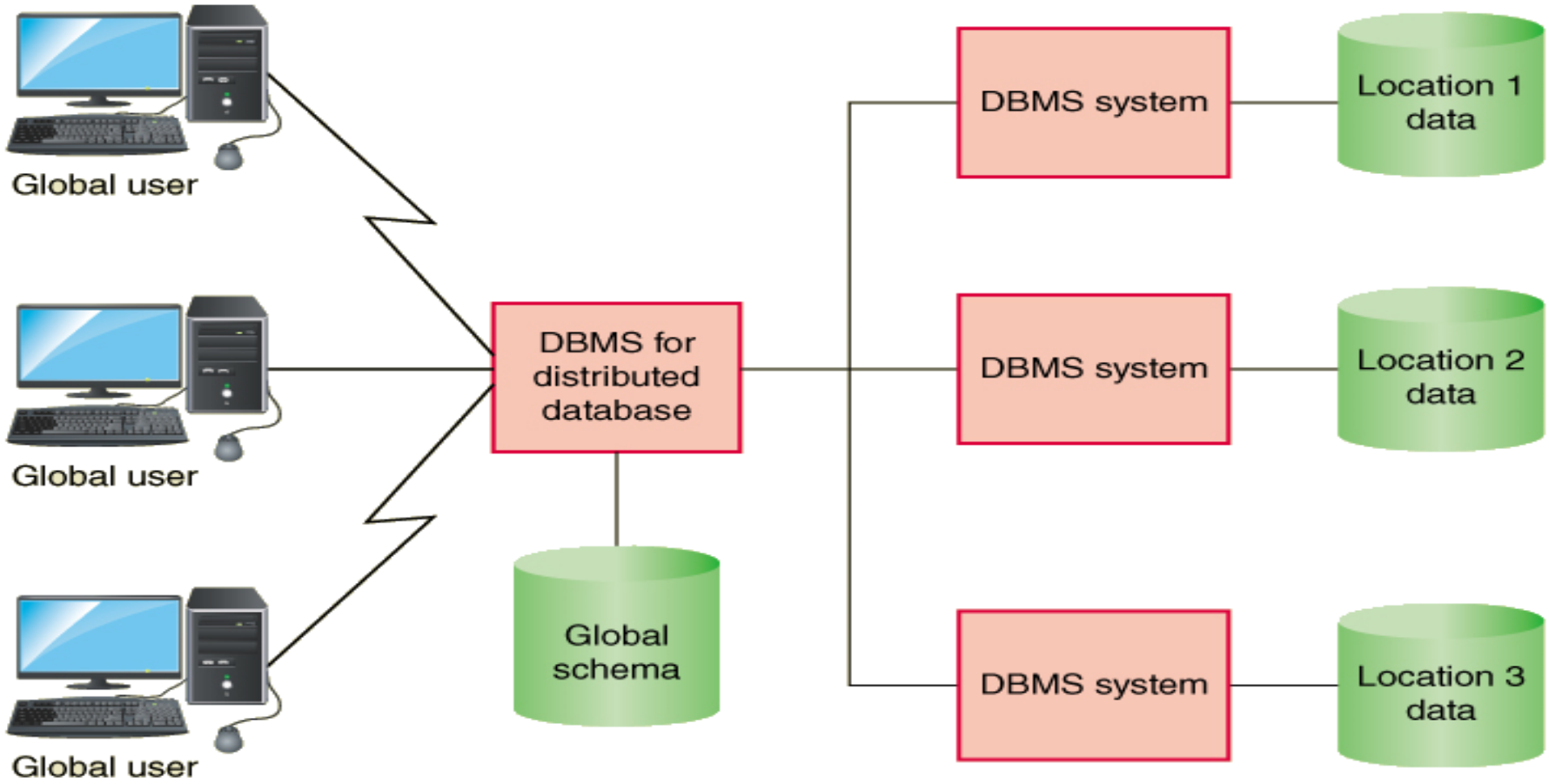
https://www.w3schools.com/MySQL

Distributed Database Architecture

- Decentralized database is stored at many locations but not requiring interconnectivity or synchronization
- Homogeneous distributed database is stored at multiple locations, with all locations using the same DBMS. Coordinated with a global schema
- Heterogeneous distribute database is stored at multiple locations and with different DBMS and may have local schemas.

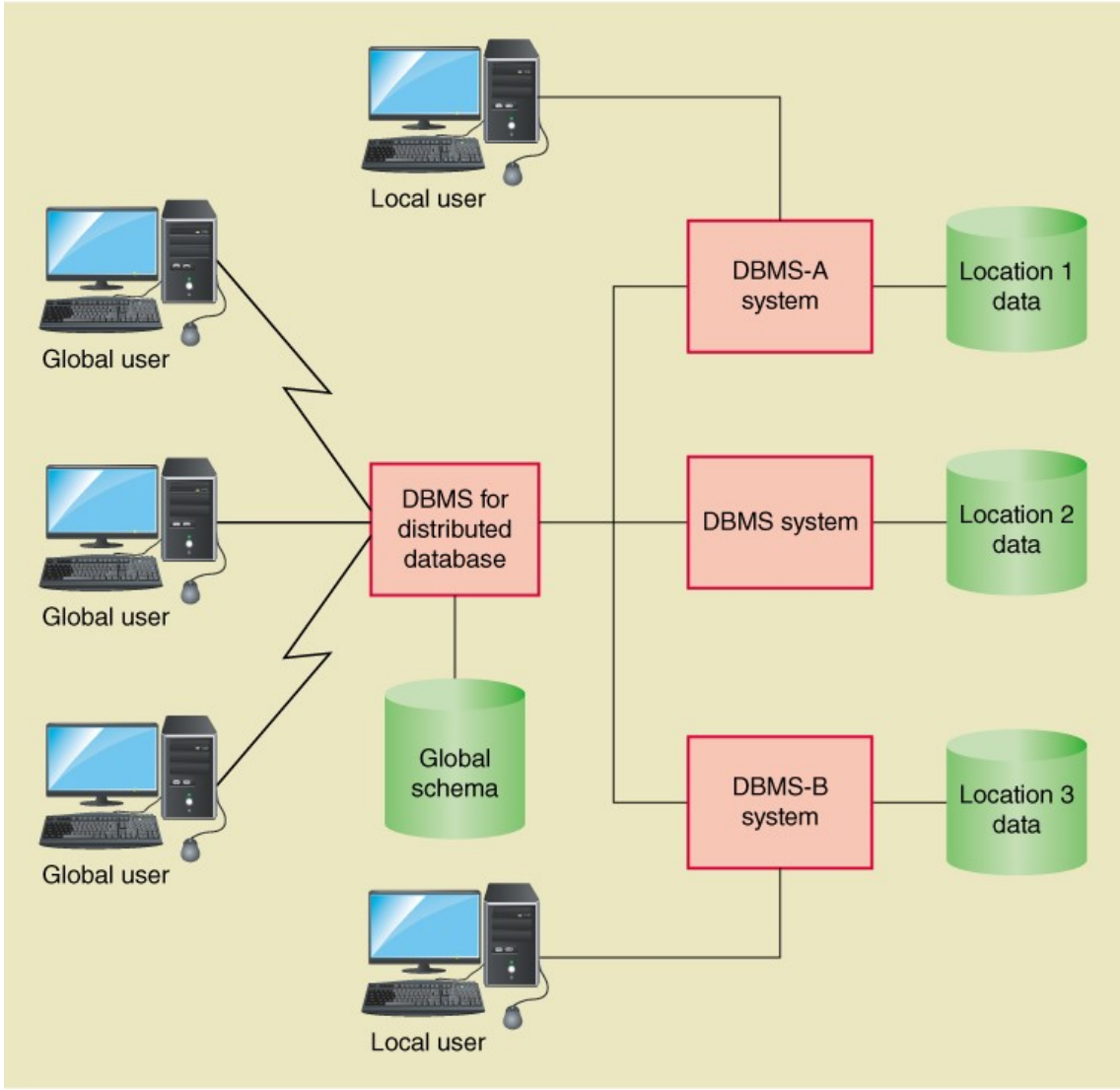
Homogeneous Distributed Database

- Access is through a common DBMS and schema



Heterogeneous Distributed Database

- Typical heterogeneous distributed database configuration



Implementation Approaches

- Data replication – each location has its own copy
  - Pros: independency, fast response time, availability
  - Cons: increased storage requirement, synchronization
  - Synchronization – updating every copy with changes made to every other copy

Implementation Approaches

- Horizontal Partition – different rows are stored at different locations.

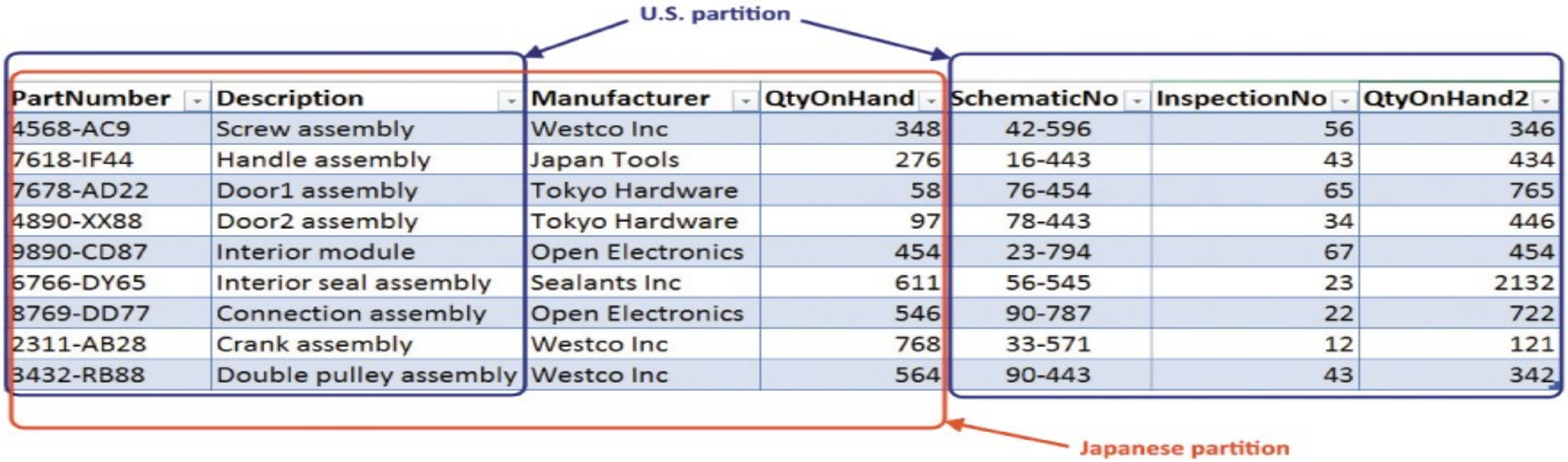
AcctNumb	LastName	FirstName	SSN	TypeOfAcct	Balance	DateLastActivity
01-85562-1	Jones	Bill	878-77-9890	Checking	$ 7,908.39	5/9/2014
01-85444-2	Johnson	Harold	676-44-3433	Checking	$25,698.33	5/2/2013
02-45443-2	Williams	Jonathon	343-44-2322	Checking	$ 3,938.77	4/4/2012
01-34999-1	Redd	Mary	898-79-3487	Savings	$12,898.71	12/2/2013
01-23989-2	Chun	Tun	233-59-6765	Savings	$ 8,932.67	1/8/2014
01-87889-4	Gang	Bao	322-48-3545	Checking	$ 568.33	3/4/2014
01-32339-2	Jiang	Rui	550-43-5454	Savings	$35,788.23	7/8/2014
02-39988-1	Ma	Shuo	343-98-2345	Checking	$ 1,893.55	8/23/2014

U.S. accounts

Hong Kong accounts

Implementation Approaches

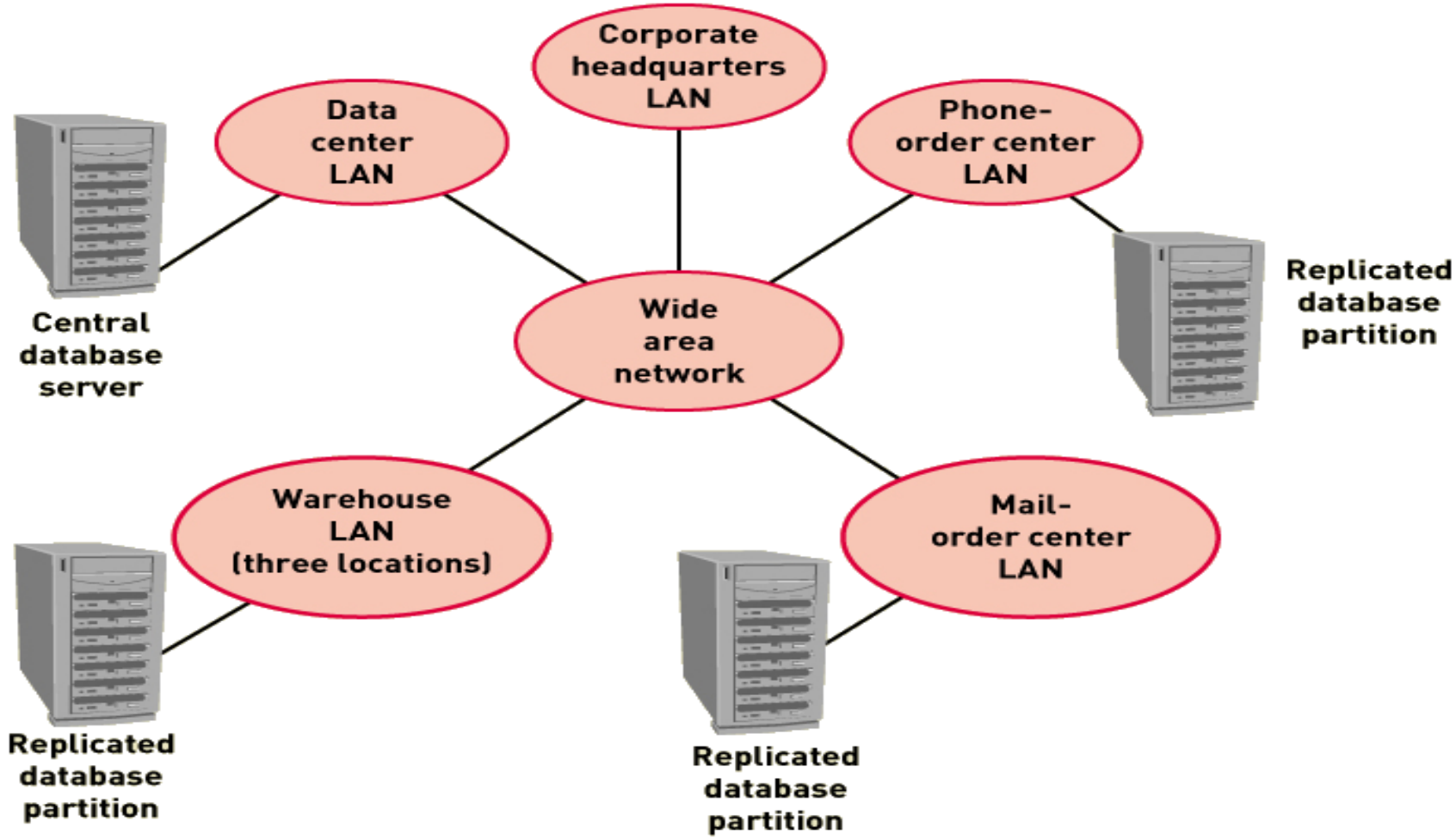
- Vertical Partition – Different columns are stored at different locations.
  - vertical partitioning of the database occurs more frequently when different functional areas must access the same database and the same data



PartNumber	Description	Manufacturer	QtyOnHand	SchematicNo	InspectionNo	QtyOnHand2
4568-AC9	Screw assembly	Westco Inc	348	42-596	56	346
7618-IF44	Handle assembly	Japan Tools	276	16-443	43	434
7678-AD22	Door1 assembly	Tokyo Hardware	58	76-454	65	765
4890-XX88	Door2 assembly	Tokyo Hardware	97	78-443	34	446
9890-CD87	Interior module	Open Electronics	454	23-794	67	454
6766-DY65	Interior seal assembly	Sealants Inc	611	56-545	23	2132
8769-DD77	Connection assembly	Open Electronics	546	90-787	22	722
2311-AB28	Crank assembly	Westco Inc	768	33-571	12	121
3432-RB88	Double pulley assembly	Westco Inc	564	90-443	43	342

- Combinations of replication, horizontal, and vertical

Architecture for RMO - Replicated and Partitioned Database



Protecting the Database

- Transaction Logging – a technique to record all updates including change, date, time, user
  - Helps to prevent fraud
  - Recovery mechanism for failures
- Concurrency and Update Controls
  - Transaction – a piece of work with several steps, either all must complete or none must be accepted
  - Database lock – technique to apply exclusive control to a portion of the database to one user at a time
  - Shared or read lock – a lock where multiple transactions (users) may read the data
  - Exclusive or write lock – a lock where only one transaction (user) may access the locked portion of the database

Summary

- Most modern information systems store data and access data using a database management systems (DBMS)
- The most common database model is a relational database (RDBMS), which is a collection of data stored in tables
- The relational database schema is developed based on the domain model class diagram Each class is represented as a table. One to many associations are represented by adding foreign keys
- Normalization is the process to produce high-quality databases without update, insertion or delete anomalies
- Distributed databases are necessary for very large databases
- Database locks permit concurrent use of databases