Database Management Systems

Chapter 1
Introduction

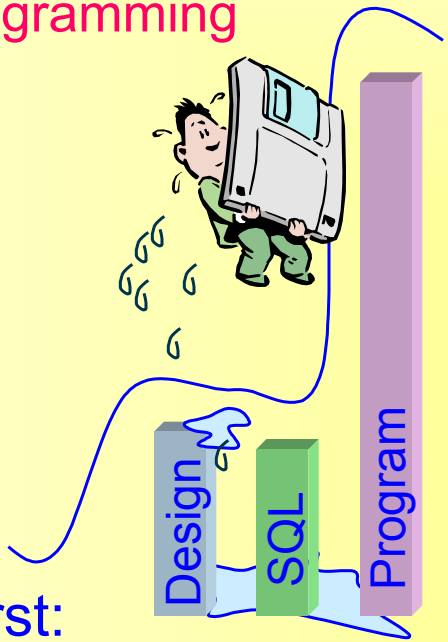
Jerry Post

DATABASE

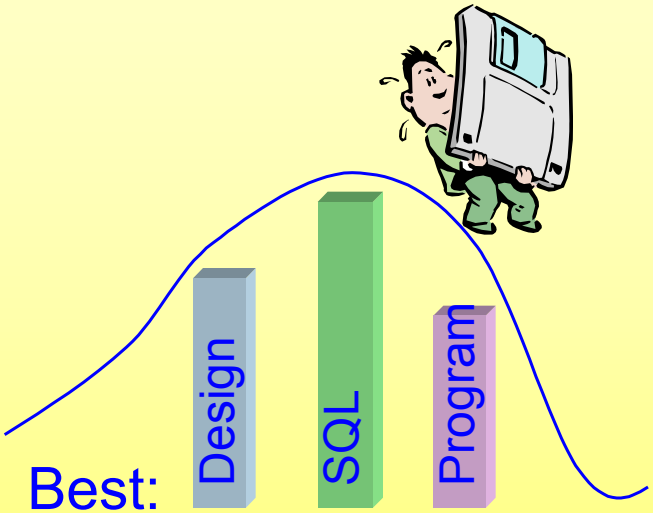
Goal: Build a Business Application

Tools:

- Database Design
- SQL (queries)
- Programming



Worst:
Compensate for poor design
and limited SQL with programming.



Best:
Spend your time
on design and SQL.

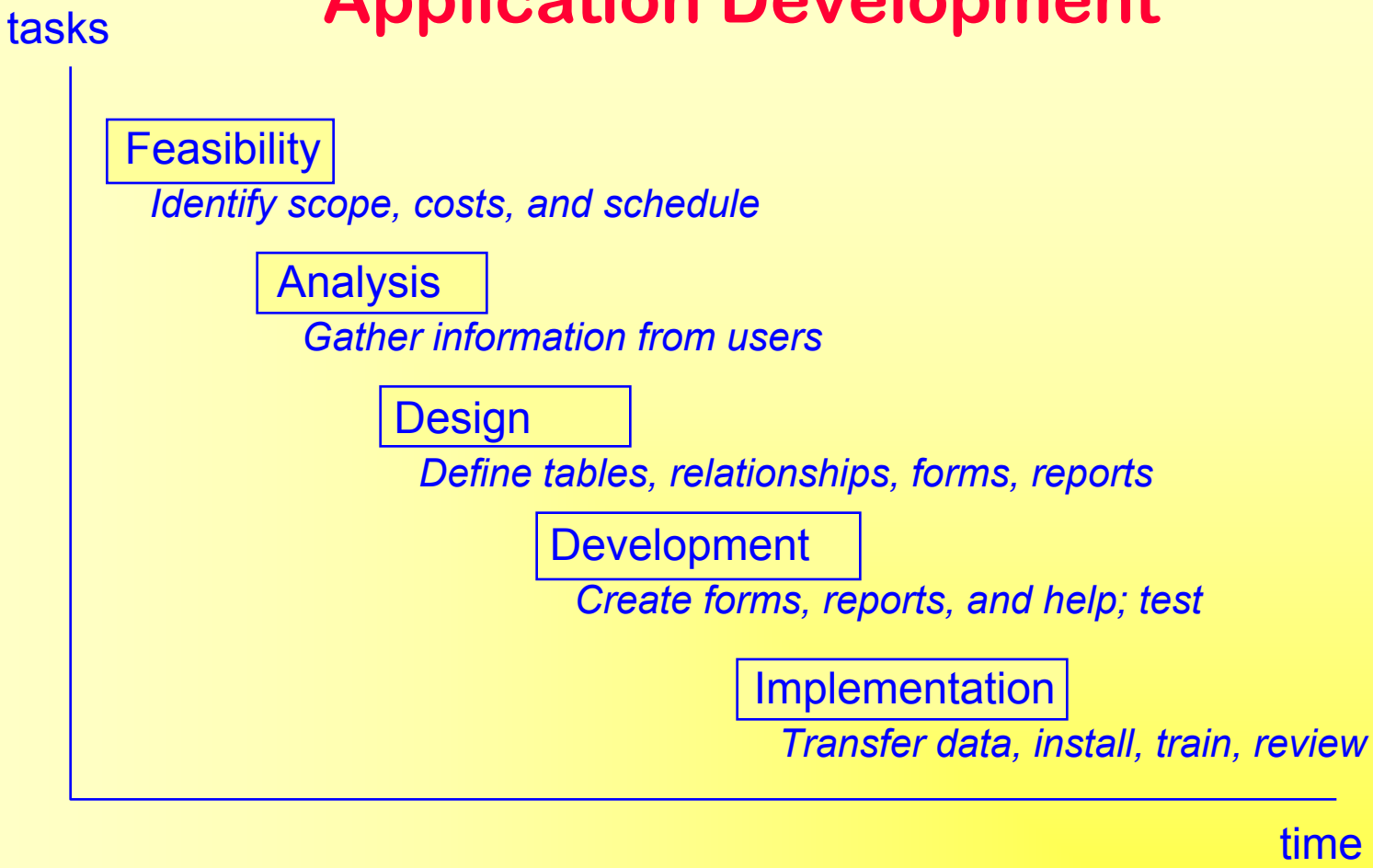
D A T A B A S E

DBMS: Database Management System

- Database
  - A collection of data stored in a standardized format, designed to be shared by multiple users.
- Database Management System
  - Software that defines a database, stores the data, supports a query language, produces reports, and creates data entry screens.

DATABASE

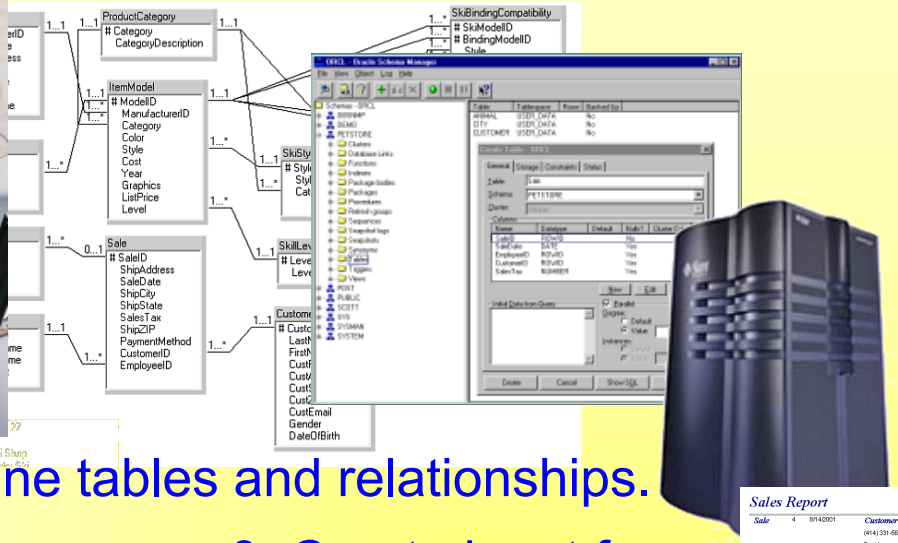
Application Development



D A T A B A S E

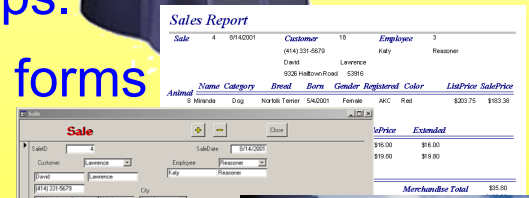
DBMS Application Design

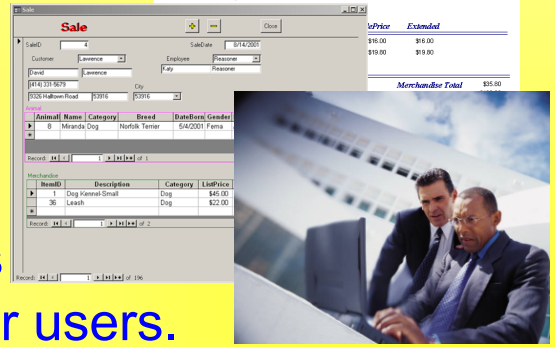
1. Identify business rules.



2. Define tables and relationships.

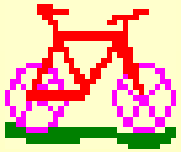
3. Create input forms and reports.





4. Combine as applications for users.

DATABASE

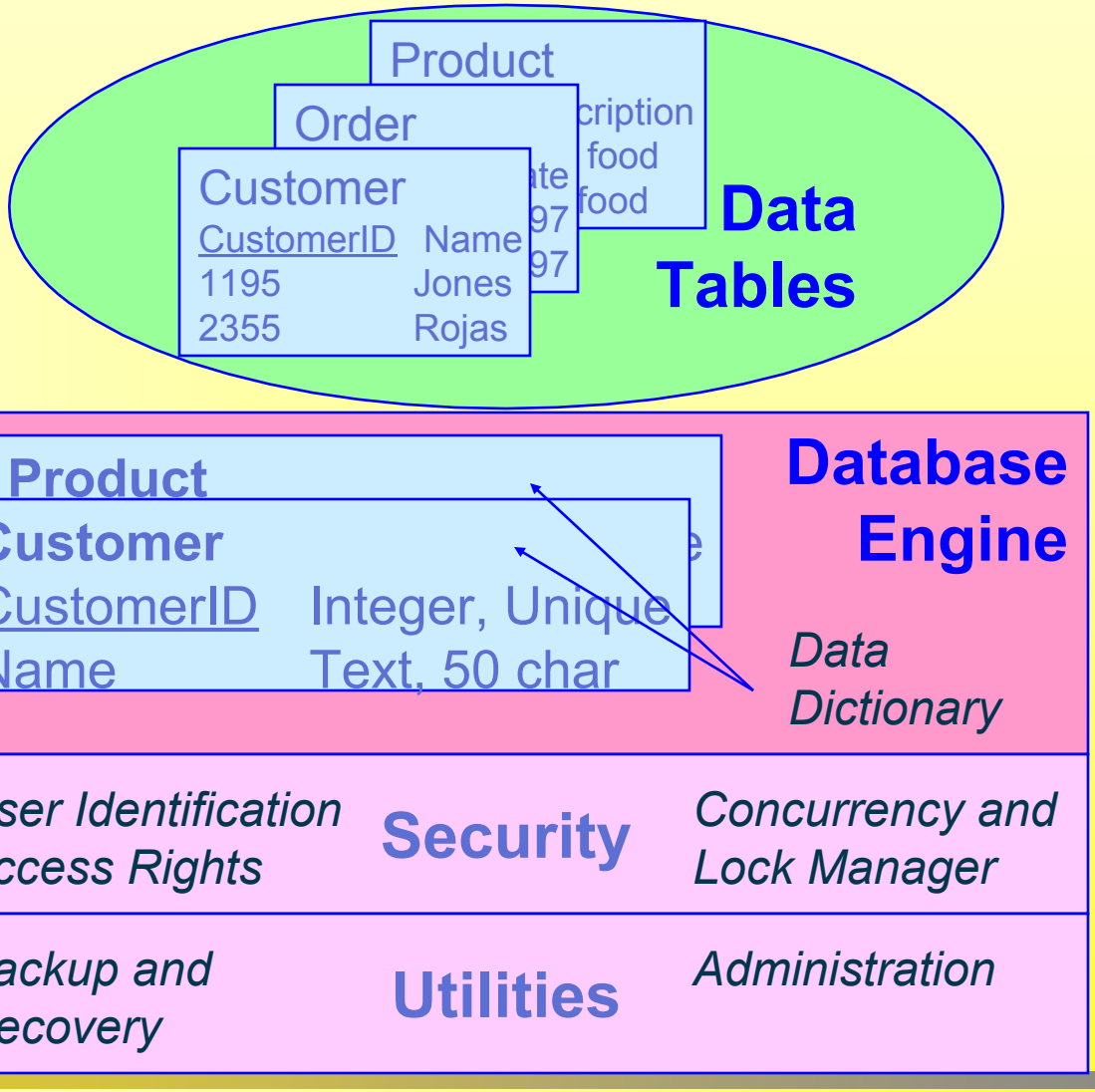


DBMS Features/Components

- Database engine
  - Storage
  - Retrieval
  - Update
- Query Processor
- Data dictionary
- Utilities
- Security
- Report writer
- Forms generator (input screens)
- Application generator
- Communications
- 3GL Interface

DATABASE

DBMS Engine, Security, Utilities



DATABASE

Database Tables (Access)

Sale : Table

SaleID	SaleDate	EmployeeID	CustomerID	SalesTax
4	8/14/2001	3	18	$14.62
5	10/31/2001	2	21	$6.86
6	9/15/2001	5	44	$9.82
7	2/10/2001	4	42	$12.31
8	3/10/2001	1	15	$17.58
9	2/10/2001	3	16	$2.81
10	11/1/2001	8	53	$7.83
11	12/24/2001	8	60	$3.67
12	8/15/2001	2	53	$1.19
13	1/30/2001	7	49	$14.81
14	9/18/2001	2	9	$3.56
15	7/20/2001	9	39	$1.13
16	9/18/2001	8	62	$12.96
17	2/12/2001	4	71	$16.31
18	12/21/2001	5	35	$14.95

Record: 1 of 196

SaleAnimal : Table

SaleID	AnimalID	SalePrice
4	8	$183.38
5	183	$114.30
6	58	$132.19
7	24	$147.58
8	42	$174.27
9	53	$46.80
10	9	$1.80
12	5	$19.80
13	162	$119.88
13	199	$100.00
15	13	$10.80
16	193	$216.05
17	11	$148.47
18	10	$150.11
19	47	$185.47

Record: 1

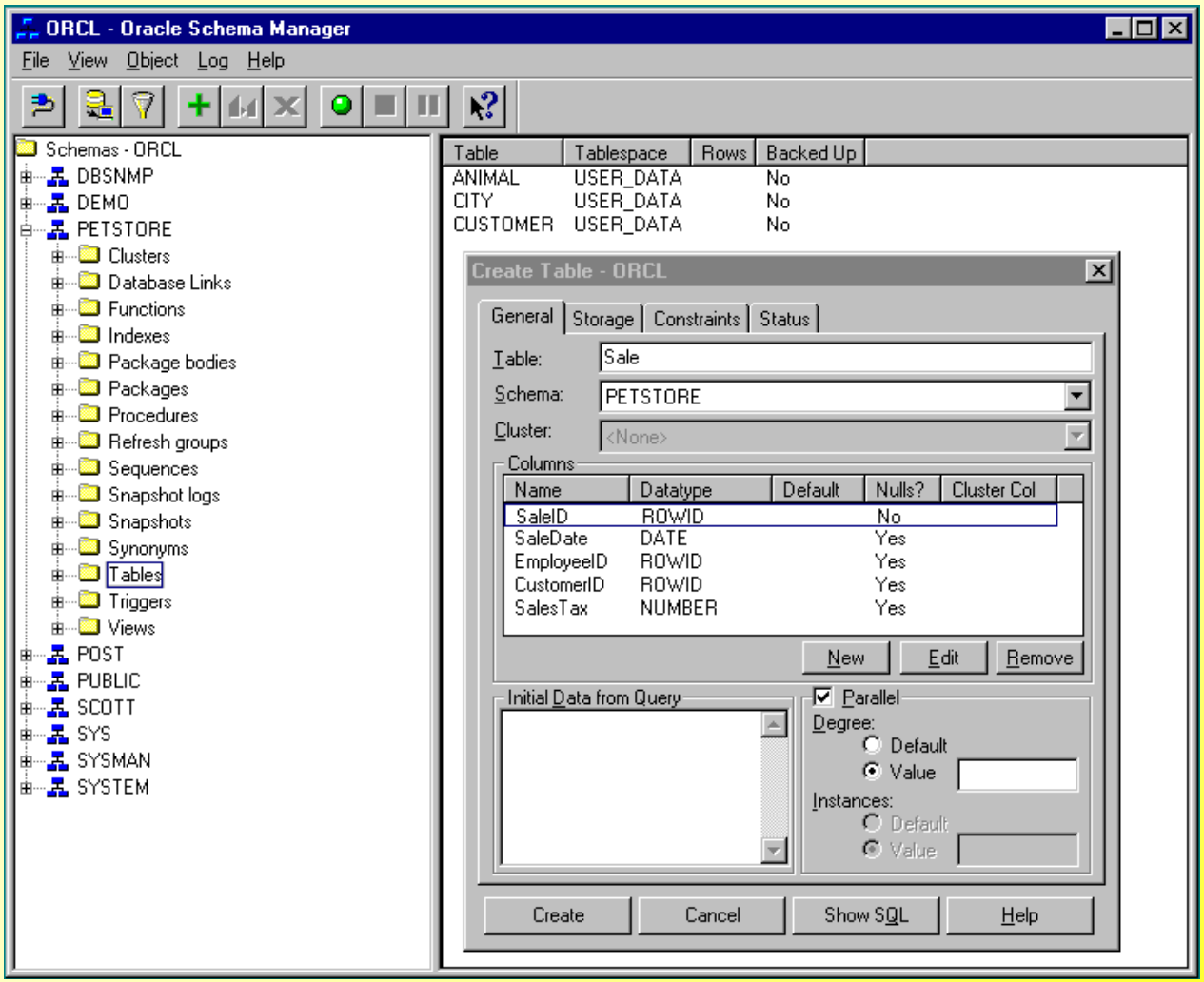
Animal : Table

AnimalID	Name	Category	Breed	DateBorn	Gender	Registered	Color
2		Fish	Angel	5/5/2001	Male		Black
4	Gary	Dog	Labrador Retriever	3/2/2001	Male	AKC	Spotted
5		Fish	Shark	1/1/2001	Female		Gray
6	Rosie	Cat	Oriental Shorthair	8/2/2001	Female	CFA	Gray
7	Eugene	Cat	Bombay	1/25/2001	Male	CFA	Black
8	Miranda	Dog	Norfolk Terrier	5/4/2001	Female	AKC	Red
9		Fish	Guppy	3/10/2001	Male		Gold
10	Sherri	Dog	Siberian Huskie	9/13/2001	Female	AKC	Black/White
11	Susan	Dog	Dalmation	1/22/2001	Female	AKC	Spotted

Record: 3 of 191

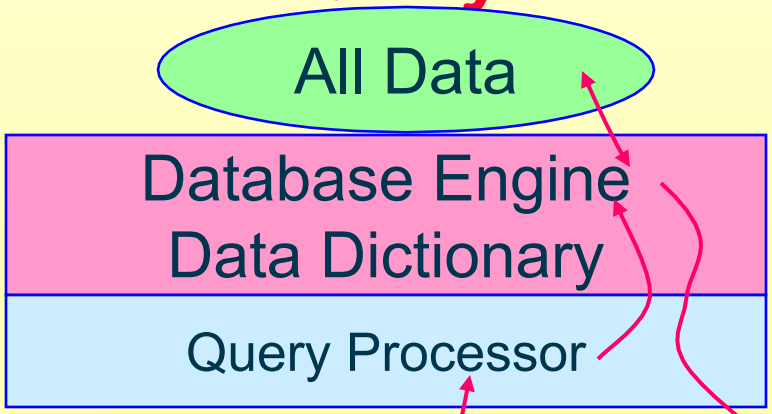
DATABASE

Database Tables (Oracle)



DATABASE

DBMS Query Processor



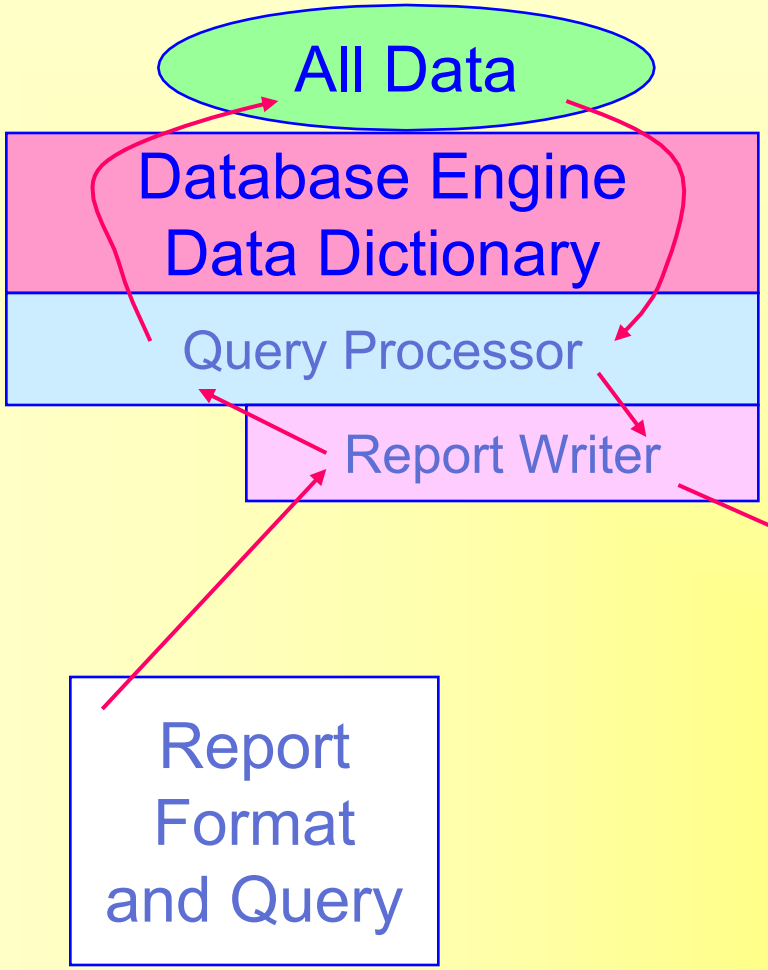
Animal
AnimalID
Name
Category
Breed

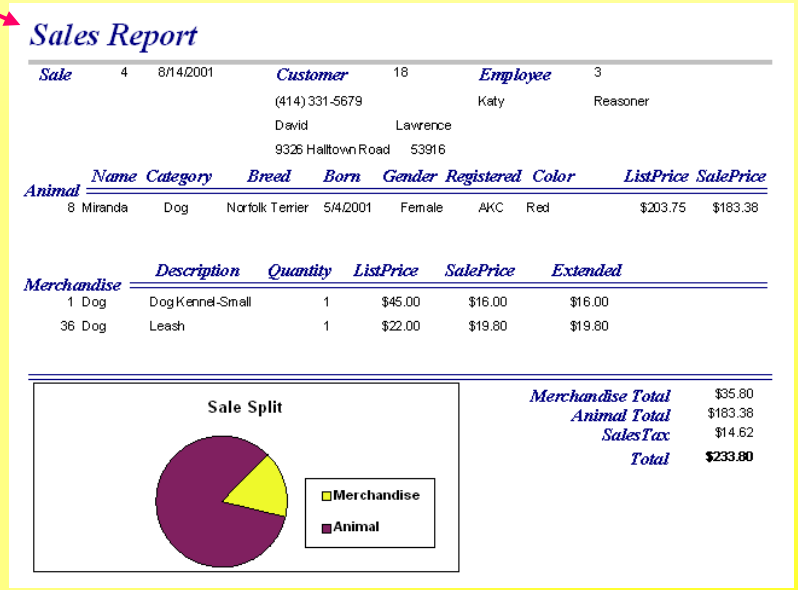
Field	Category	AnimalID
Table	Animal	Animal
Totals	Group By	Count
Sort		Descending
Criteria		
Or		

Category	CountOfAnimalID
Dog	100
Cat	47
Bird	15
Fish	14
Reptile	6
Mammal	6
Spider	3

DATABASE

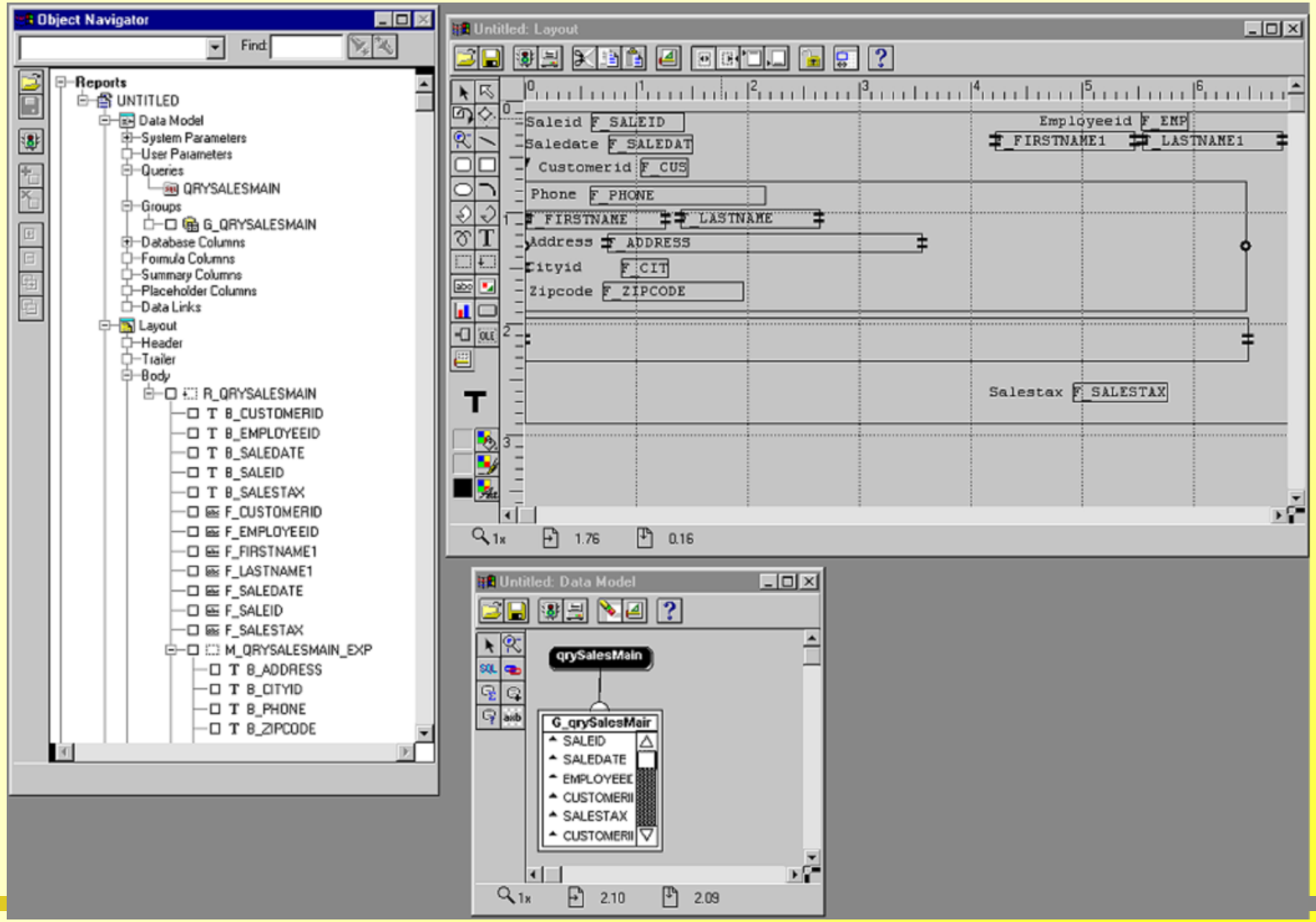
DBMS Report Writer





DATABASE

Report Writer (Oracle)



DATABASE

DBMS Input Forms

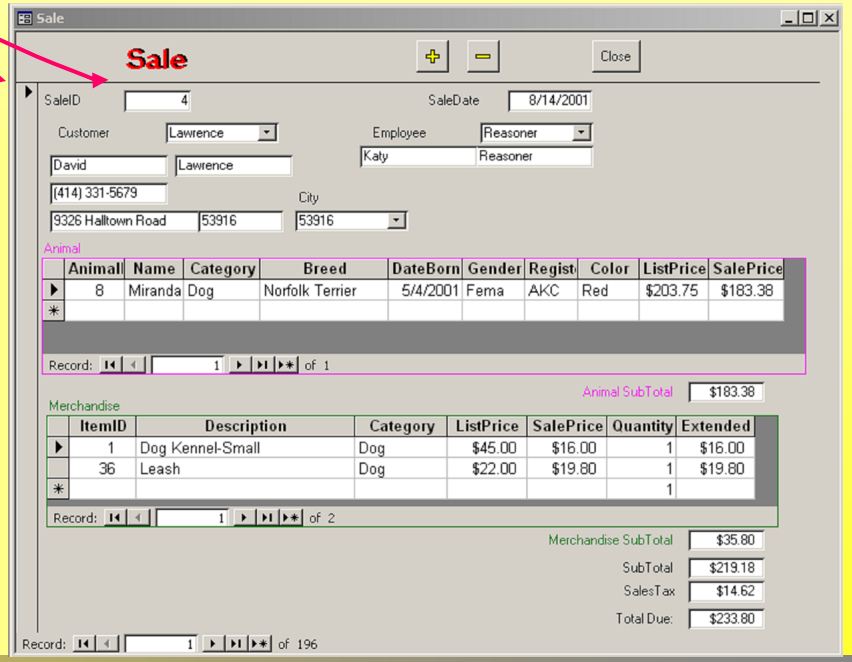
All Data

Database Engine
Data Dictionary

Query Processor

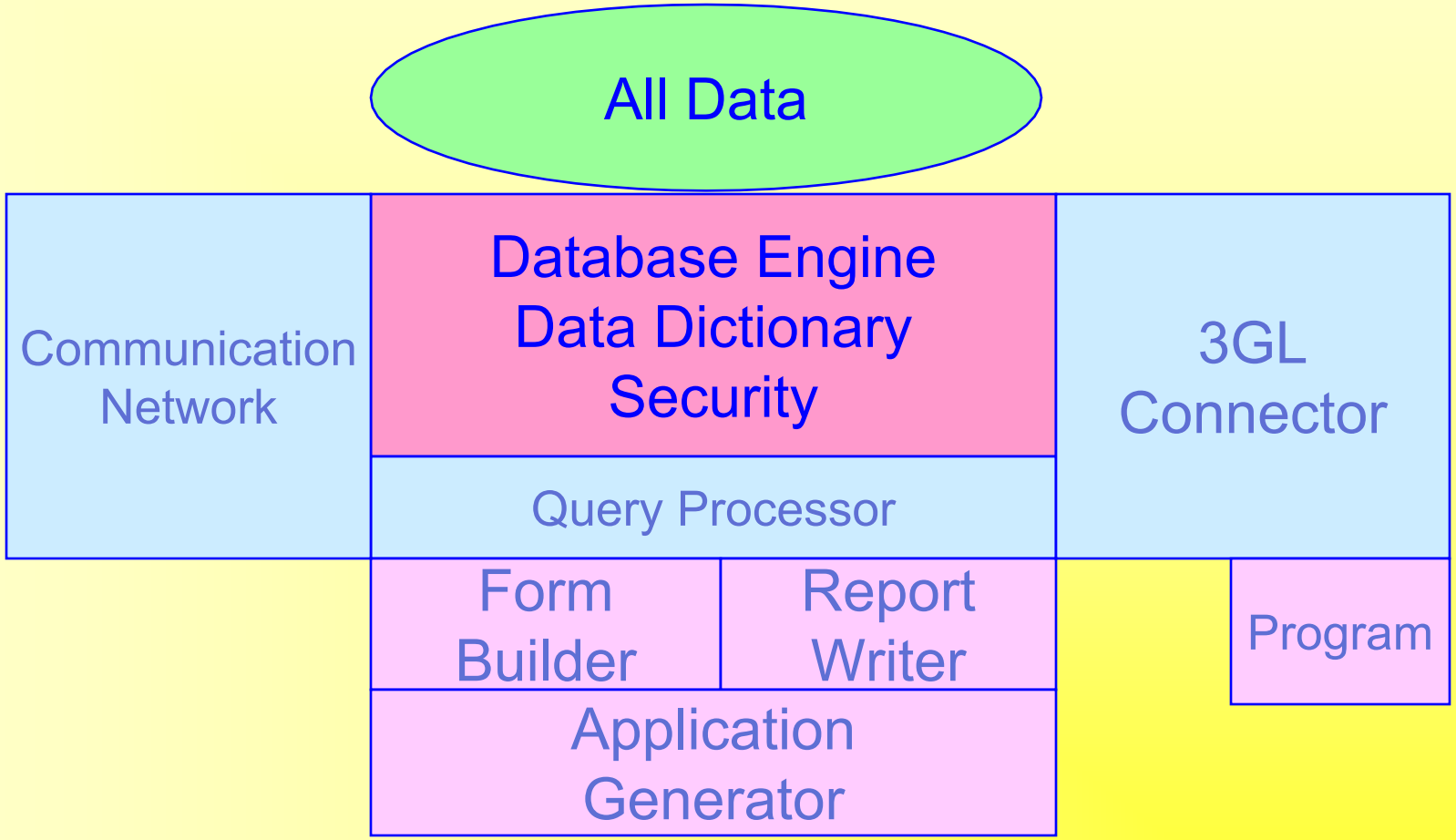
Form Builder

Input
Form
Design



DATABASE

DBMS Components



D A T A B A S E

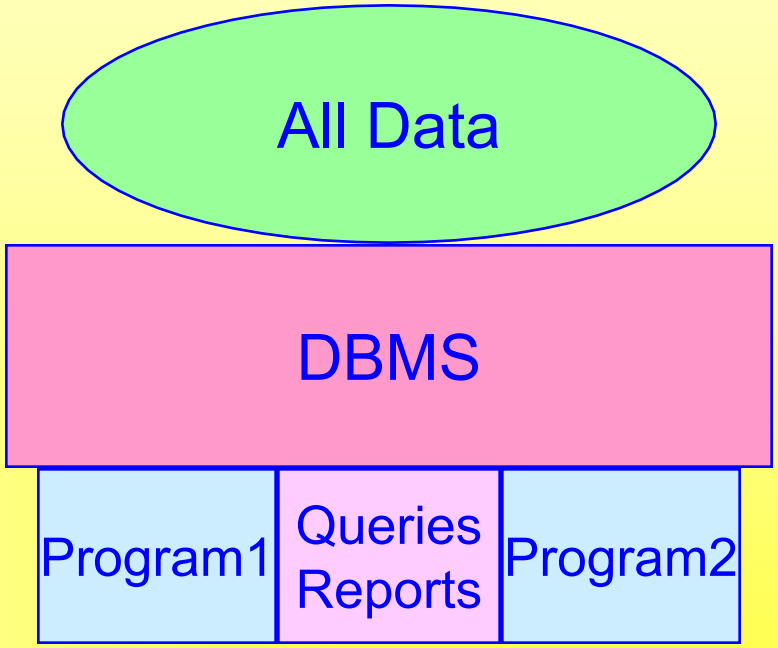
Advantages of Database Approach

- Minimal data redundancy.
- Data consistency.
- Integration of data.
- Sharing of data.
- Enforcement of standards.
- Ease of application development.
- Uniform security, privacy and integrity.
- Data independence.

DATABASE

Database Management Approach

- Data is most important
  - Data defined first
  - Standard format
- Access through DBMS
  - Queries, Reports, Forms
  - Application Programs
  - 3GL Interface
- Data independence
  - Change data definition without changing code
  - Alter code without changing data
  - Move/split data without changing code



DATABASE

Modifying Data with DBMS

- Add cell number to employee table
  - Open table definition
  - Add data element
  - If desired, modify reports
    - Use report writer
    - No programming
- Existing reports, queries, code will all run as before with no changes.

Field Name	Data Type	Description
EmployeeID	Number	Autonumber..
TaxpayerID	Text	Federal ID
LastName	Text	
FirstName	Text	
. . .		
Phone	Text	
. . .		
CellPhone	Text	Cellular . . .

DATABASE

Drawbacks of old File methods

- Uncontrolled Duplication
  - Wastes space
  - Hard to update all files
- Inconsistent data
- Inflexibility
  - Hard to change data
  - Hard to change programs
- Limited data sharing
- Poor enforcement of standards
- Poor programmer productivity
- Excessive program maintenance

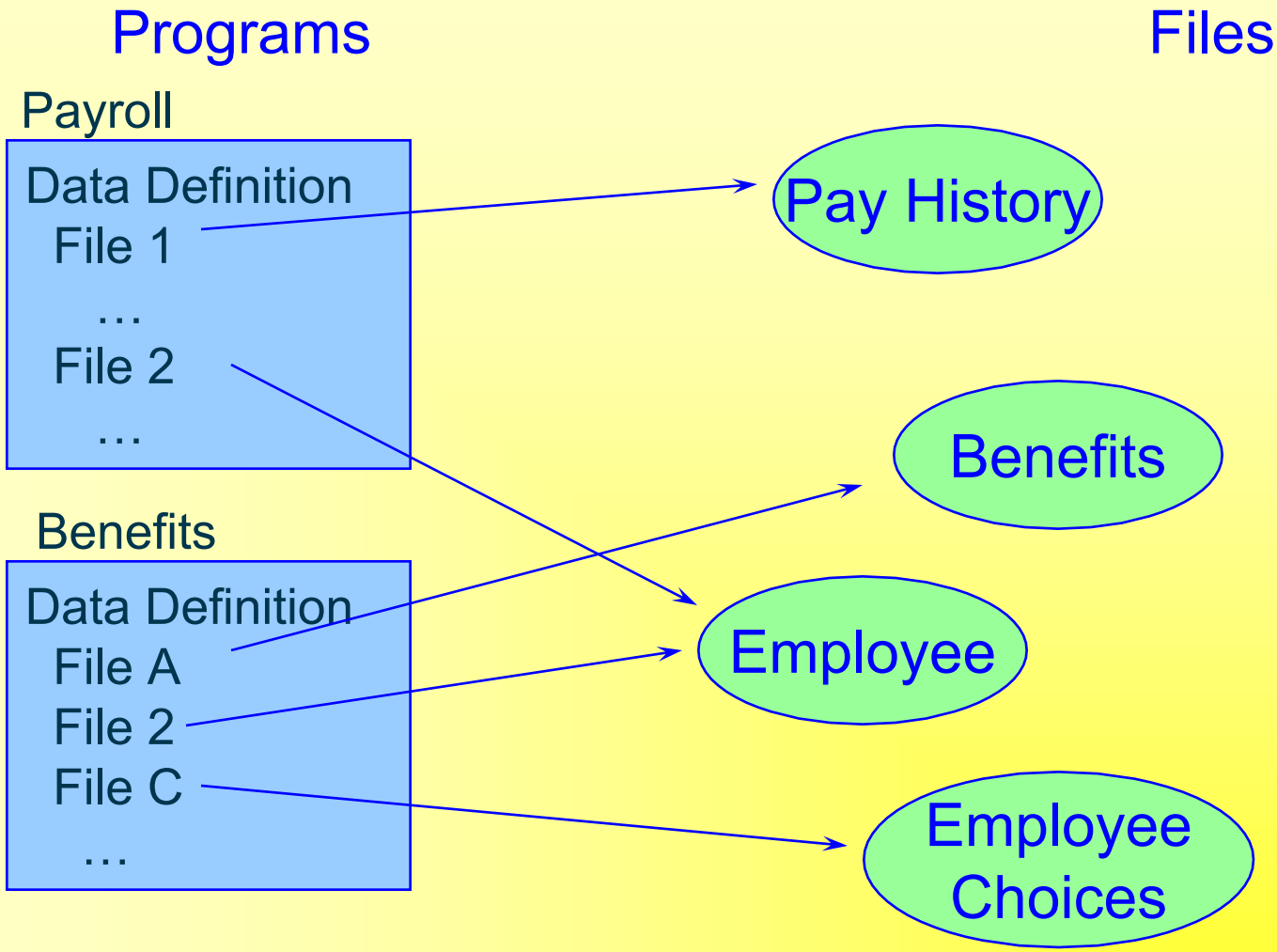
DATABASE

File Method Problems

- Files defined in program
  - Cannot read file without definition
  - Hard to find definition
  - Every time you alter file, you must rewrite code
  - Change in a program/file will crash other code
  - Cannot tell which programs use each file
- Multiuser problems
  - Concurrency
  - Security
    - Access
    - Backup & Restore
  - Efficiency
    - Indexes
    - Programmer talent
      - System
      - Application

DATABASE

Old File Method/3GL



DATABASE

Example of File Method v DBMS

COBOL

```
File Division
01 Employees
  02 ID
  02 Name
  02 Address
       02 Cell Phone
01 Department
  02 ID
  02 . . .
```

More programs

```
File Division
01 Employees
...
```

Employee File

```
112 Davy Jones 999 Elm
Street . . . 113 Peter Smith
101 Oak St . . .
```

- Add to file (e.g.Cell phone)
  - Write code to copy employee file and add empty cell phone slot.
  - Find all programs that use employee file.
    - Modify file definitions.
    - Modify reports (as needed)
    - Recompile, fix new bugs.
- Easier: Keep two employee files?

DATABASE

Examples of Commercial Systems

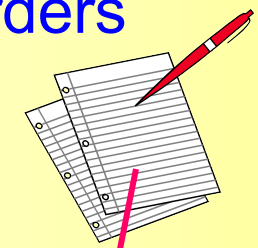
- Oracle
- Informix (Unix)
- DB2, SQL/DS (IBM)
- Access (Microsoft)
- SQL Server (Microsoft +)
- Many older (Focus, IMS, ...)
- mySQL
- ProgresSQL

DATABASE

Hierarchical Database

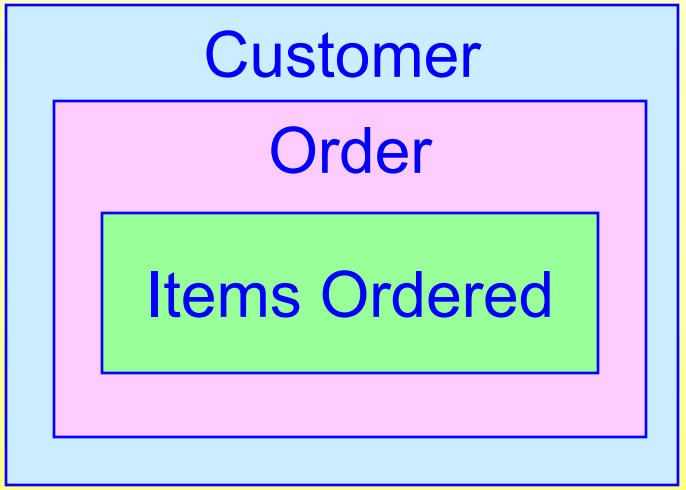
Customers

Orders



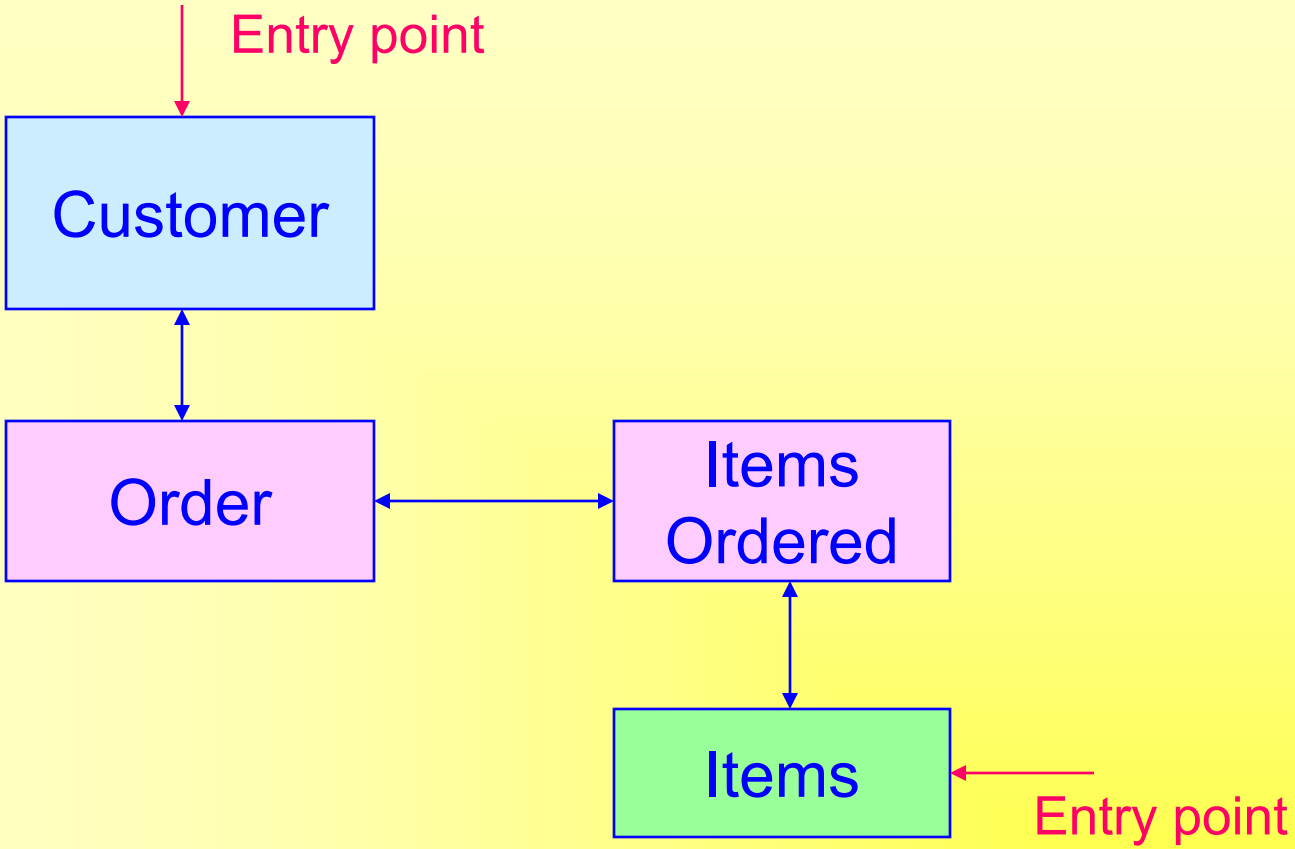
Items

Item	Description	Quantity
998	Dog Food	12
764	Cat Food	11



To retrieve data, you must start at the top (customer). When you retrieve a customer, you retrieve all nested data.

Network Database



DATABASE

Relational Database

Customer(CustomerID, Name, …

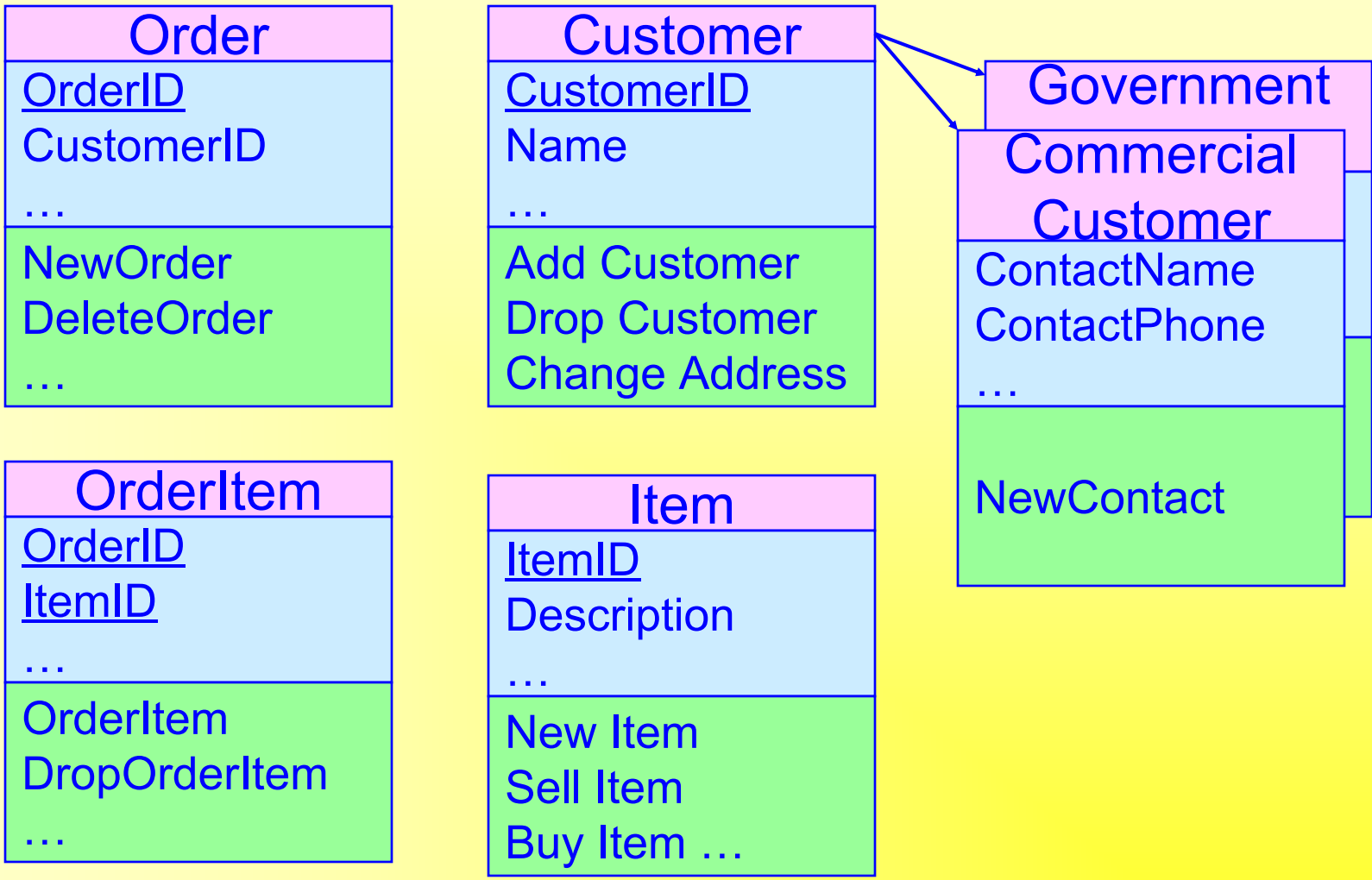
Order(OrderID, CustomerID, OrderDate, …

ItemsOrdered(OrderID, ItemID, Quantity, …

Items(ItemID, Description, Price, …

DATABASE

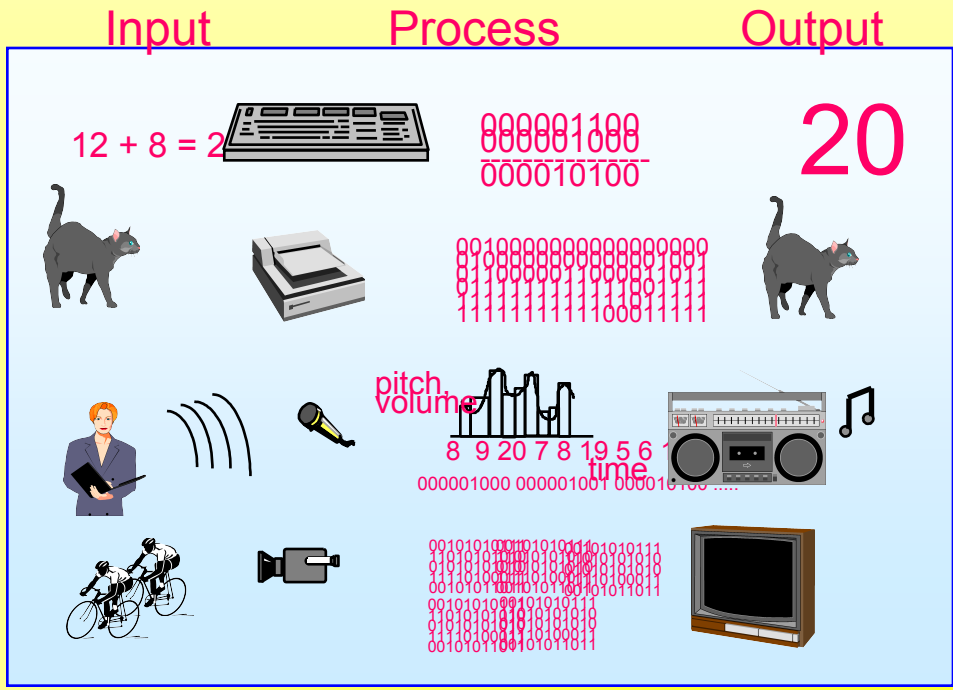
Object-Oriented DBMS



DATABASE

Base Data Types

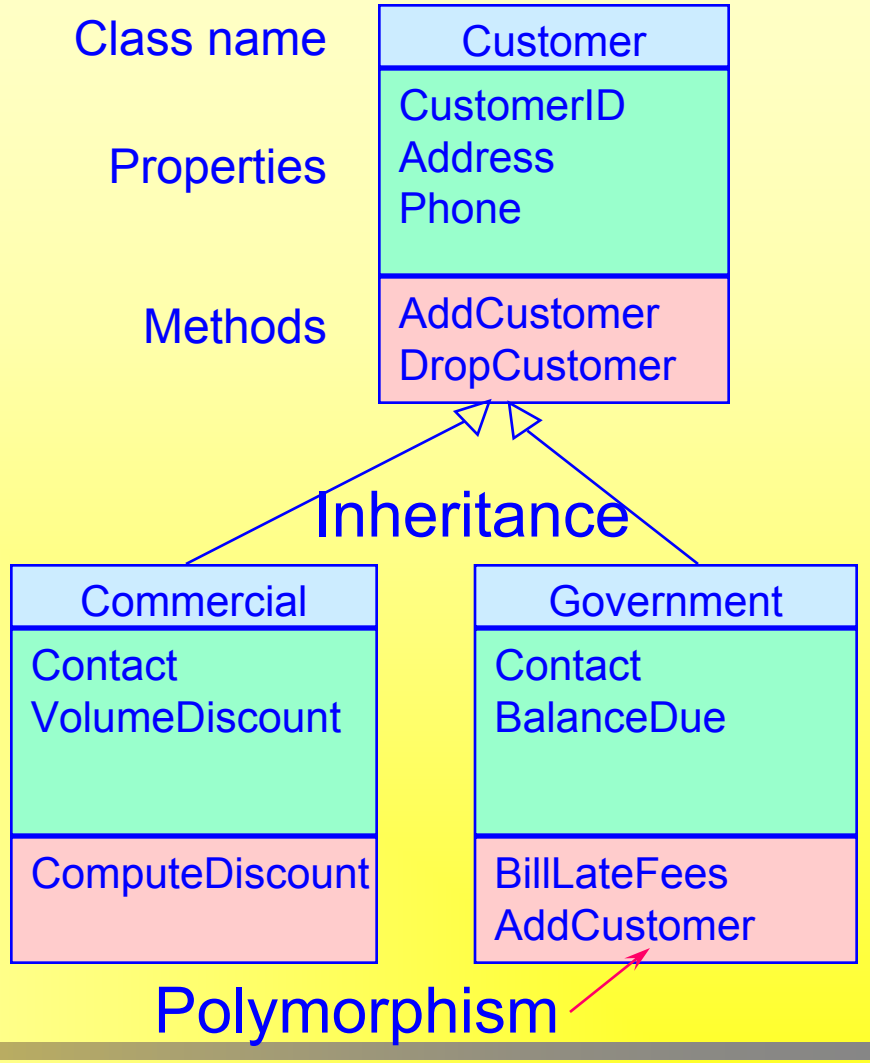
- Numbers
  - Integers
  - Reals
- Text
  - Length
  - International
- Date/Time
- Images
  - Bitmap
  - Vector
- Sound
  - Samples
  - MIDI
- Video



DATABASE

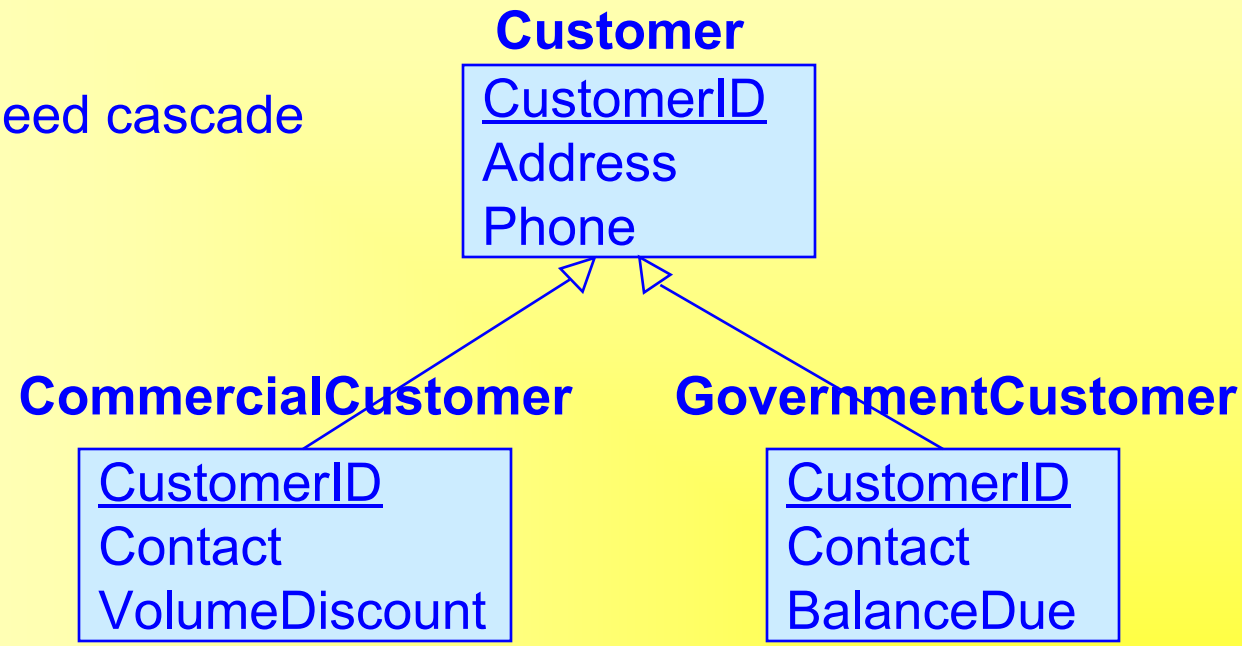
Objects

- Object Definition--encapsulation.
  - Object Name
  - Properties
  - Methods
- Most existing DBMS do **not** handle inheritance.
  - Combine into one table.
  - Use multiple tables and link by primary key.
    - More efficient.
    - Need to add rows to many tables.



Objects in a Relational Database

- Separate inherited classes.
- Link by primary key.
- Adding a new customer requires new rows in each table.
- Definitely need cascade delete.

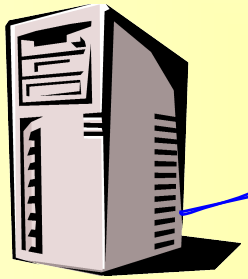


OO Difficulties: Methods

IBM Server

Database Object

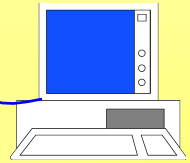
Unix Server



Database Object

Customer
Method:
Add New Customer
Program code

Personal Computer



Application

Customer
Name
Address
Phone

How can a method run on different computers?

Different processors use different code.

Possibility: Java

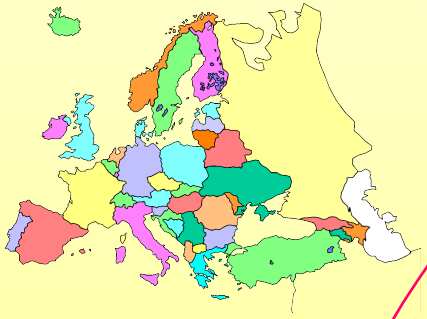
SQL 99: OO Features

- Abstract data type
  - User defined data types.
  - Equality and ordering functions.
  - Encapsulation: Public, Private, Protected.
  - Inheritance.
- Sub-tables that inherit all columns from another table.
- Persistent Stored Modules (Programming Language).
  - Create methods.
  - SQL and extensions.
  - External language.
- User defined operators.
- Triggers for events.
- External language support
  - Call-Level Interface (CLI)
    - Direct access to DBMS
  - Embedded SQL
    - SQL commands in an external language.

DATABASE

Abstract Data Types

```
Procedure: DrawRegion
{
   Find region components.
      SQL: Select …
   For each component {
      Fetch MapLine
      Set line attributes
      MapLine.Draw
   }
}
```



GeoPoint
Latitude
Longitude
Altitude

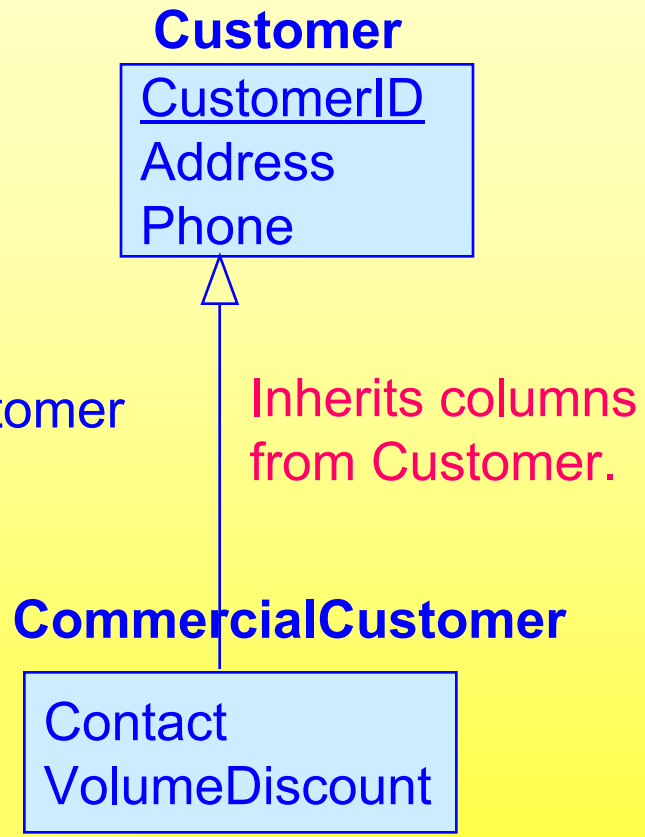
GeoLine
NumberOfPoints
ListOfGeoPoints

RegionID	Name	Size	Superset	MapLine	…
12	Europe	…	World		
394	Spain	…	Europe		
222	France	…	Europe		

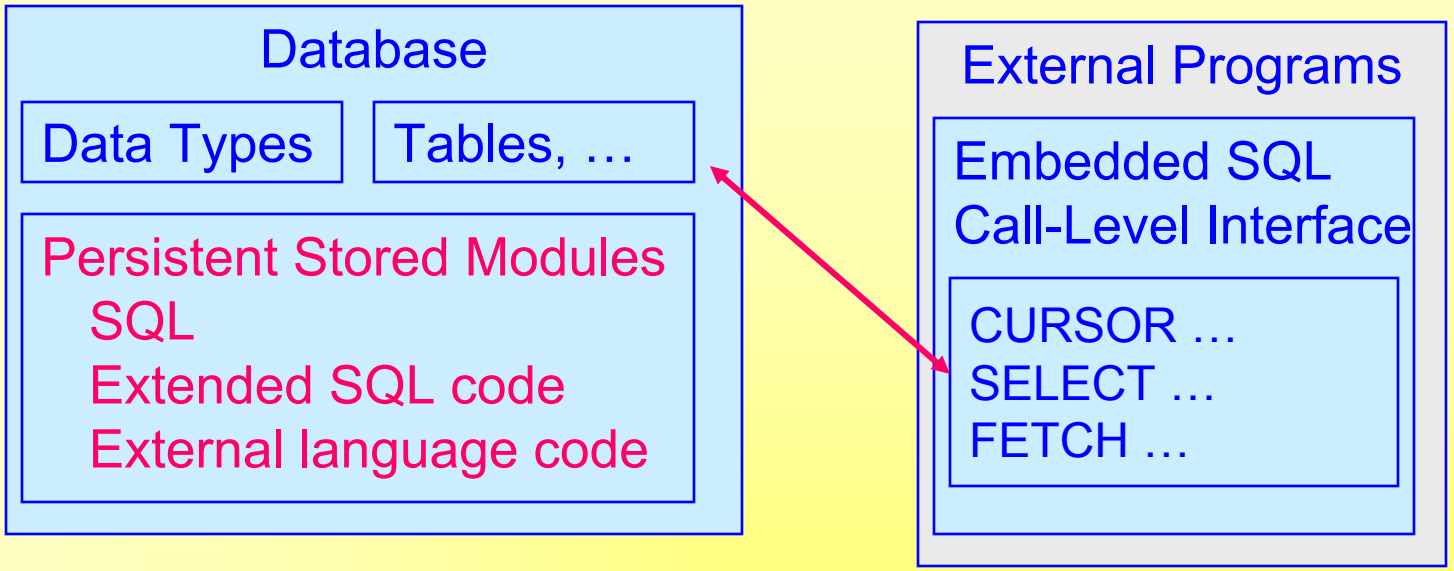
SQL 99 Sub-Tables

```
CREATE SET TABLE Customer
(
   CustomerID        INTEGER,
   Address           VARCHAR,
   Phone             CHAR(15)
)

CREATE SET TABLE CommercialCustomer
(
   Contact           VARCHAR,
   VolumeDiscount    NUMERIC(5,2)
)
   UNDER Customer;
```



SQL 99: Programming



OODBMS Vendors

GemStone Systems, Inc.
Hewlett-Packard, Inc. (OpenODB)
IBEX Corporation, SA.
Illustra (Informix, Inc.)
Matisse Software, Inc.
O2 Technology, Inc.
Objectivity, Inc.
Object Design, Inc.
ONTOS, Inc.
POET Software Corporation
UniSQL
Unisys Corporation (OSMOS)
Versant Object Technology

DATABASE

Why don't all developers use a DBMS?

- Most new projects (in last 5 years) do use a DBMS
- Need specialized personnel
  - Programmers
  - Designers/Analysts
  - Database administrators
- Need to define data for organization
- Cost
  - PC: $400 - $2000
  - Large: $100,000 +

DATABASE

How do you sell a DBMS approach?

- Applications change a lot, but same data.
- Need for *ad hoc* questions and queries.
- Need to reduce development times.
- Need shared data.
- Improve quality of data.
- Enable users to do more development.

DATABASE

Building the Right System: Feasibility

- Costs
  - Up-front/one-time
    - Software ($ millions !)
    - Hardware
    - Communications
    - Data conversion
    - Studies and Design
    - Training
  - On-going costs
    - Personnel
    - Software upgrades
    - Supplies
    - Support
    - Software & Hardware maintenance

Easy to estimate

- Benefits
  - Cost Savings
    - Software maintenance
    - Fewer errors
    - Less data maintenance
    - Less user training
  - Increased Value
    - Better access to data
    - Better decisions
    - Better communication
    - More timely reports
    - Faster reaction to change
    - New products & services
  - Strategic Advantages
    - Lock out competitors

Hard to value

DATABASE

Economic Feasibility: NPV

Year	Benefits	Costs	Net
0	0	50000	-50000
1	18000	5000	13000
2	18000	5000	13000
3	18000	5000	13000
4	18000	5000	13000
5	18000	5000	13000
		Discount Rate	
	0.05	0.07	0.10
NPV	$6,283.20	$3,302.57	($719.77)

=NPV(B14,$D$7:$D$11)+$D$6

=NPV(rate, range) + starting

DATABASE

Exercise: Build a First Database

Employee(EmployeeID, LastName, FirstName, Address, DateHired)

332	Ant	Adam	354 Elm	5/5/1964
442	Bono	Sonny	765 Pine	8/8/1972
553	Cass	Mama	886 Oak	2/2/1985
673	Donovan	Michael	421 Willow	3/3/1971
773	Moon	Keith	554 Cherry	4/4/1972
847	Morrison	Jim	676 Sandalwood	5/5/1968

Client(ClientID, LastName, FirstName, Balance, EmployeeID)

1101	Jones	Joe	113.42	442
2203	Smith	Mary	993.55	673
2256	Brown	Laura	225.44	332
4456	Dieter	Jackie	664.90	442
5543	Wodkoski	John	984.00	847
6673	Sanchez	Paula	194.87	773
7353	Chen	Charles	487.34	332
7775	Hagen	Fritz	595.55	673
8890	Hauer	Marianne	627.39	773
9662	Nguyen	Suzie	433.88	553
9983	Martin	Mark	983.31	847

DATABASE

Exercise: Report

Ant, Adam	5/5/1964	
	Brown, Laura	225.24
	Chen, Charles	487.34
		712.58
Bono, Sonny	8/8/1972	
	Dieter, Jackie	664.90
	Jones, Joe	114.32
		779.22

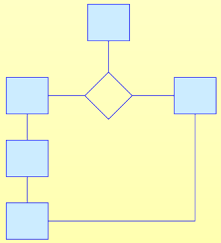
Database Management Systems

Chapter 2
Database Design

Jerry Post

DATABASE

Database System Design



Customer(CustomerID, Name, Address, ...)
SalesPerson(EmployeeID, Name, Commission, ...)
Order(OrderID, OrderDate, CustomerID, EmployeeID, ...)
OrderItem(OrderID, ItemID, Quantity, Price, ...)
Item(ItemID, Description, ListPrice, ...)

User views of data.

Conceptual data model.

Implementation (relational) data model.

Physical data storage.

Class diagram that shows business entities, relationships, and rules.

List of nicely-behaved tables. Use data normalization to derive the list.

Indexes and storage methods to improve performance.

DATABASE

The Need for Design

- Goal: To produce an information system that adds value for the user
  - Reduce costs
  - Increase sales/revenue
  - Provide competitive advantage
- Objective: To understand the system
  - To improve it
  - To communicate with users and IT staff
- Methodology: Build models of the system

DATABASE

Designing Systems

- Designs are a model of existing & proposed systems
  - They provide a picture or representation of *reality*
  - They are a simplification
  - Someone should be able to read your design (model) and describe the features of the actual system.
- You build models by talking with the users
  - Identify processes
  - Identify objects
  - Determine current problems and future needs
  - Collect user documents (views)
- Break complex systems into pieces and levels

DATABASE

Design Stages

- Initiation
  - Scope
  - Feasibility
  - Cost & Time estimates
- Requirements Analysis
  - User Views & Needs
    - Forms
    - Reports
  - Processes & Events
  - Objects & Attributes
- Conceptual Design
  - Models
    - Data flow diagram
    - Entity Relationships
    - Objects
  - User feedback
- Physical Design
  - Table definitions
  - Application development
    - Queries
    - Forms
    - Reports
    - Application integration
  - Data storage
  - Security
  - Procedures
- Implementation
  - Training
  - Purchases
  - Data conversion
  - Installation
- Evaluation & Review

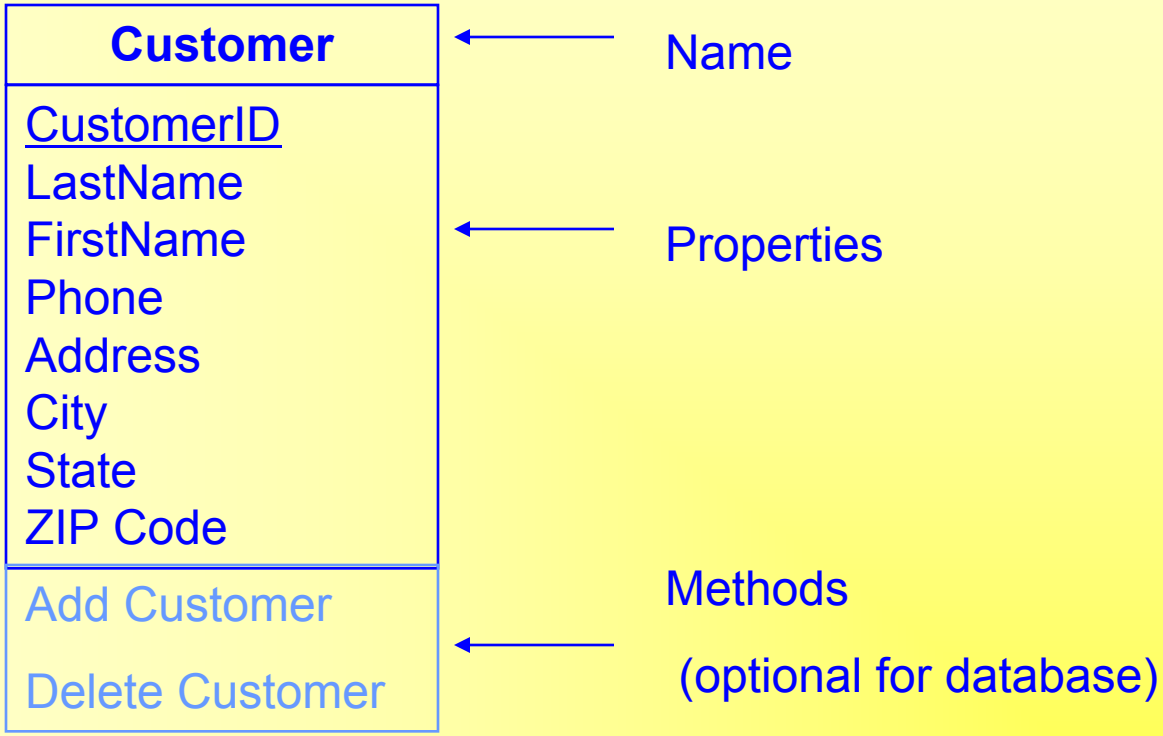
DATABASE

Initial Steps of Design

1. Identify the exact goals of the system.
2. Talk with the users to identify the basic forms and reports.
3. Identify the data items to be stored.
4. Design the classes (tables) and relationships.
5. Identify any business constraints.
6. Verify the design matches the business rules.

DATABASE

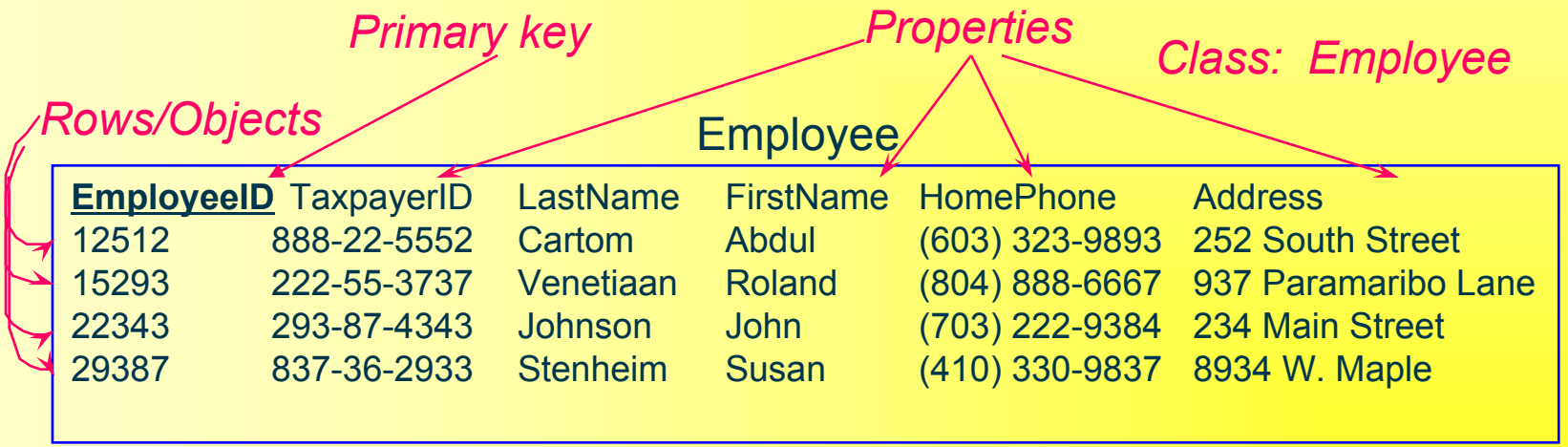
Entities/Classes



DATABASE

Definitions

- Relational database: A collection of tables.
- Table: A collection of columns (attributes) describing an entity. Individual objects are stored as rows of data in the table.
- Property (attribute): a characteristic or descriptor of a class or entity.
- Every table has a primary key.
  - The smallest set of columns that uniquely identifies any row
  - Primary keys can span more than one column (concatenated keys)
  - We often create a primary key to insure uniqueness (e.g., CustomerID, Product#, . . .) called a surrogate key.



DATABASE

Definitions

Entity:	Something in the *real world* that we wish to describe or track.
Class:	Description of an entity, that includes its attributes (properties) and behavior (methods).
Object:	One instance of a class with specific data.
Property:	A characteristic or descriptor of a class or entity.
Method:	A function that is performed by the class.
Association:	A relationship between two or more classes.

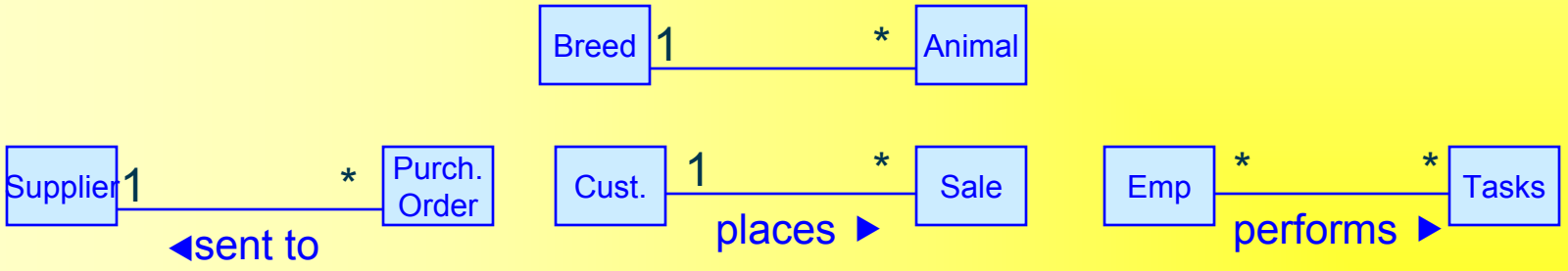
Pet Store Examples

Entity:	Customer, Merchandise, Sales
Class:	Customer, Merchandise, Sale
Object:	Joe Jones, Premium Cat Food, Sale #32
Property:	LastName, Description, SaleDate
Method:	AddCustomer, UpdateInventory, ComputeTotal
Association:	Each Sale can have only one Customer.

Associations

- General
  - One-to-one (1:1)
  - One-to-many (1:M)
  - Many-to-many (M:N)
- Relationships represent business rules
  - Sometimes *common-sense*
  - Sometimes unique to an organization
- Users often know current relationships, rarely future

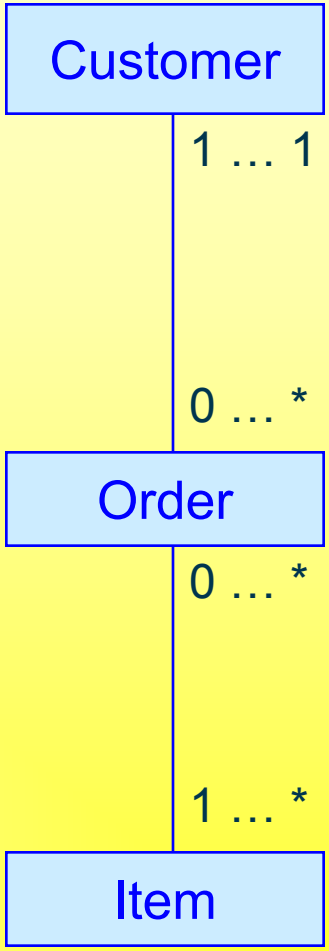
- Objects related to objects
  - An employee can work in only one department
  - Many departments can work on many different products
- Objects related to properties
  - An employee can have only one name
  - Many employees can have the same last name



DATABASE

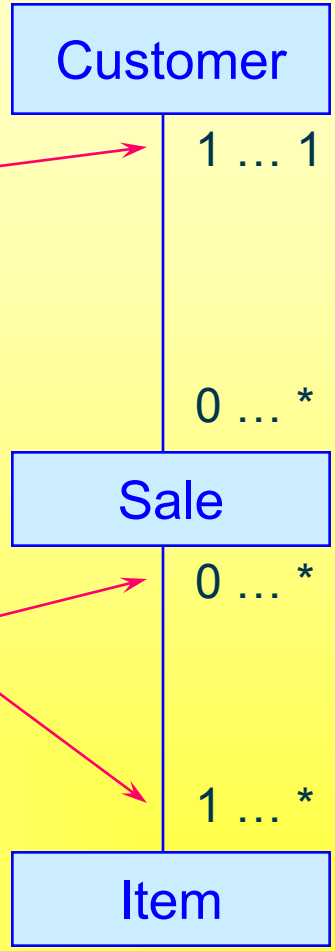
Class Diagram

- Class/Entity (box)
- Association/Relationship
  - Lines
  - Minimum
    - 0: optional
    - 1: required
  - Maximum
    - Arrows
    - 1, M

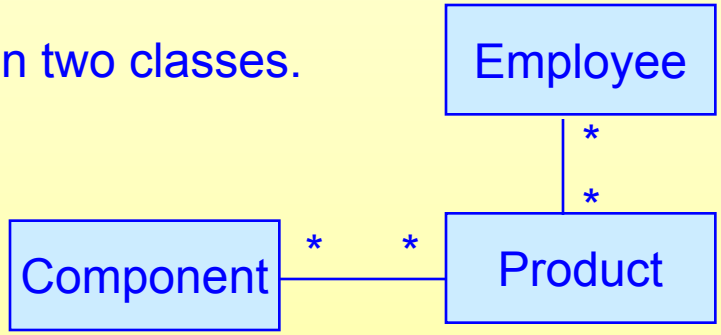


Sample Association Rules (Multiplicity)

- An order must have exactly 1 customer,
  - 1 … 1 Minimum of 1
  - 1 … 1 Maximum of 1
- And at least one item.
  - 1 … * Minimum of 1
  - 1 … * Maximum many
- An item can show up on no orders or many orders.
  - 0 … * Optional (0)
  - 0 … * Maximum many

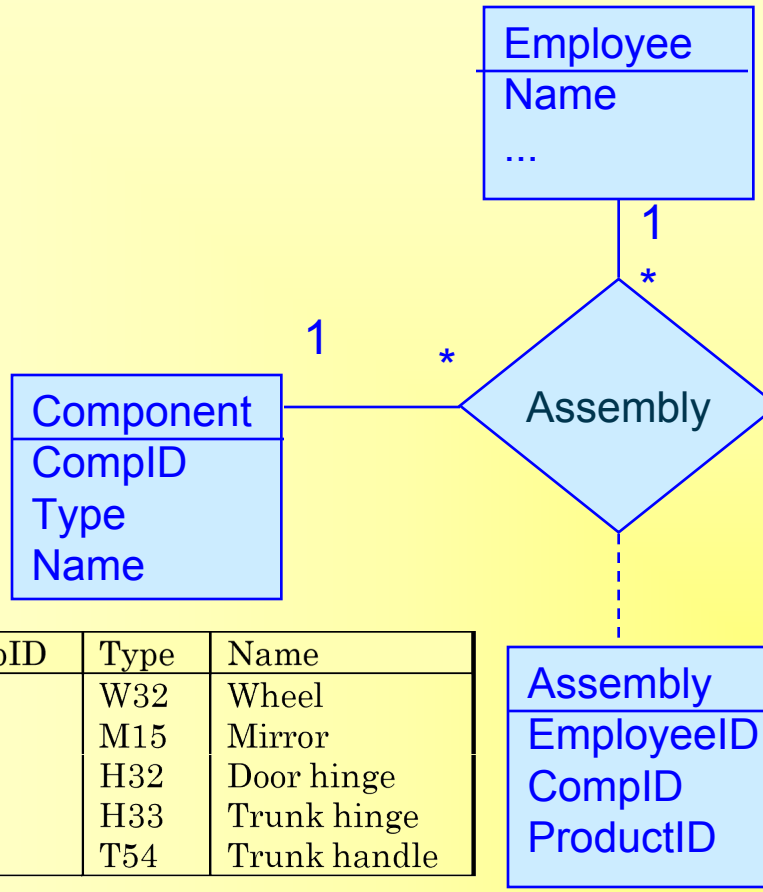


N-ary Associations



- Associations can connect more than two classes.
- Associations can become classes.
  - Events
  - Many-to-many
  - Need to keep data
- Example has two many-to-many relationships.
  - We know which components go into each product.
  - We know which employees worked on a product.
- We need to expand the relationships to show which employees installed which components into each product.
  - Each assembly entry lists one employee, one component, and one product.
  - By appearing on many assembly rows, the many-to-many relationships can still exist.

N-ary Association Example



EmployeeID	Name	...
11	Joe Jones	...
12	Maria Rio	...

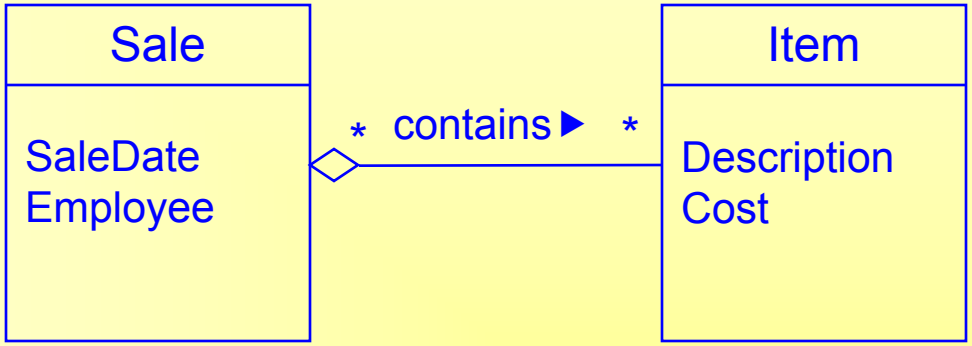
ProductID	Type	Name
A3222	X32	Corvette
A5411	B17	Camaro

CompID	Type	Name
563	W32	Wheel
872	M15	Mirror
882	H32	Door hinge
883	H33	Trunk hinge
888	T54	Trunk handle

EmployeeID	CompId	ProductID
11	563	A3222
11	872	A3222
11	563	A5411
11	872	A5411
12	563	A3222
12	882	A3222
12	888	A3222
12	883	A5411

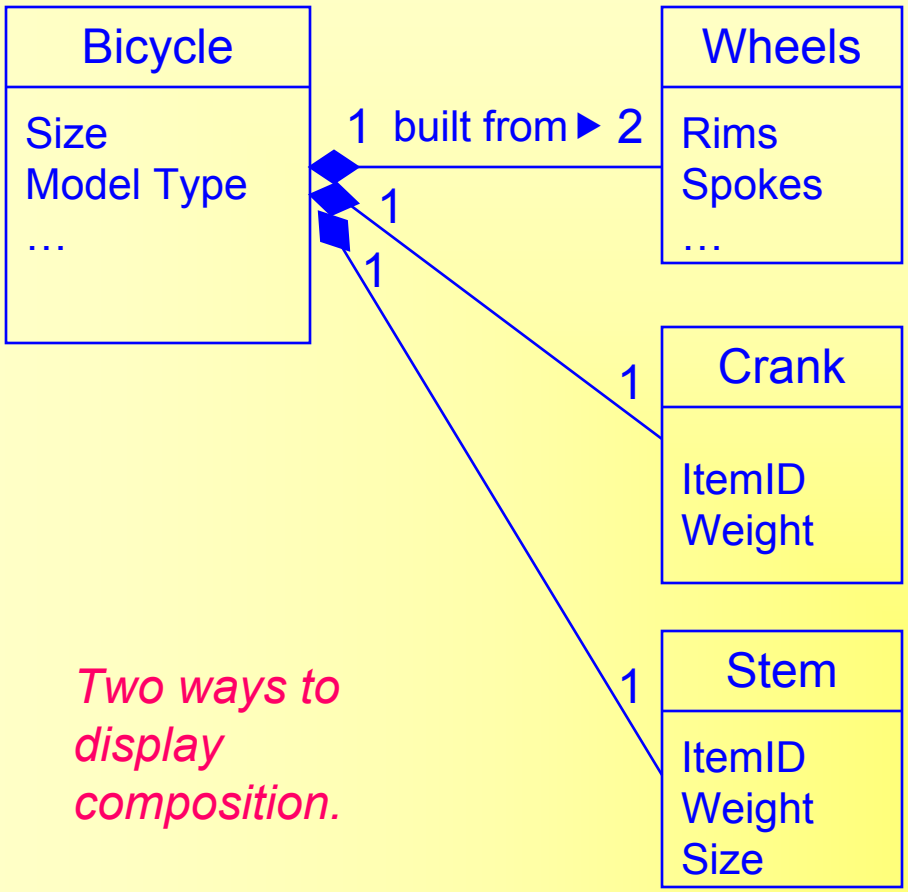
Multiplicity is defined as the number of items that could appear if the other N-1 objects are fixed. Almost always “many.”

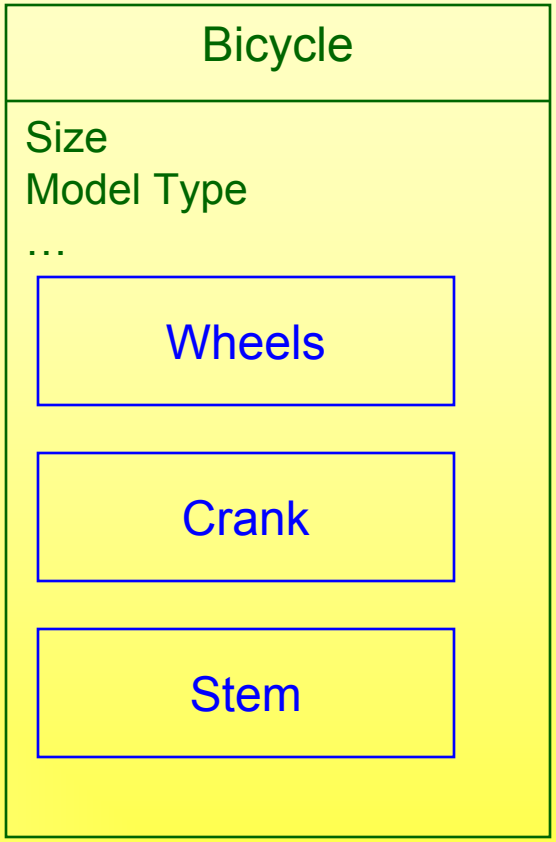
Association Details: Aggregation



Aggregation: the Sale consists of a set of Items being sold.

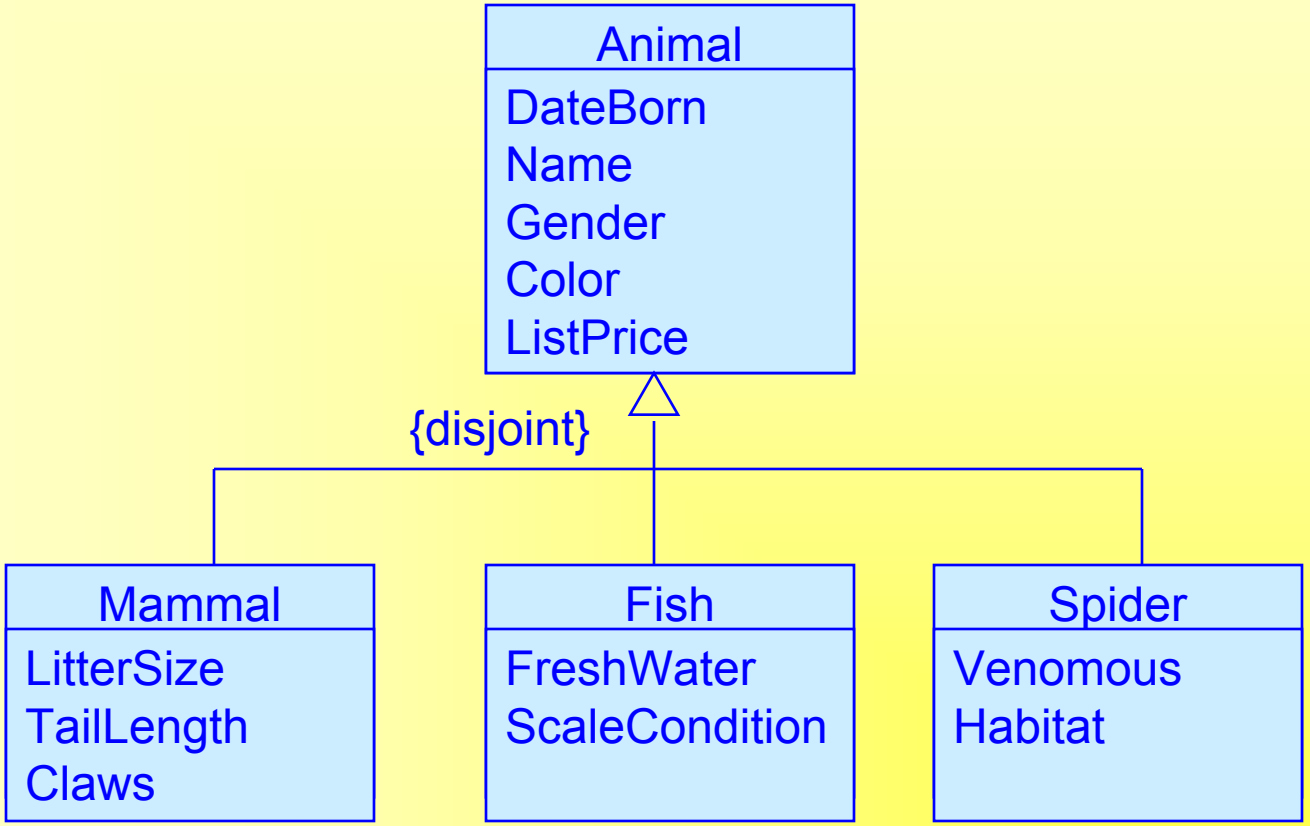
Association Details: Composition





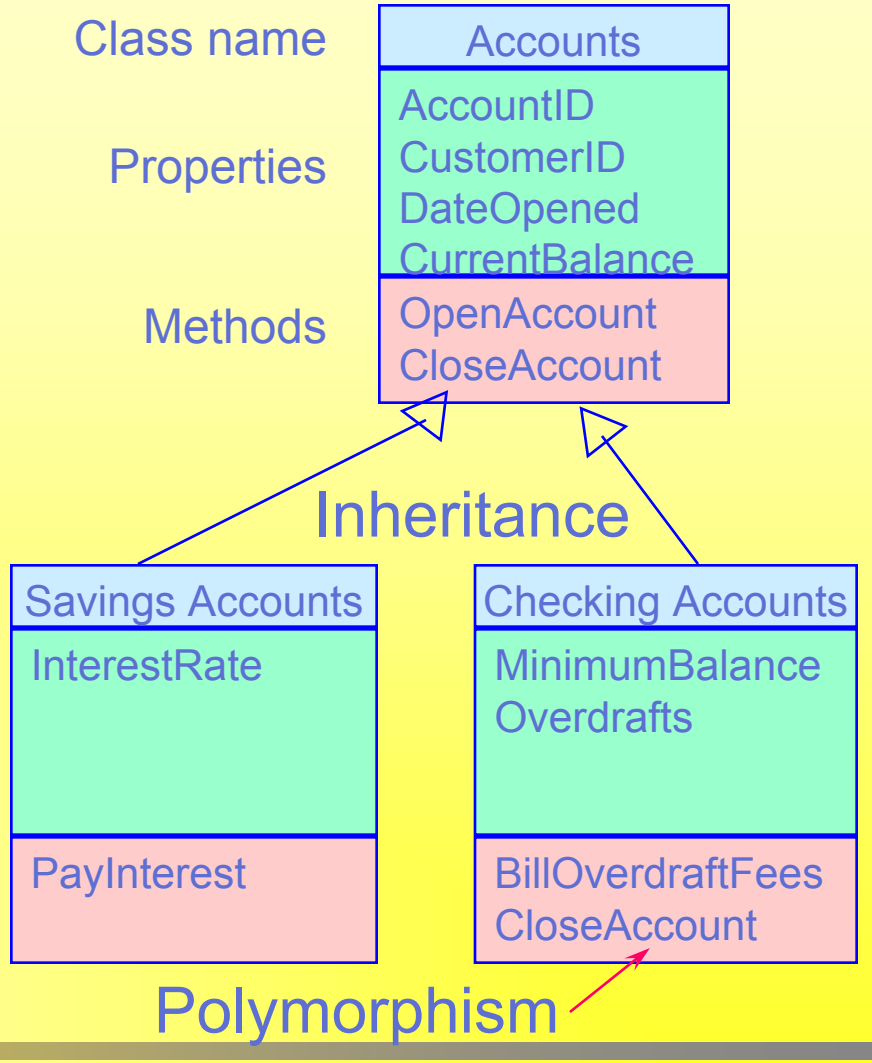
Composition: aggregation where the components become the new object.

Association Details: Generalization

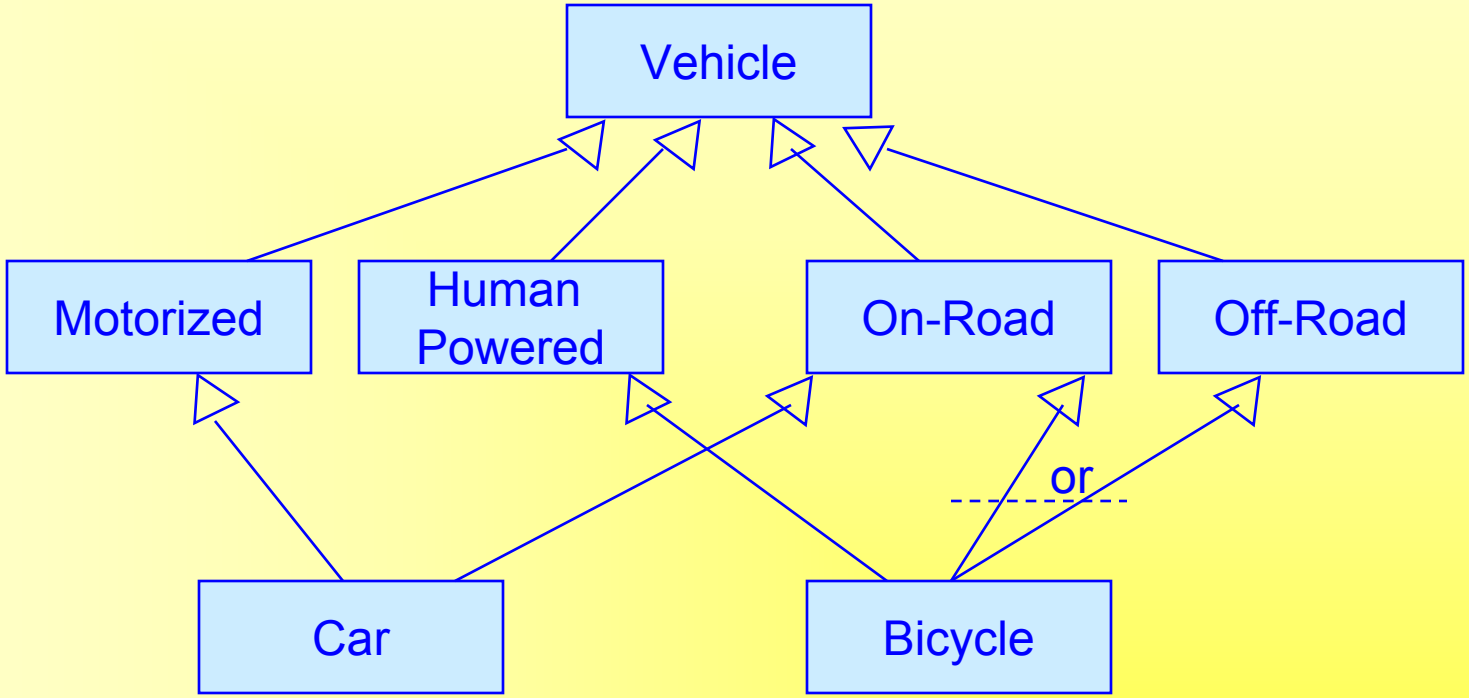


Inheritance

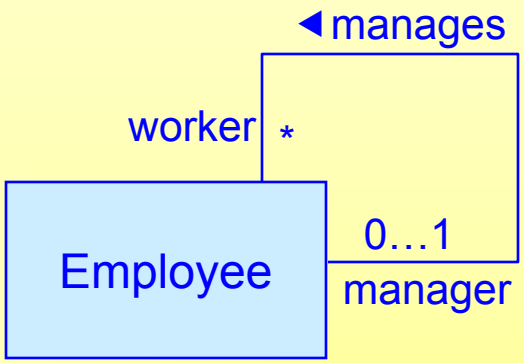
- Class Definition--encapsulation
  - Class Name
  - Properties
  - Methods
- Inheritance Relationships
  - Generic classes
  - Focus on differences
  - Polymorphism
  - Most existing DBMS do **not** handle inheritance



Multiple Parents



Association Details: Reflexive Relationship

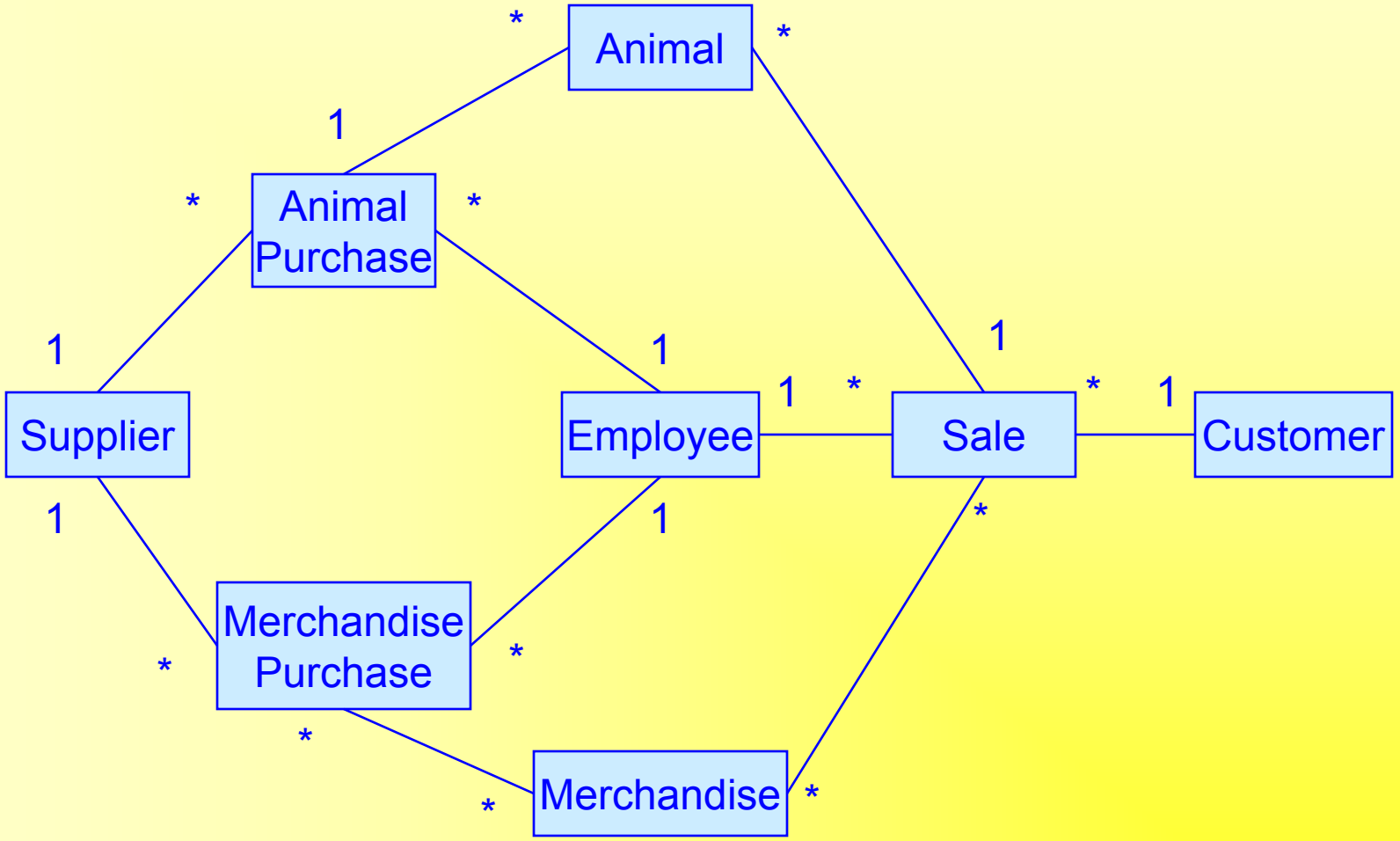


A reflexive relationship is an association from one class back to itself.

In this example, an employee can also be a manager of other employees.

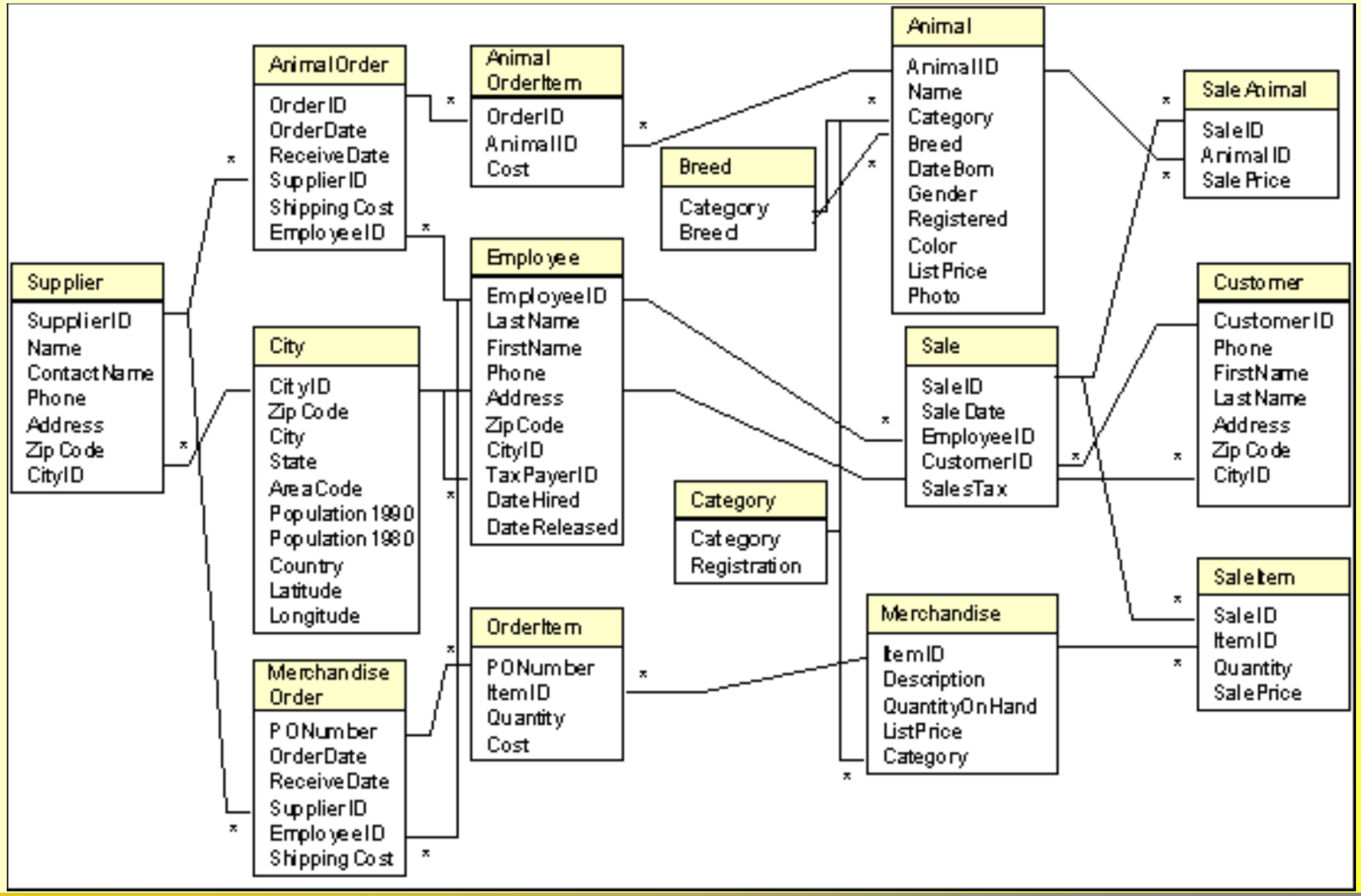
DATABASE

PetStore Overview Class Diagram



DATABASE

Pet Store Class Diagram: Access



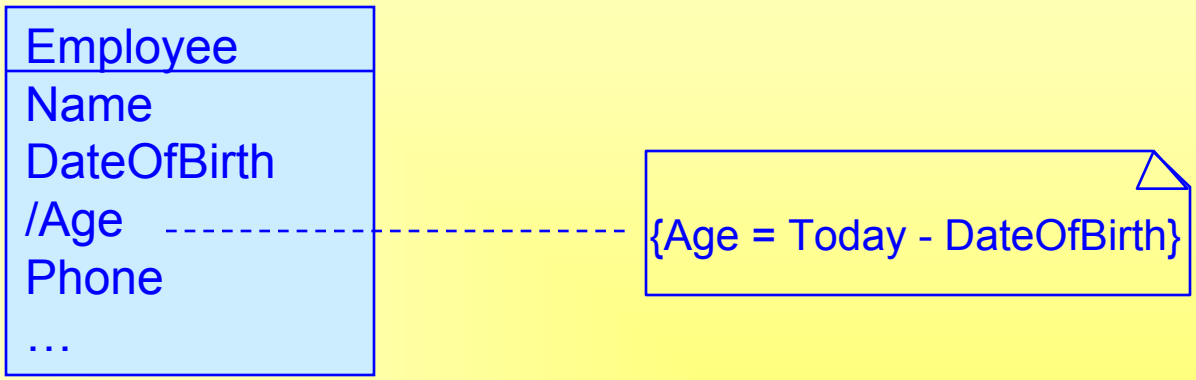
DATABASE

Data Type Sizes

	Access	SQL Server	Oracle
Text fixed variable Unicode memo	Text Memo	char, varchar nchar, nvarchar text	CHAR VARCHAR2 NVARCHAR2 LONG
Number Byte (8 bits) Integer (16 bits) Long (32 bits) (64 bits) Fixed precision Float Double Currency Yes/No	Byte Integer Long NA NA Float Double Currency Yes/No	tinyint smallint int bigint decimal(p,s) real float money bit	INTEGER INTEGER INTEGER NUMBER(38,0) NUMBER(p,s) NUMBER, FLOAT NUMBER NUMBER(38,4) INTEGER
Date/Time Interval	Date/Time NA	datetime smalldatetime interval year …	DATE INTERVAL YEAR …
Image	OLE Object	image	LONG RAW, BLOB
AutoNumber	AutoNumber	Identity rowguidcol	SEQUENCES ROWID

Computed Attributes

Denote computed values with a preceding slash (/).



DATABASE

Event Examples

- Business Event
  - Item is sold.
  - Decrease Inventory count.
- Data Event
  - Inventory drops below preset level.
  - Order more inventory.
- User Event
  - User clicks on icon.
  - Send purchase order to supplier.

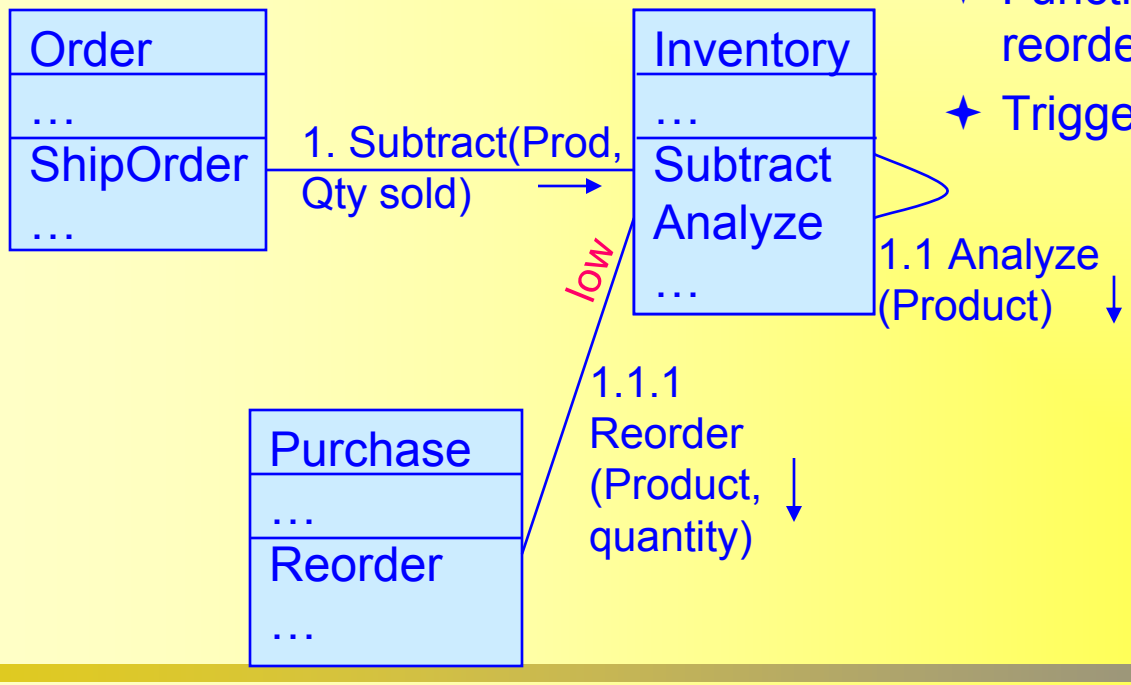
Trigger

ON (QuantityOnHand < 100)

THEN Notify Purchasing Manager

Event Triggers

- Business Process: Ship Product
  - Trigger: Inventory Change
  - Executes function/trigger in Inventory object.
- Object: Inventory
  - Property: Current Inventory.
  - Function: Update Inventory.
  - Trigger: On Update, call Analyze function.
- Process: Analyze Inventory
  - Function: Determine need to reorder.
  - Trigger: Generate new order.



DATABASE

Design Importance: Large Projects

- Design is harder on large projects.
  - Communication with multiple users.
  - Communication between IT workers.
  - Need to divide project into pieces for teams.
  - Finding data/components.
  - Staff turnover--retraining.
- Need to monitor design process.
  - Scheduling.
  - Evaluation.
- Build systems that can be modified later.
  - Documentation.
  - Communication/underlying assumptions and model.

DATABASE

Large Projects

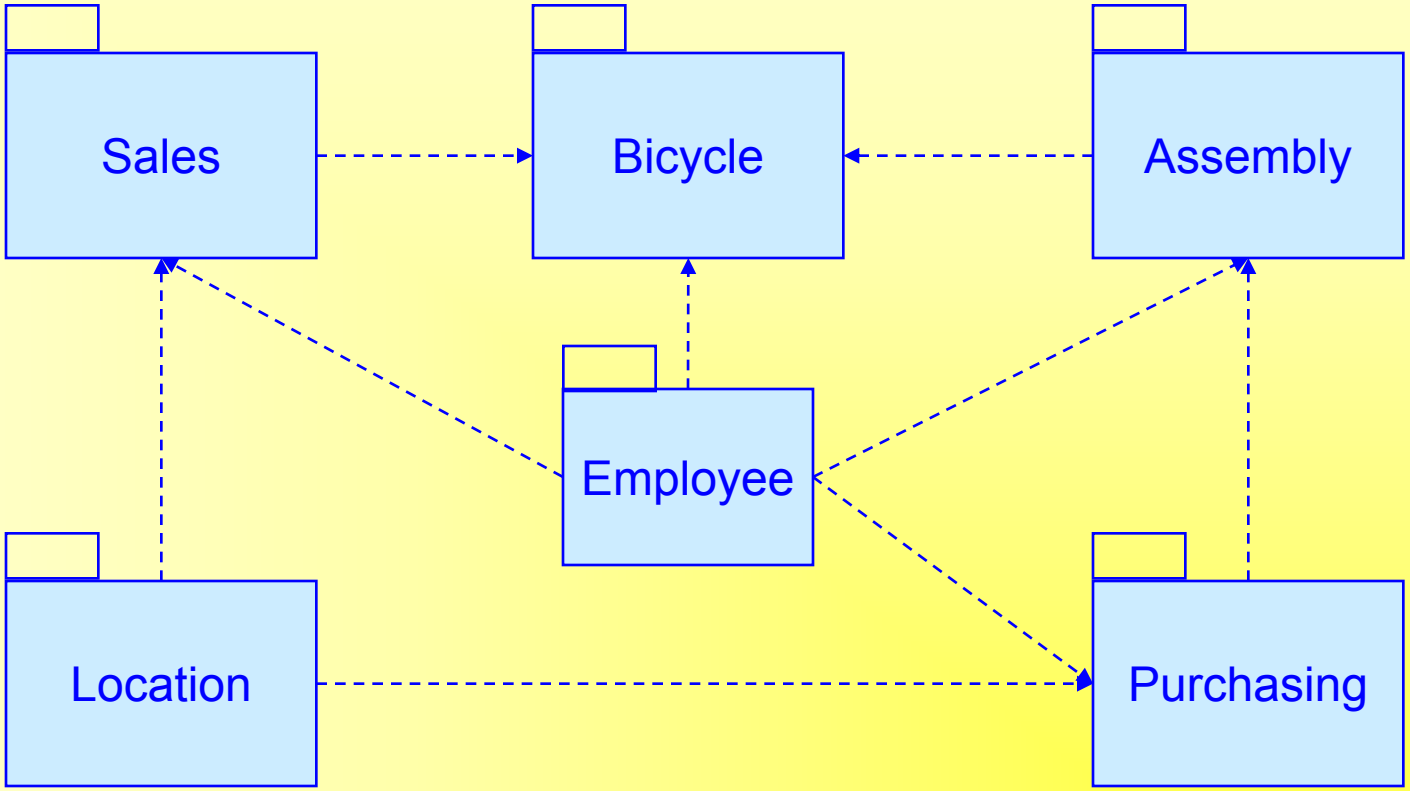
- Project Teams
  - Divide the work
  - Fit pieces together
  - Evaluate progress
- Standards
  - Design
  - Templates
  - Actions
  - Events
  - Objects
    - Naming convention
    - Properties
- Project planning software
  - Schedules
  - Gantt charts
- CASE tools
- Groupware tools
  - Track changes
  - Document work
  - Track revisions

DATABASE

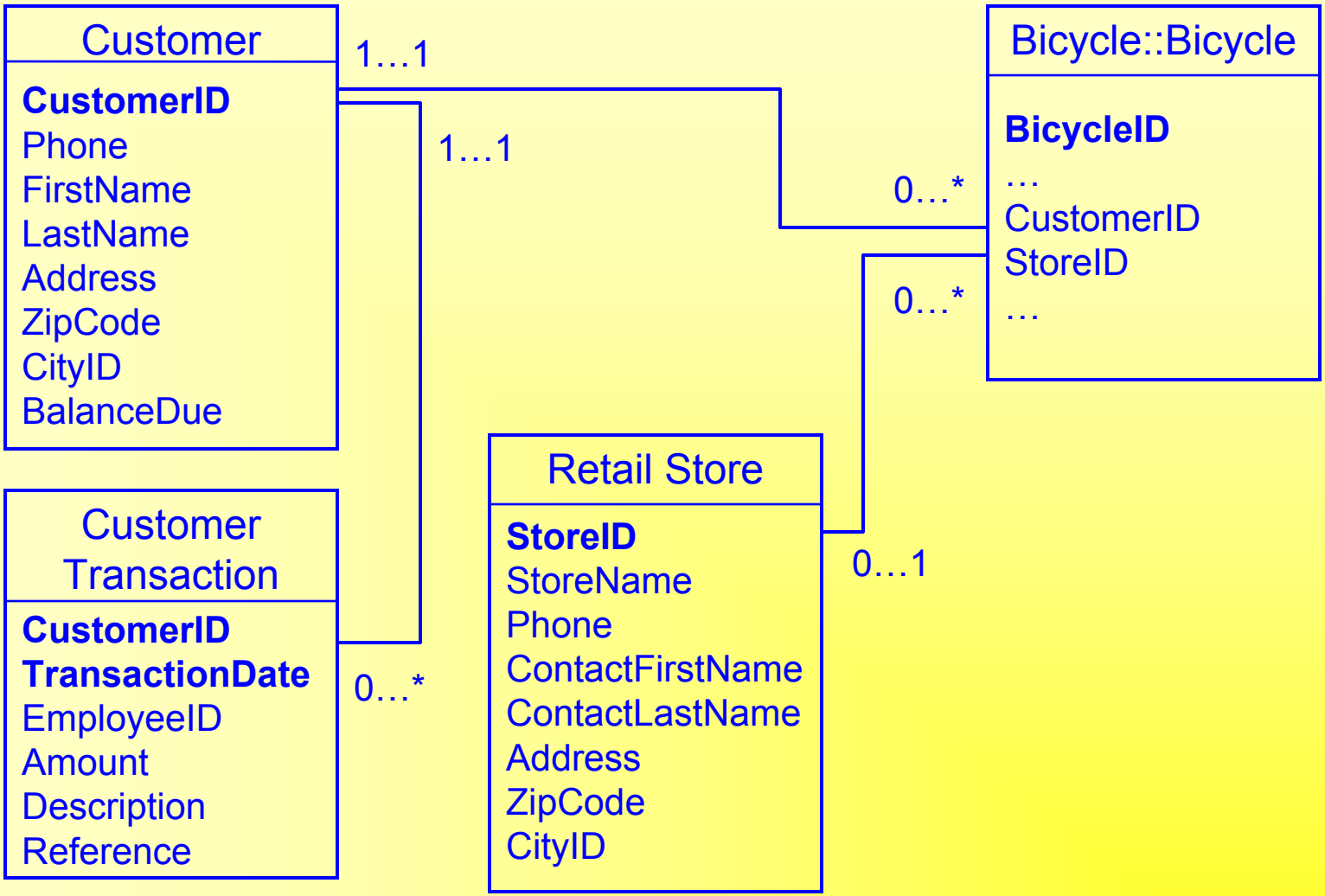
CASE Tools

- Computer-Aided Software Engineering
  - Diagrams (linked)
  - Data Dictionary
  - Teamwork
  - Prototyping
    - Forms
    - Reports
    - Sample data
  - Code generation
  - Reverse Engineering
- Examples
  - Rational Rose
  - Sterling
    - COOL: Dat
    - COOL: Jex (UML)
  - Oracle
  - IBM

Rolling Thunder: Top-Level

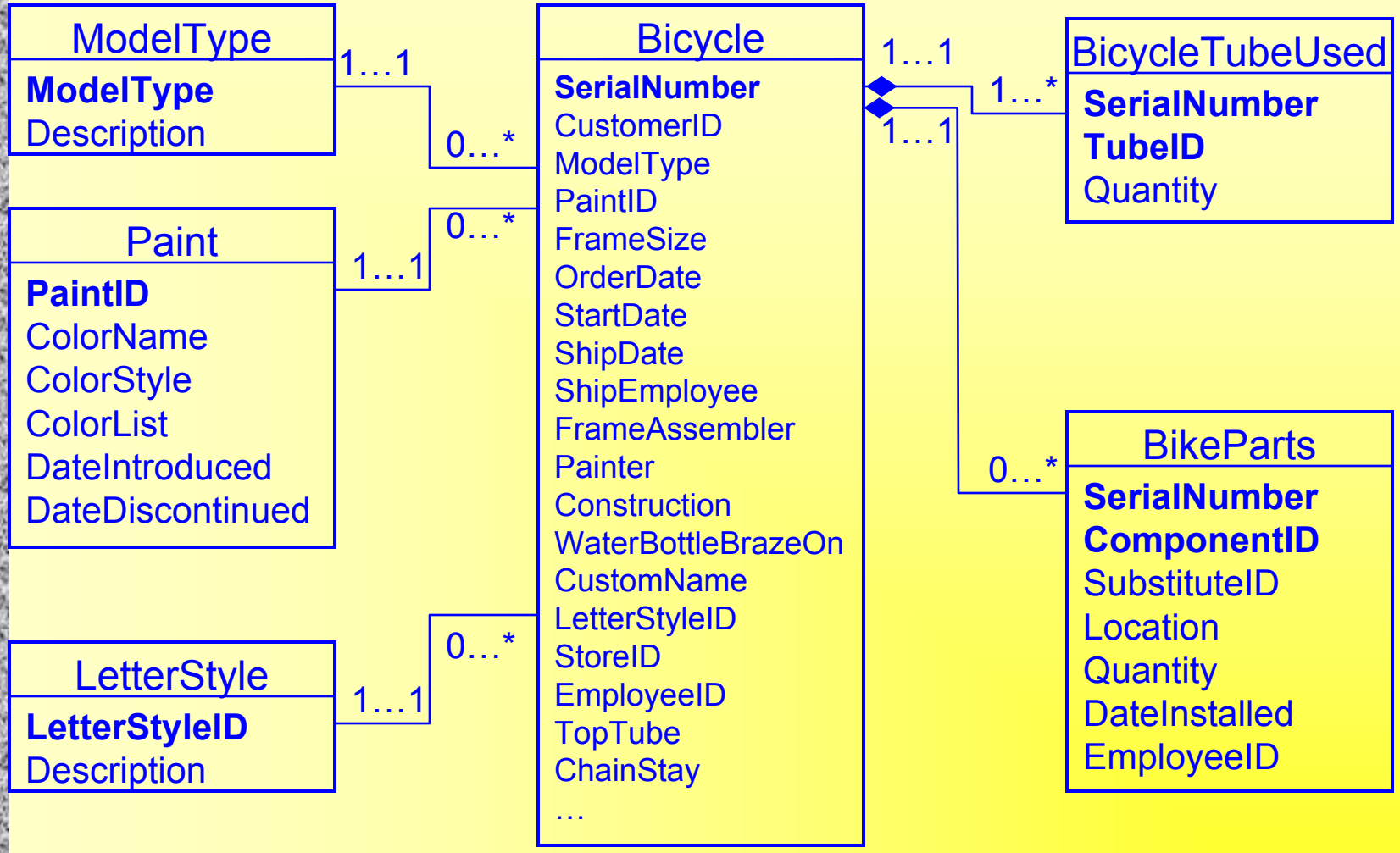


Rolling Thunder: Sales



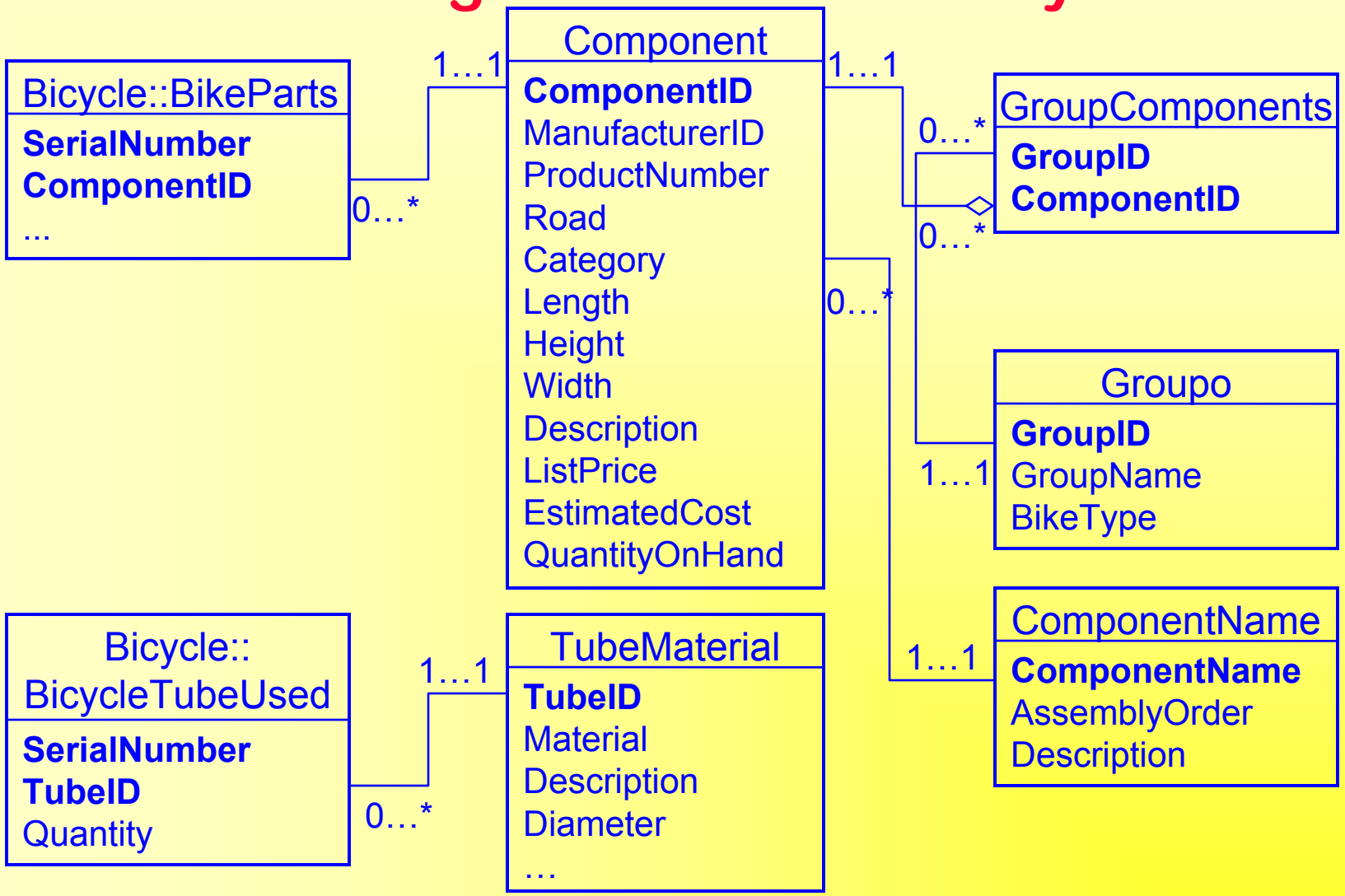
DATABASE

Rolling Thunder: Bicycle



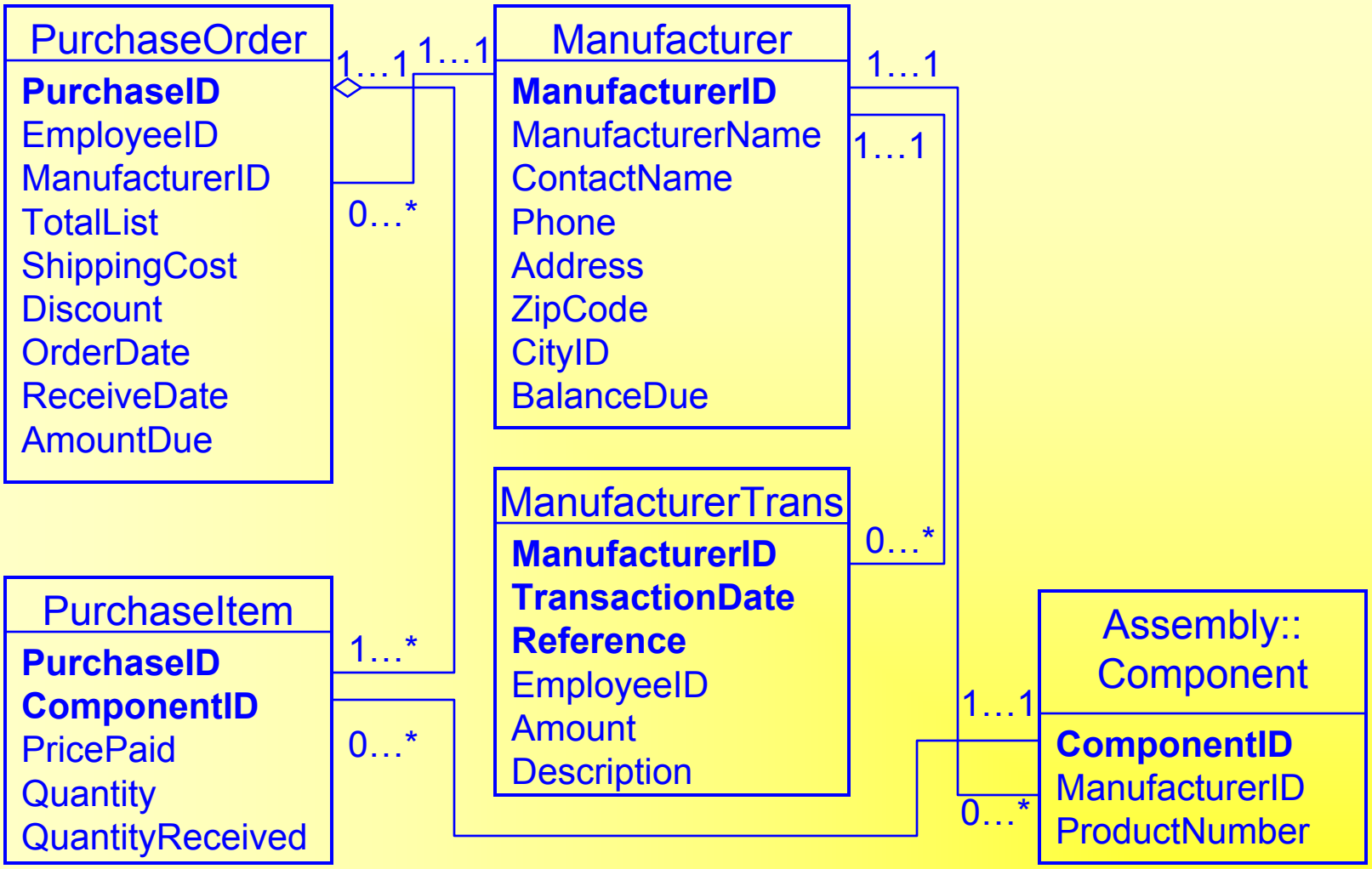
DATABASE

Rolling Thunder: Assembly

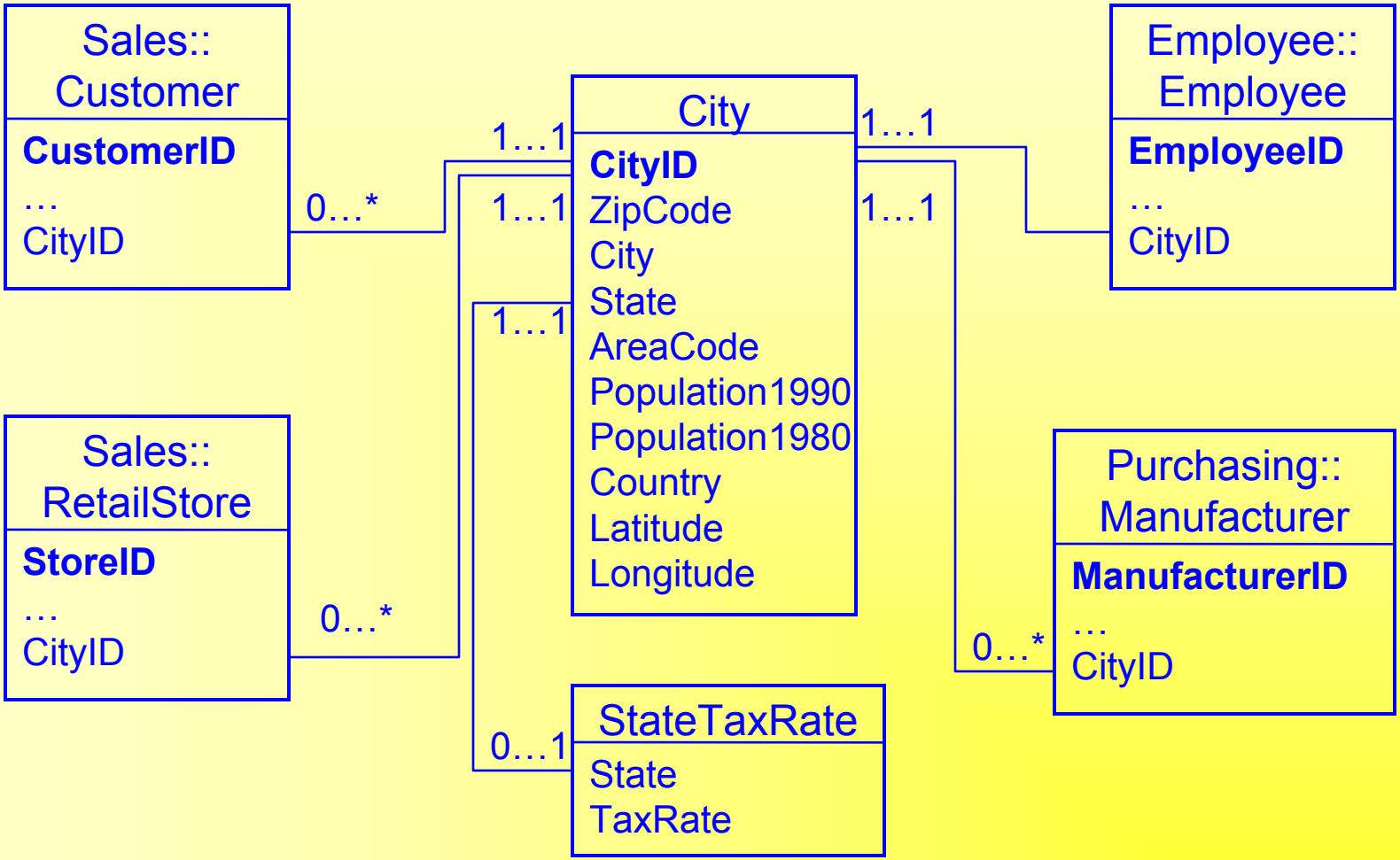


DATABASE

Rolling Thunder: Purchasing

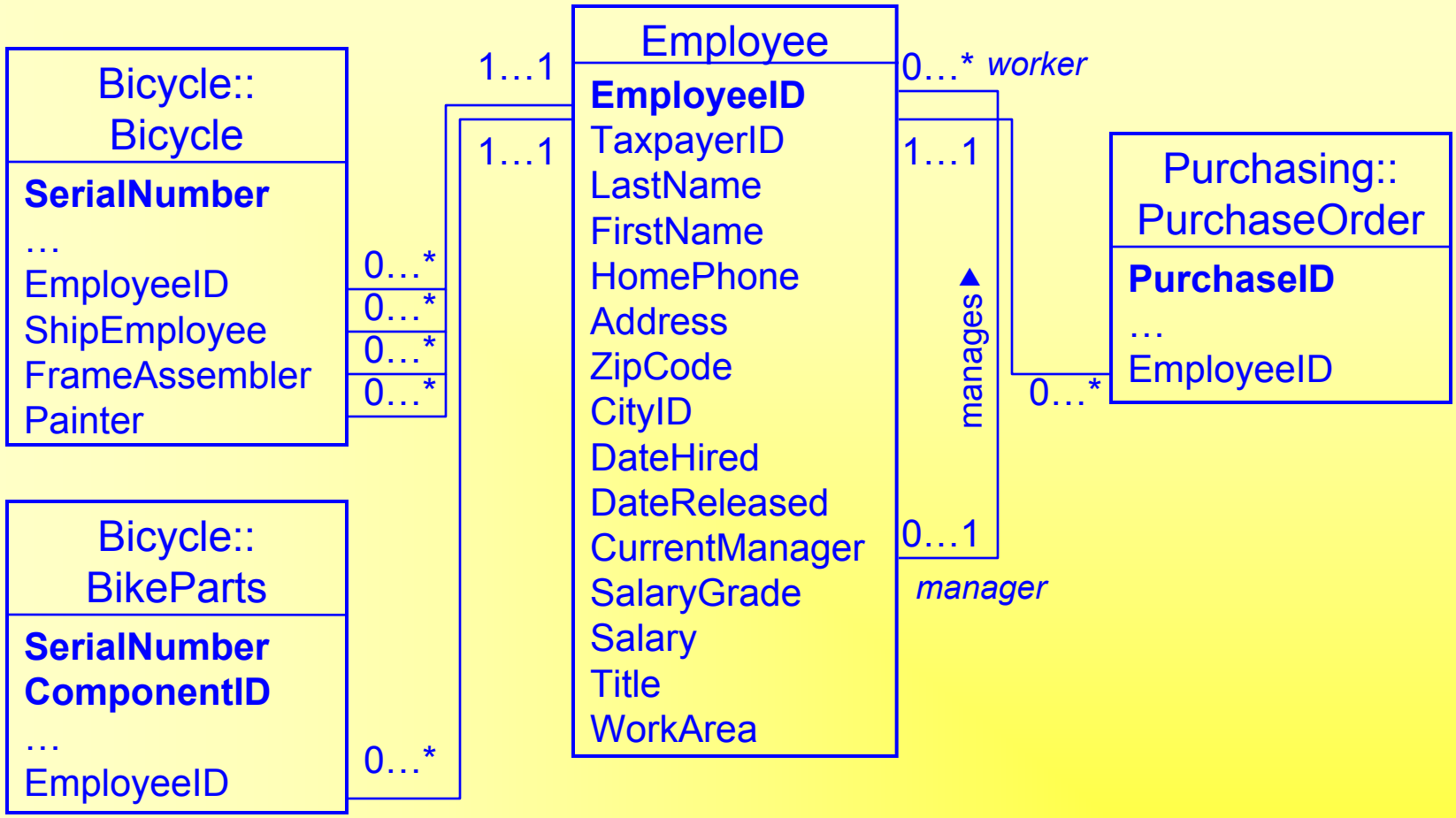


Rolling Thunder: Location



DATABASE

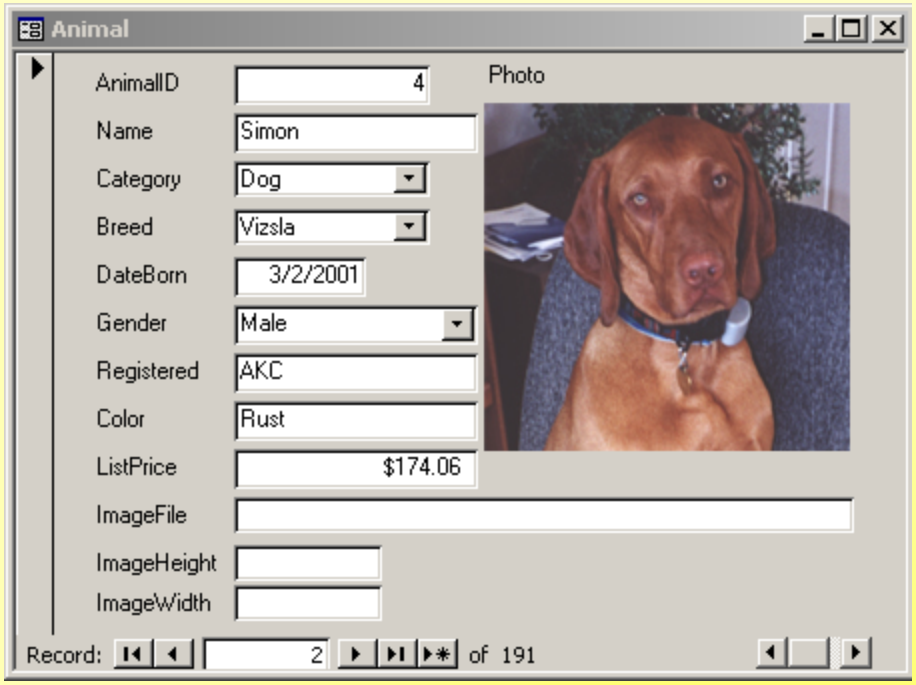
Rolling Thunder: Employee



Rolling Thunder Combined

DATABASE

Application Design



- Simple form based on one table (Animal).
- But also need lookup tables for Category and Breed.

DATABASE

Appendix: DB Design System

- http://time-post.com/dbdesign

- Students and instructors need only an Internet connection and a Java-enabled Web browser.
- Instructor can sign up free by sending email to: jpost@time-post.com
- Instructors set up the class and select assignments.
- Students create accounts and work on the assignments.
- The system provides immediate feedback in the form of comments and questions for each proposed table.

DATABASE

Appendix: Typical Customer Order

Typical Order Form

Order Form

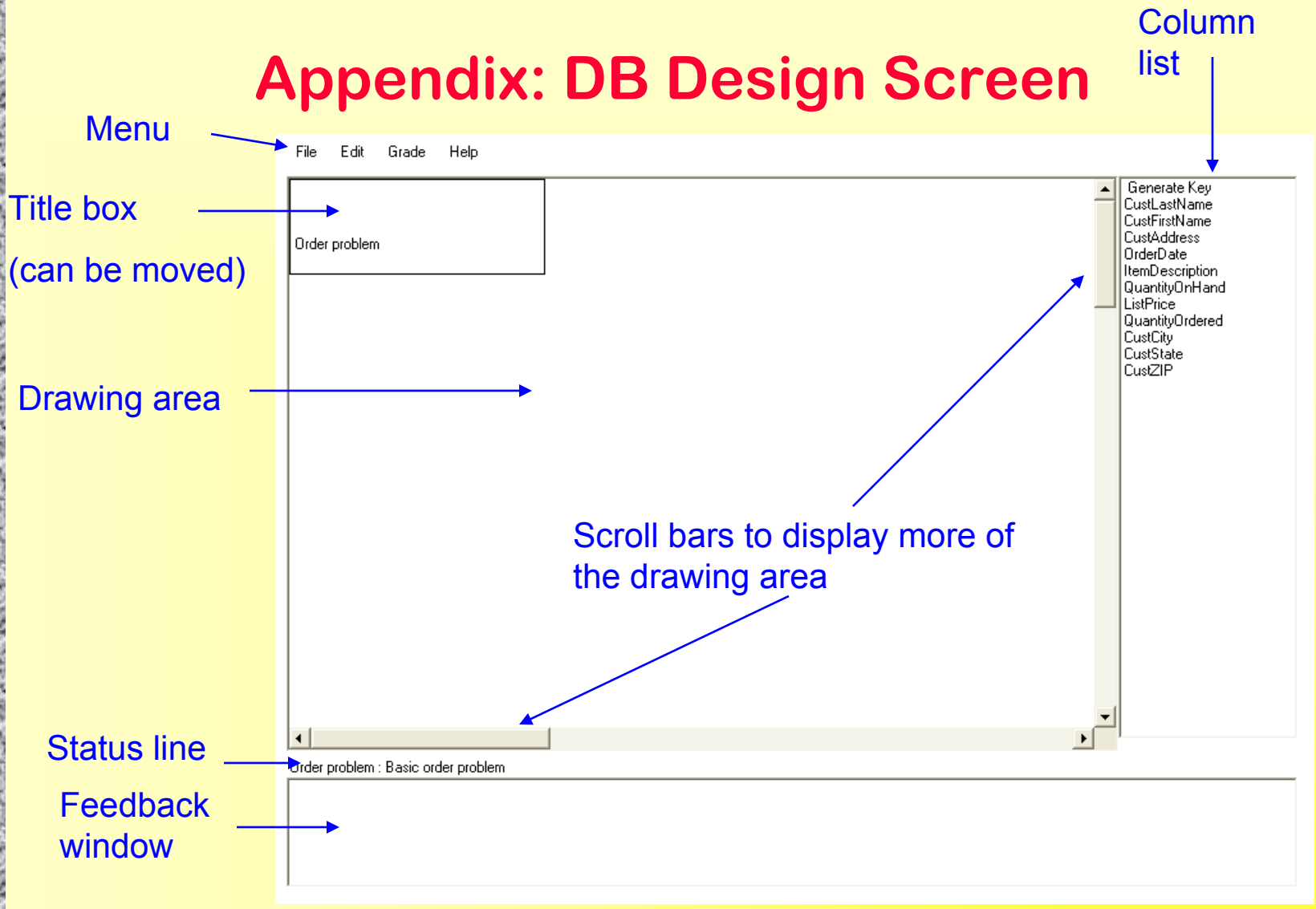
Order # Date

Customer
FirstName LastName
Address
City, State, ZIP

Item	Description	List Price	Quantity	QOH	Value
				Order Total	

DATABASE

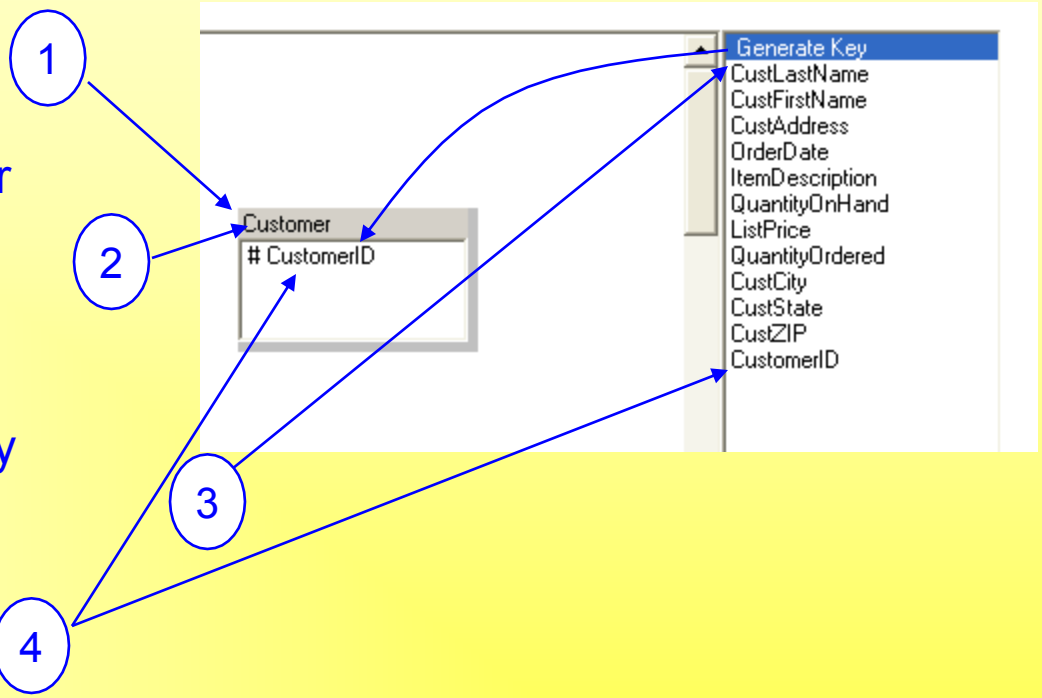
Appendix: DB Design Screen



DATABASE

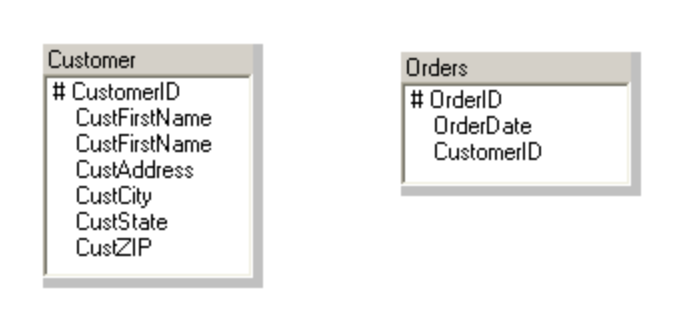
Appendix: Adding a Table and a Key

- Right click in the main drawing window and select the option to Add table.
- Right click the gray bar at the top of the table, select the Rename table option and enter “Customer”
- Drag the Generate Key item onto the new Customer table.
- Right click on the new column name, select the Rename option and enter “CustomerID”



DATABASE

Appendix: Two Tables

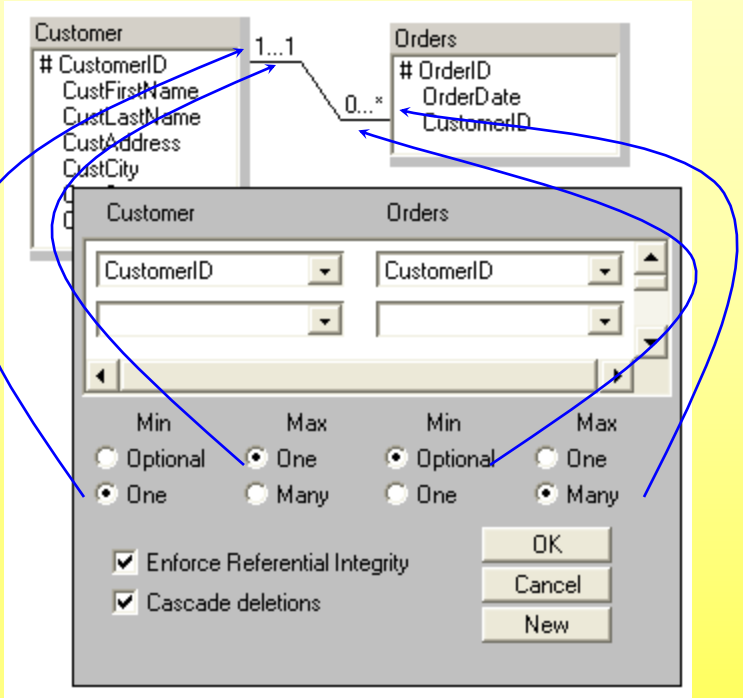


- The Customer table has a generated key of CustomerID
- Each column in the table represents data collected for each customer.
- Each column depends completely on the primary key.
- Each Order is identified by a unique OrderID generated by the database system.
- The CustomerID column is used because the customer number can be used to look up the corresponding data in the Customer table.

DATABASE

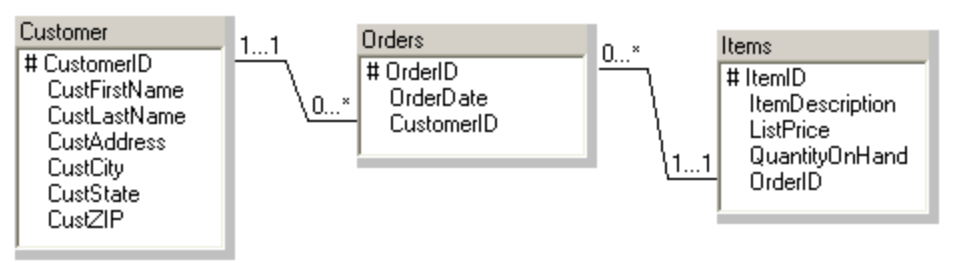
Appendix: Relationships—Linking Tables

- Drag the CustomerID column from the Customer table and drop it on the CustomerID column in the Orders table.
- For the Min value in Customer, select One instead of Optional.
- Click the OK button to accept the relationship definition.



DATABASE

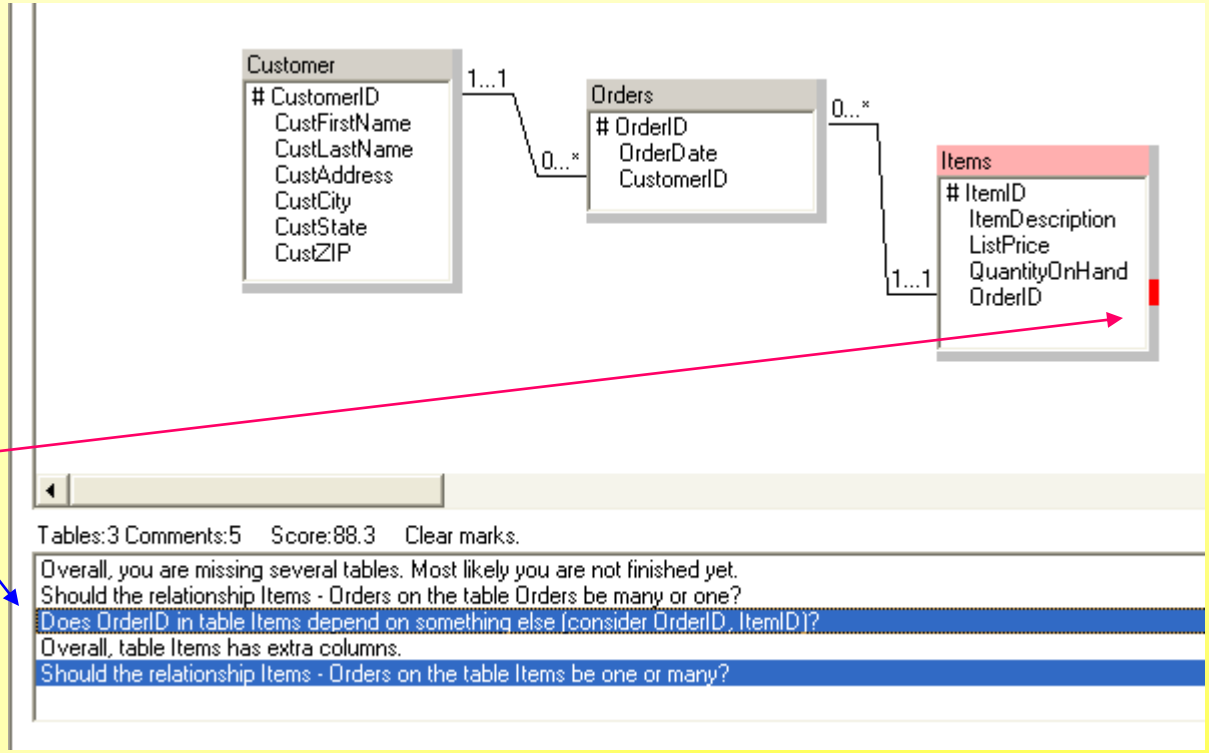
Appendix: Creating Problems



DATABASE

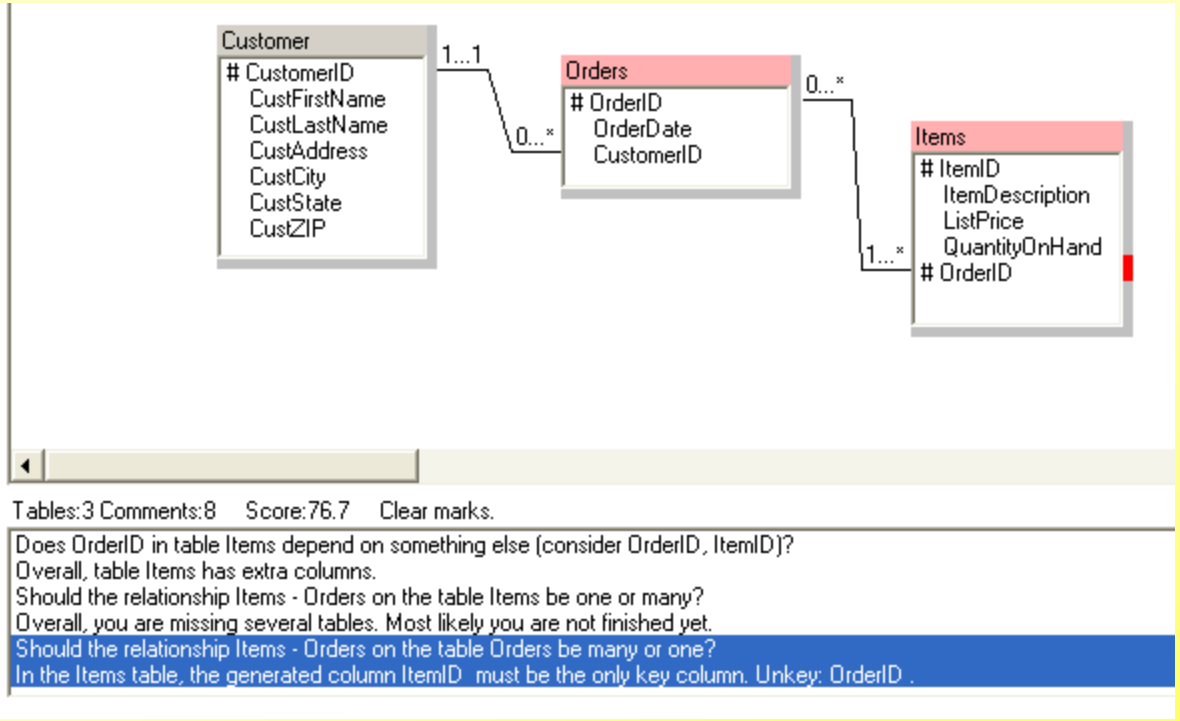
Appendix: Detecting Problems (Grading)

Double click a line to mark the errors.



DATABASE

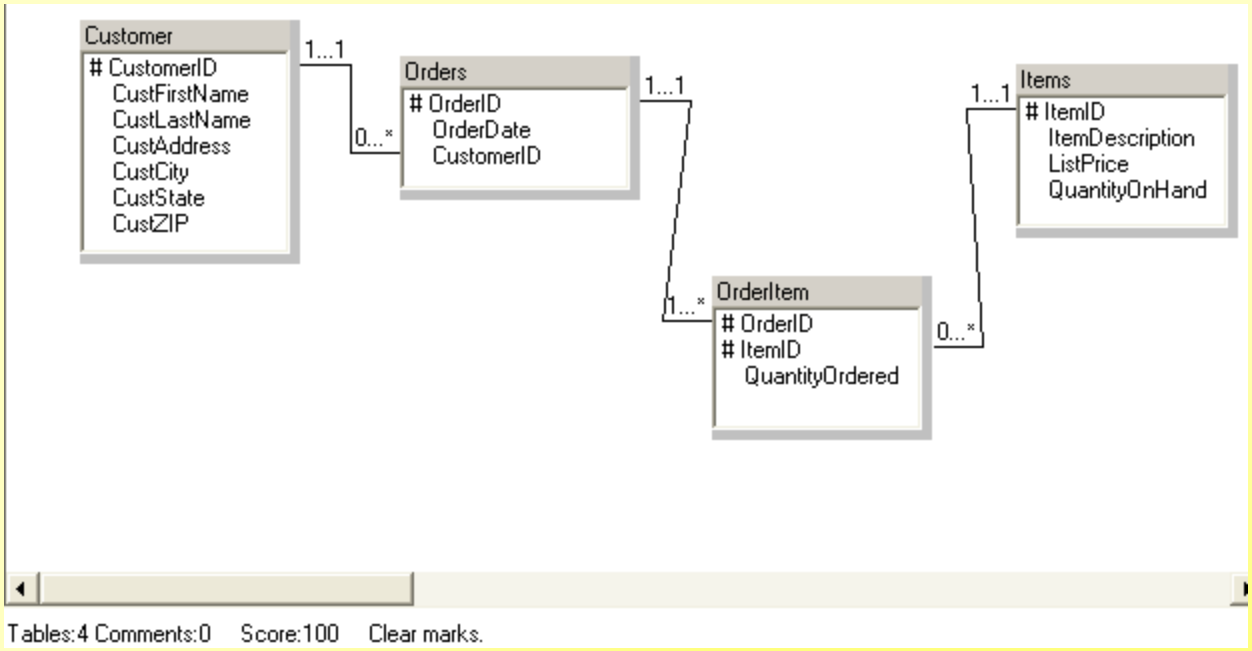
Appendix: Testing a Change



- Attempted fix
  - Make the relationship many-to-many
  - Make OrderID a key
- But, the score went down!!!

DATABASE

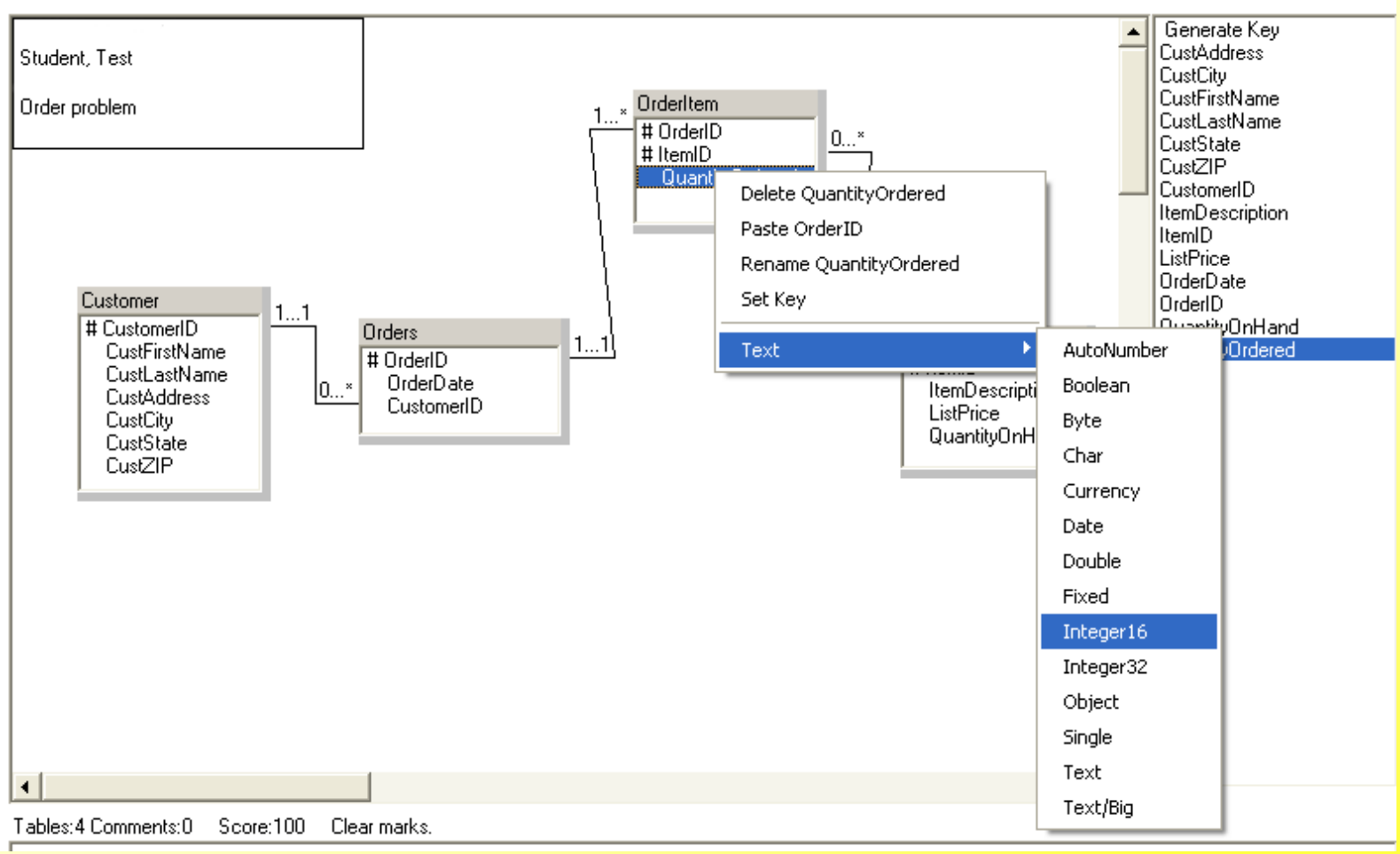
Appendix: A Solution



- The intermediate table OrderItem converts the many-to-many relationship into two one-to-many relationships.
- Both OrderID and ItemID are keys, indicating that each order can have many items, and each item can be sold on many orders.

DATABASE

Appendix: Data Types



Right click the column names and set the data type.

Database Management Systems

Chapter 3

Data Normalization

Jerry Post

DATABASE

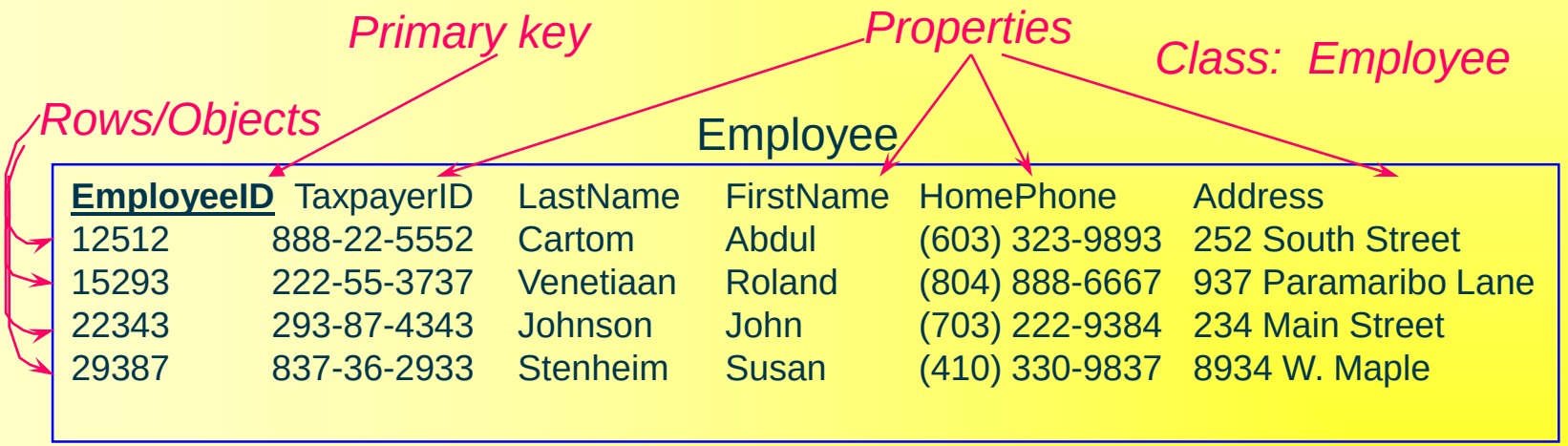
Why Normalization?

- Need standardized data definition
  - Advantages of DBMS require careful design
  - Define data correctly and the rest is much easier
  - It especially makes it easier to expand database later
  - Method applies to most models and most DBMS
- Similar to Entity-Relationship
- Similar to Objects (without inheritance and methods)
- Goal: Define tables carefully
  - Save space
  - Minimize redundancy
  - Protect data

DATABASE

Definitions

- Relational database: A collection of tables.
- Table: A collection of columns (attributes) describing an entity. Individual objects are stored as rows of data in the table.
- Property (attribute): a characteristic or descriptor of a class or entity.
- Every table has a primary key.
  - The smallest set of columns that uniquely identifies any row
  - Primary keys can span more than one column (concatenated keys)
  - We often create a primary key to insure uniqueness (e.g., CustomerID, Product#, . . .) called a surrogate key.



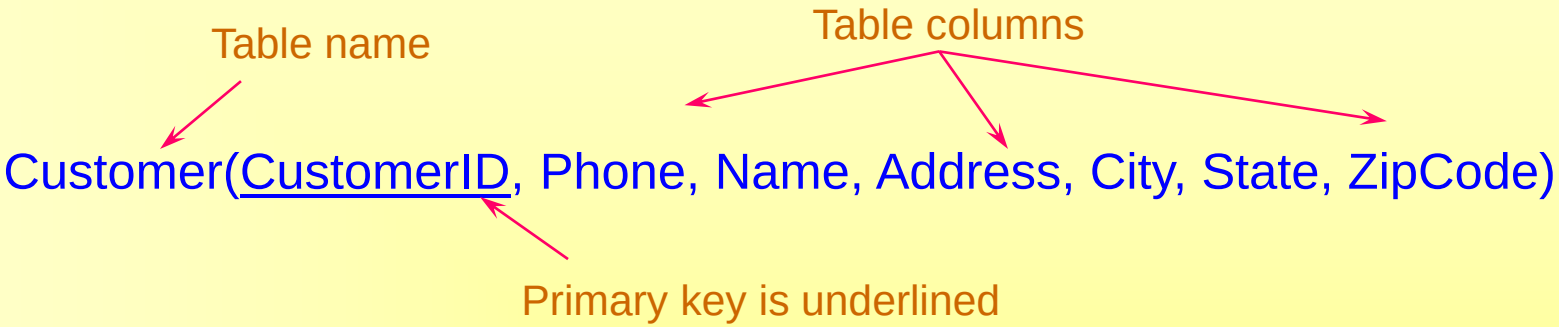
DATABASE

Keys

- Primary key
  - Every table (object) must have a primary key
  - Uniquely identifies a row (one-to-one)
- Concatenated (or composite) key
  - Multiple columns needed for primary key
  - Identify repeating relationships (1 : M or M : N)
- Key columns are <u>underlined</u>
- First step
  - Collect user documents
  - Identify possible keys: unique or repeating relationships

DATABASE

Notation



CustomerID	Phone	LastName	FirstName	Address	City	State	Zipcode
1	502-666-7777	Johnson	Martha	125 Main Street	Alvaton	KY	42122
2	502-888-6464	Smith	Jack	873 Elm Street	Bowling Green	KY	42101
3	502-777-7575	Washington	Elroy	95 Easy Street	Smith's Grove	KY	42171
4	502-333-9494	Adams	Samuel	746 Brown Drive	Alvaton	KY	42122
5	502-474-4746	Rabitz	Victor	645 White Avenue	Bowling Green	KY	42102
6	616-373-4746	Steinmetz	Susan	15 Speedway Drive	Portland	TN	37148
7	615-888-4474	Lasater	Les	67 S. Ray Drive	Portland	TN	37148
8	615-452-1162	Jones	Charlie	867 Lakeside Drive	Castalian Springs	TN	37031
9	502-222-4351	Chavez	Juan	673 Industry Blvd.	Caneyville	KY	42721
10	502-444-2512	Rojo	Maria	88 Main Street	Cave City	KY	42127

DATABASE

Identifying Key Columns

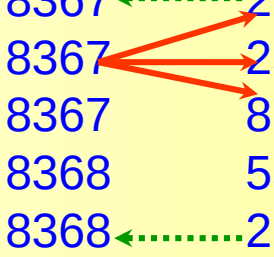
Orders

OrderID	Date	Customer
8367	5-5-04	6794
8368	5-6-04	9263

Each order has only one customer. So Customer is **not** part of the key.

OrderItems

OrderID	Item	Quantity
8367	229	2
8367	253	4
8367	876	1
8368	555	4
8368	229	1



Each order has many items.
Each item can appear on many orders.
So OrderID and Item are **both** part of the key.

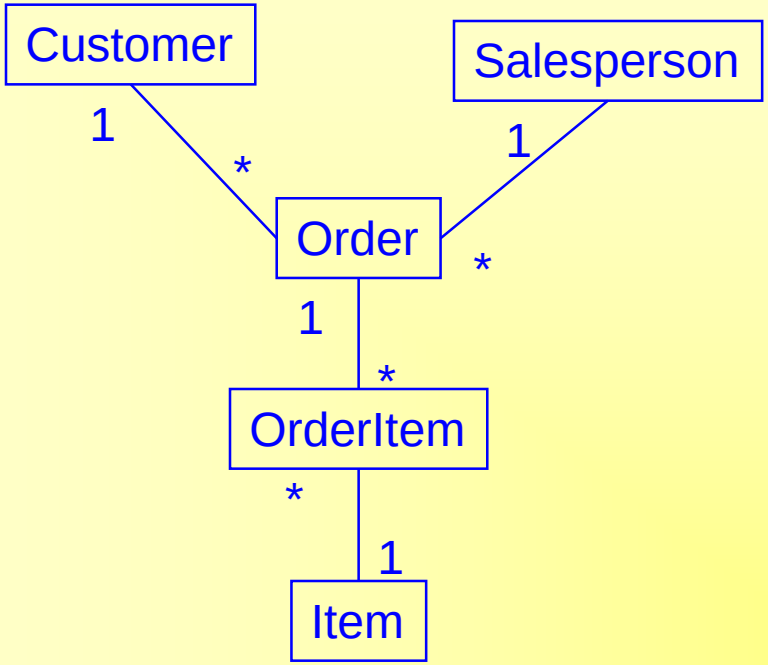
DATABASE

Surrogate Keys

- Real world keys sometimes cause problems in a database.
- Example: Customer
  - Avoid phone numbers: people may not notify you when numbers change.
  - Avoid SSN (privacy and most businesses are not authorized to ask for verification, so you could end up with duplicate values)
- Often best to let the DBMS generate unique values
  - Access: AutoNumber
  - SQL Server: Identity
  - Oracle: Sequences (but require additional programming)
- Drawback: Numbers are not related to any business data, so the application needs to hide them and provide other look up mechanisms.

DATABASE

Common Order System



Customer(CustomerID, Name, Address, City, Phone)

Salesperson(EmployeeID, Name, Commission, DateHired)

Order(OrderID, OrderDate, CustomerID, EmployeeID)

OrderItem(OrderID, ItemID, Quantity)

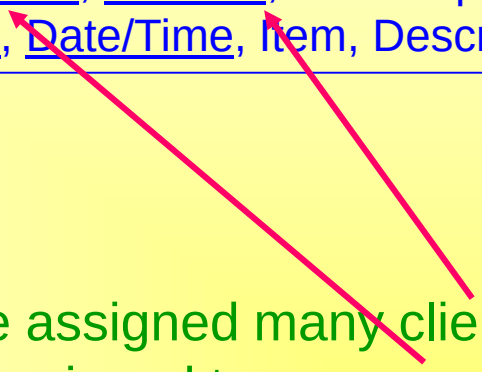
Item(ItemID, Description, ListPrice)

DATABASE

Client Billing Example

Client Billing

Client(ClientID, Name, Address, BusinessType)
Partner(PartnerID, Name, Speciality, Office, Phone)
PartnerAssignment(PartnerID, ClientID, DateAcquired)
Billing(ClientID, PartnerID, Date/Time, Item, Description, Hours, AmountBilled)



Each partner can be assigned many clients.
Each client can be assigned to many partners.

DATABASE

Client Billing--Different Rules

Client(ClientID, Name, Address, BusinessType)

Partner(PartnerID, Name, Speciality, Office, Phone)

PartnerAssignment(PartnerID, ClientID, DateAcquired)

Billing(ClientID, PartnerID, Date/Time, Item, Description, Hours, AmountBilled)

combine

Each client is assigned to only one partner.
Cannot key PartnerID.
Combine Client and PartnerAssignment tables, since they have the same key.

DATABASE

Client Billing--New Assumptions

Billing

ClientID	PartnerID	Date/Time	Item	Description	Hours	AmountBilled
115	963	8-4-04 10:03	967	Stress analysis	2	$500
295	967	8-5-04 11:15	754	New Design	3	$750
115	963	8-8-04 09:30	967	Stress analysis	2.5	$650

More realistic assumptions for a large firm.
Each Partner may work with many clients.
Each client may work with many partners.
Each partner and client may work together many times.
The identifying feature is the date/time of the service.

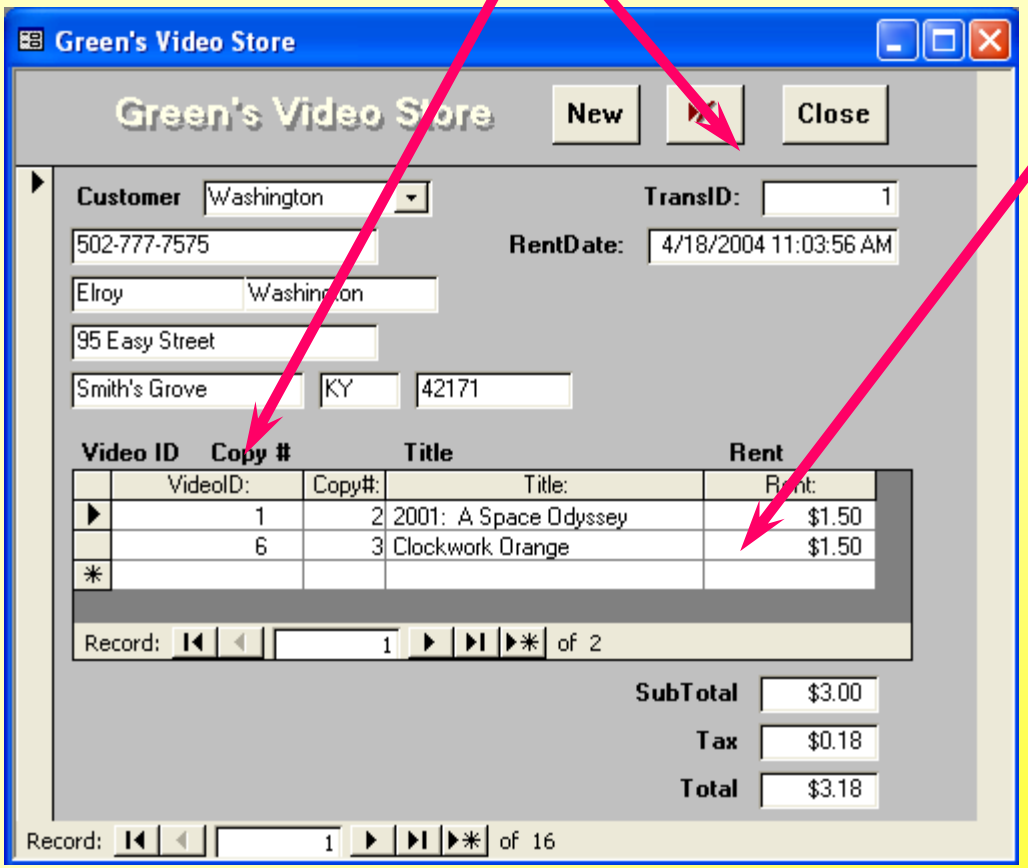
What happens if you do not include Date/Time as a key?

DATABASE

Sample: Video Database

Possible Keys

Repeating section



DATABASE

Initial Objects

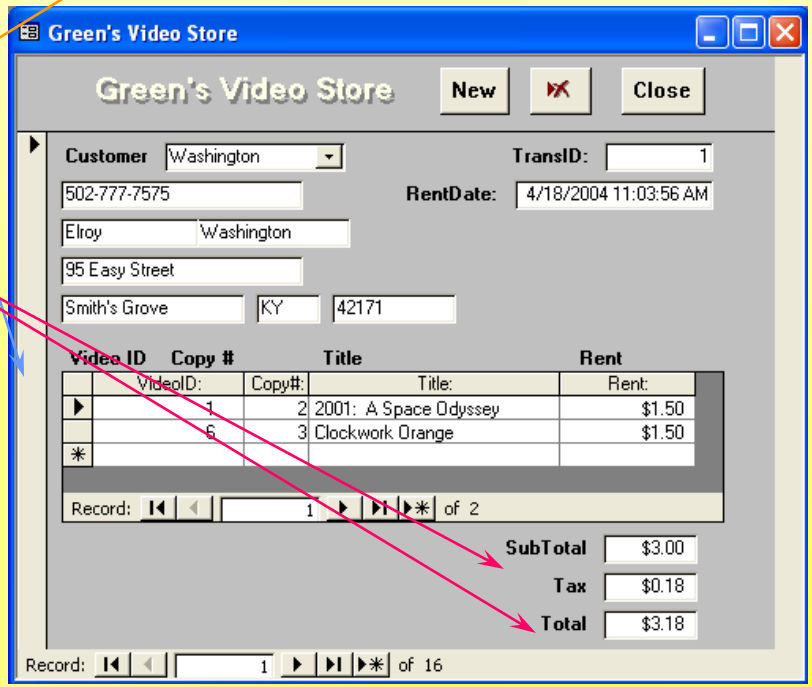
- Customers
  - Key: Assign a CustomerID
  - Sample Properties
    - Name
    - Address
    - Phone
- Videos
  - Key: Assign a VideoID
  - Sample Properties
    - Title
    - RentalPrice
    - Rating
    - Description
- RentalTransaction
  - Event/Relationship
  - Key: Assign TransactionID
  - Sample Properties
    - CustomerID
    - Date
- VideosRented
  - Event/Repeating list
  - Keys: TransactionID + VideoID
  - Sample Properties
    - VideoCopy#

DATABASE

Initial Form Evaluation

RentalForm(TransID, RentDate, CustomerID, Phone, Name, Address, City, State, ZipCode,
(VideoID, Copy#, Title, Rent))

- Collect forms from users
- Write down properties
- Find repeating groups (. . .)
- Look for potential keys: key
- Identify computed values
- Notation makes it easier to identify and solve problems
- Results equivalent to diagrams, but will fit on one or two pages



DATABASE

Problems with Repeating Sections

RentalForm(TransID, RentDate, CustomerID, Phone, Name, Address, City, State, ZipCode, (VideoID, Copy#, Title, Rent))

Storing data in this raw form would not work very well. For example, repeating sections will cause problems.

Note the duplication of data.

Also, what if a customer has not yet checked out a movie--where do we store that customer's data?

Repeating Section

Causes duplication

TransID	RentDate	CustomerID	LastName	Phone	Address	VideoID	Copy#	Title	Rent
1	4/18/04	3	Washington	502-777-7575	95 Easy Street	1	2	2001: A Space Odyssey	$1.50
1	4/18/04	3	Washington	502-777-7575	95 Easy Street	6	3	Clockwork Orange	$1.50
2	4/30/04	7	Lasater	615-888-4474	67 S. Ray Drive	8	1	Hopscotch	$1.50
2	4/30/04	7	Lasater	615-888-4474	67 S. Ray Drive	2	1	Apocalypse Now	$2.00
2	4/30/04	7	Lasater	615-888-4474	67 S. Ray Drive	6	1	Clockwork Orange	$1.50
3	4/18/04	8	Jones	615-452-1162	867 Lakeside Drive	9	1	Luggage Of The Gods	$2.50
3	4/18/04	8	Jones	615-452-1162	867 Lakeside Drive	15	1	Fabulous Baker Boys	$2.00
3	4/18/04	8	Jones	615-452-1162	867 Lakeside Drive	4	1	Boy And His Dog	$2.50
4	4/18/04	3	Washington	502-777-7575	95 Easy Street	3	1	Blues Brothers	$2.00
4	4/18/04	3	Washington	502-777-7575	95 Easy Street	8	1	Hopscotch	$1.50
4	4/18/04	3	Washington	502-777-7575	95 Easy Street	13	1	Surf Nazis Must Die	$2.50
4	4/18/04	3	Washington	502-777-7575	95 Easy Street	17	1	Witches of Eastwick	$2.00

DATABASE

Problems with Repeating Sections

- Store repeating data
  - Allocate space
  - How much?
    - Can't be short
    - Wasted space
- e.g., How many videos will be rented at one time?
- A better definition eliminates this problem.

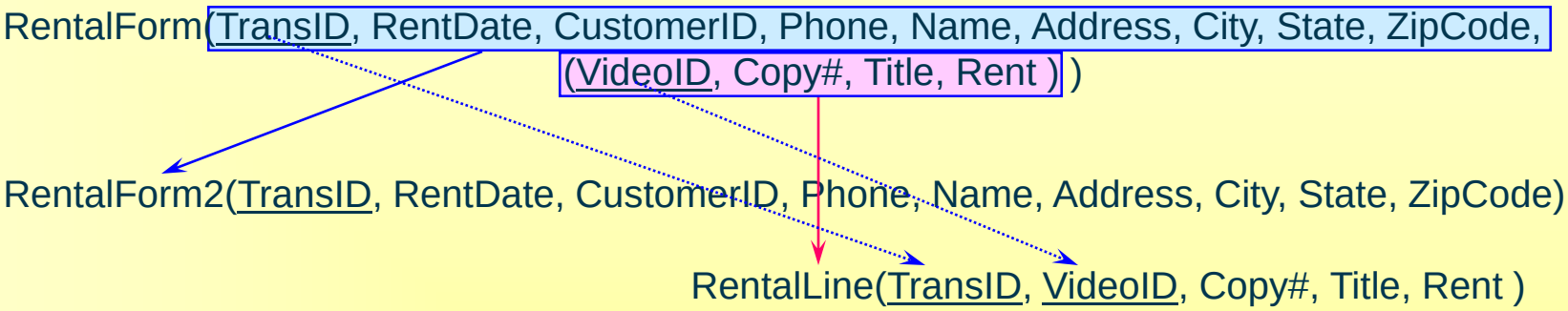
Name
Phone
Address
City
State
ZipCode

Customer Rentals

	VideoID	Copy#	Title	Rent
1.	6	1	Clockwork Orange	1.50
2.	8	2	Hopscotch	1.50
3.				
4.			{Unused Space}	
5.				

Not in First Normal Form

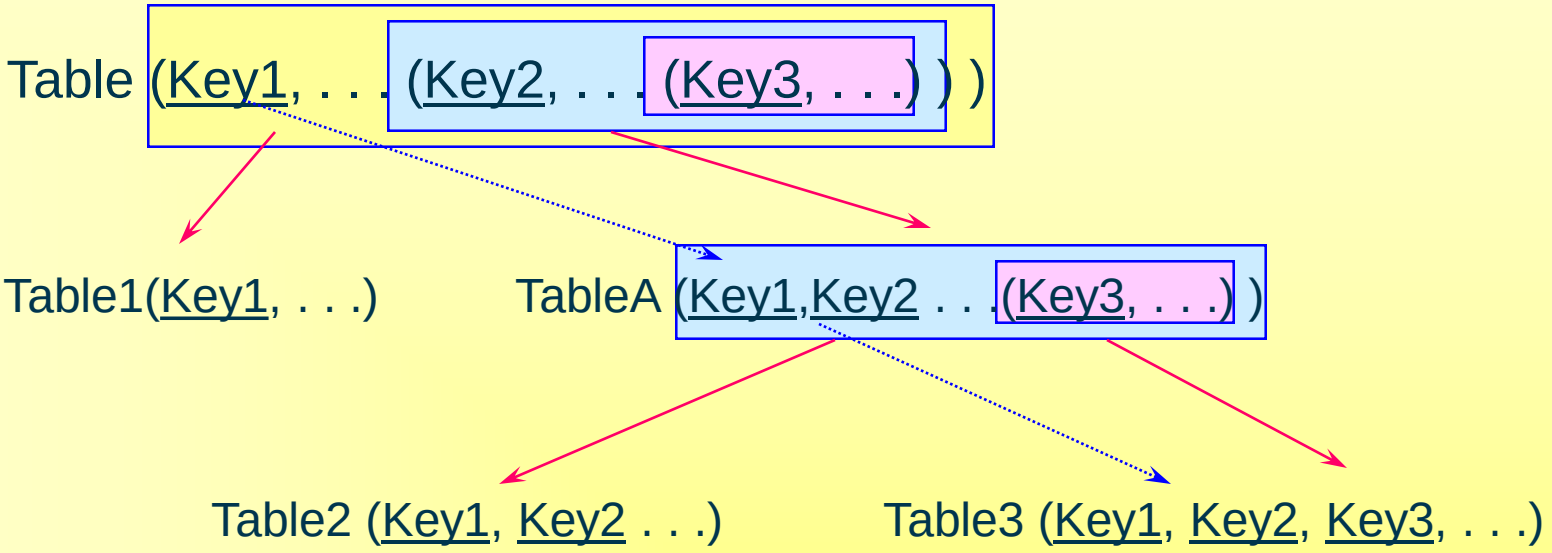
First Normal Form



- Remove repeating sections
  - Split into two tables
  - Bring key from main and repeating section
- RentalLine(TransID, VideoID, Copy#, . . .)
  - Each transaction can have many videos (key VideoID)
  - Each video can be rented on many transactions (key TransID)
  - For each TransID **and** VideoID, only one Copy# (no key on Copy#)

DATABASE

Nested Repeating Sections



- Nested: Table (Key1, aaa. . . (Key2, bbb. . . (Key3, ccc. . .)))
- First Normal Form (1NF)
  - Table1(Key1, aaa . . .)
  - Table2(Key1, Key2, bbb . .)
  - Table3(Key1, Key2, Key3, ccc. . .)

DATABASE

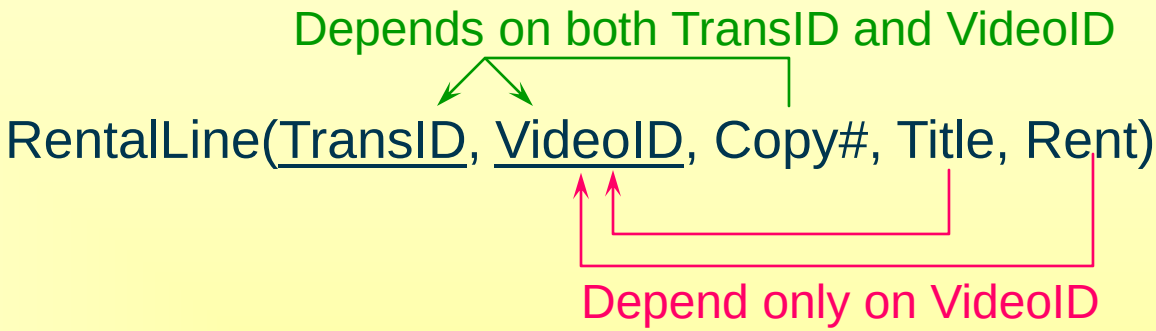
First Normal Form Problems (Data)

TransID	RentDate	CustID	Phone	LastName	FirstName	Address	City	State	ZipCode
1	4/18/04	3	502-777-7575	Washington	Elroy	95 Easy Street	Smith's Grove	KY	42171
2	4/30/04	7	615-888-4474	Lasater	Les	67 S. Ray Drive	Portland	TN	37148
3	4/18/04	8	615-452-1162	Jones	Charlie	867 Lakeside Drive	Castalian Springs	TN	37031
4	4/18/04	3	502-777-7575	Washington	Elroy	95 Easy Street	Smith's Grove	KY	42171

- 1NF splits repeating groups
- Still have problems
  - Replication
  - Hidden dependency:
  - If a video has not been rented yet, then what is its title?

TransID	VideoID	Copy#	Title	Rent
1	1	2	2001: A Space Odyssey	$1.50
1	6	3	Clockwork Orange	$1.50
2	8	1	Hopscotch	$1.50
2	2	1	Apocalypse Now	$2.00
2	6	1	Clockwork Orange	$1.50
3	9	1	Luggage Of The Gods	$2.50
3	15	1	Fabulous Baker Boys	$2.00
3	4	1	Boy And His Dog	$2.50
4	3	1	Blues Brothers	$2.00
4	8	1	Hopscotch	$1.50
4	13	1	Surf Nazis Must Die	$2.50
4	17	1	Witches of Eastwick	$2.00

Second Normal Form Definition



- Each non-key column must depend on the **entire** key.
  - Only applies to concatenated keys
  - Some columns only depend on part of the key
  - Split those into a new table.
- Dependence (definition)
  - If given a value for the key you always know the value of the property in question, then that property is said to depend on the key.
  - If you change part of a key and the questionable property does not change, then the table is **not** in 2NF.

DATABASE

Second Normal Form Example

RentalLine(TransID, VideoID, Copy#, Title, Rent)

VideosRented(TransID, VideoID, Copy#)

Videos(VideoID, Title, Rent)

- Title depends only on VideoID
  - Each VideoID can have only one title
- Rent depends on VideoID
  - This statement is actually a business rule.
  - It might be different at different stores.
  - Some stores might charge a different rent for each video depending on the day (or time).
- Each non-key column depends on the whole key.

Second Normal Form Example (Data)

VideosRented(TransID, VideoID, Copy#)

TransID	VideoID	Copy#
1	1	2
1	6	3
2	2	1
2	6	1
2	8	1
3	4	1
3	9	1
3	15	1
4	3	1
4	8	1
4	13	1
4	17	1

Videos(VideoID, Title, Rent)

VideoID	Title	Rent
1	2001: A Space Odyssey	$1.50
2	Apocalypse Now	$2.00
3	Blues Brothers	$2.00
4	Boy And His Dog	$2.50
5	Brother From Another Planet	$2.00
6	Clockwork Orange	$1.50
7	Gods Must Be Crazy	$2.00
8	Hopscotch	$1.50

(Unchanged)

RentalForm2(TransID, RentDate, CustomerID, Phone, Name, Address, City, State, ZipCode)

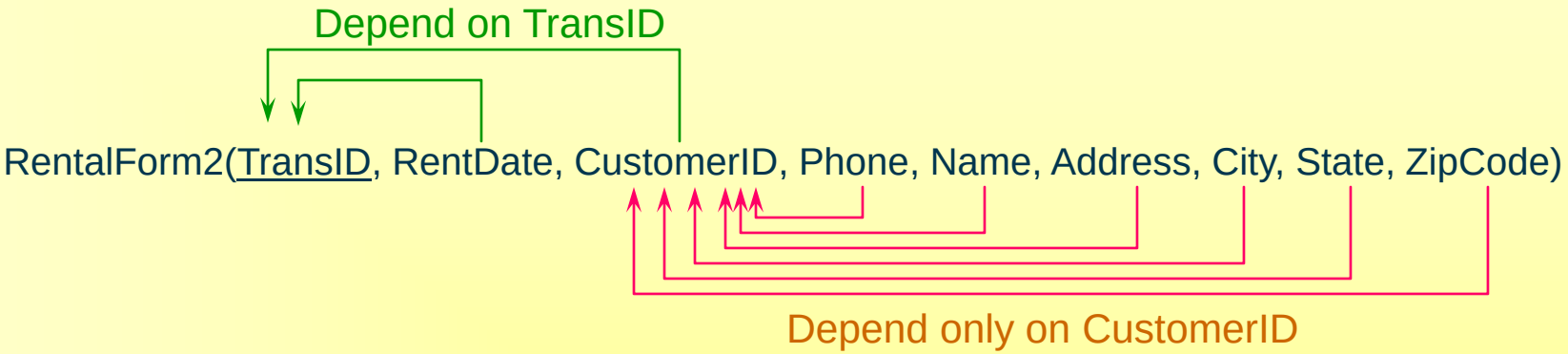
Second Normal Form Problems (Data)

RentalForm2(TransID, RentDate, CustomerID, Phone, Name, Address, City, State, ZipCode)

TransID	RentDate	CustID	Phone	LastName	FirstName	Address	City	State	ZipCode
1	4/18/04	3	502-777-7575	Washington	Elroy	95 Easy Street	Smith's Grove	KY	42171
2	4/30/04	7	615-888-4474	Lasater	Les	67 S. Ray Drive	Portland	TN	37148
3	4/18/04	8	615-452-1162	Jones	Charlie	867 Lakeside Drive	Castalian Springs	TN	37031
4	4/18/042	3	502-777-7575	Washington	Elroy	95 Easy Street	Smith's Grove	KY	42171

- Even in 2NF, problems remain
  - Replication
  - Hidden dependency
  - If a customer has not rented a video yet, where do we store their personal data?
- Solution: split table.

Third Normal Form Definition



- Each non-key column must depend on **nothing but** the key.
  - Some columns depend on columns that are not part of the key.
  - Split those into a new table.
  - Example: Customers name does **not** change for every transaction.
- Dependence (definition)
  - If given a value for the key you always know the value of the property in question, then that property is said to depend on the key.
  - If you change the key and the questionable property does not change, then the table is **not** in 3NF.

DATABASE

Third Normal Form Example

RentalForm2(TransID, RentDate, CustomerID, Phone, Name, Address, City, State, ZipCode)

Rentals(TransID, RentDate, CustomerID)

Customers(CustomerID, Phone, Name, Address, City, State, ZipCode)

- Customer attributes depend only on Customer ID
  - Split them into new table (Customer)
  - Remember to leave CustomerID in Rentals table.
  - We need to be able to reconnect tables.
- 3NF is sometimes easier to see if you identify primary objects at the start--then you would recognize that Customer was a separate object.

DATABASE

Third Normal Form Example Data

RentalForm2(TransID, RentDate, CustomerID, Phone, Name, Address, City, State, ZipCode

Rentals(TransID, RentDate, CustomerID)

TransID	RentDate	CustomerID
1	4/18/04	3
2	4/30/04	7
3	4/18/04	8
4	4/18/04	3

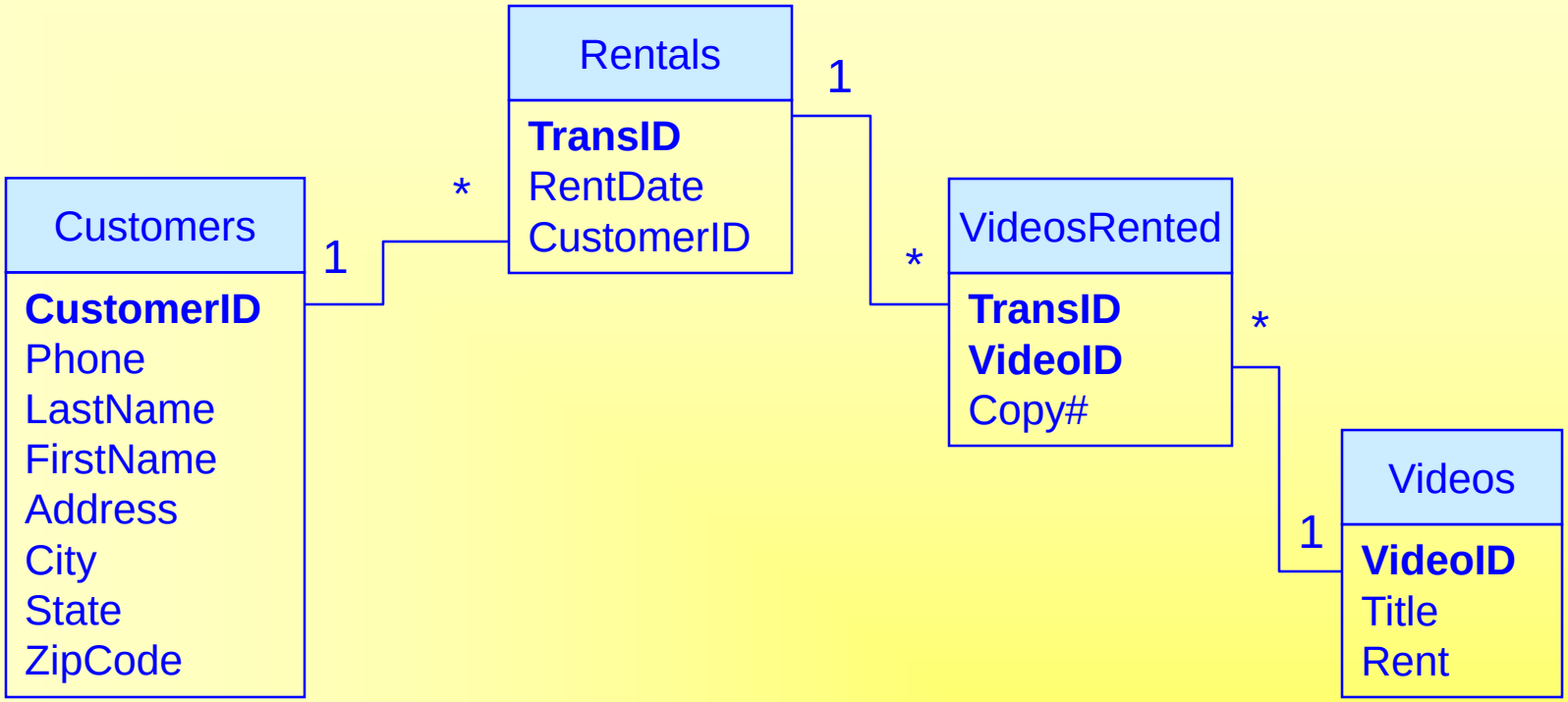
Customers(CustomerID, Phone, Name, Address, City, State, ZipCode)

CustomerID	Phone	LastName	FirstName	Address	City	State	ZipCode
1	502-666-7777	Johnson	Martha	125 Main Street	Alvaton	KY	42122
2	502-888-6464	Smith	Jack	873 Elm Street	Bowling Green	KY	42101
3	502-777-7575	Washington	Elroy	95 Easy Street	Smith's Grove	KY	42171
4	502-333-9494	Adams	Samuel	746 Brown Drive	Alvaton	KY	42122
5	502-474-4746	Rabitz	Victor	645 White Avenue	Bowling Green	KY	42102
6	615-373-4746	Steinmetz	Susan	15 Speedway Drive	Portland	TN	37148
7	615-888-4474	Lasater	Les	67 S. Ray Drive	Portland	TN	37148
8	615-452-1162	Jones	Charlie	867 Lakeside Drive	Castalian Springs	TN	37031
9	502-222-4351	Chavez	Juan	673 Industry Blvd.	Caneyville	KY	42721
10	502-444-2512	Rojo	Maria	88 Main Street	Cave City	KY	42127

VideosRented(TransID, VideoID, Copy#)

Videos(VideoID, Title, Rent)

Third Normal Form Tables (3NF)



Rentals(TransID, RentDate, CustomerID)

Customers(CustomerID, Phone, Name, Address, City, State, ZipCode)

VideosRented(TransID, VideoID, Copy#)

Videos(VideoID, Title, Rent)

DATABASE

3NF Rules/Procedure

- Split out repeating sections
  - Be sure to include a key from the parent section in the new piece so the two parts can be recombined.
- Verify that the keys are correct
  - Is each row uniquely identified by the primary key?
  - Are one-to-many and many-to-many relationships correct?
  - Check “many” for keyed columns and “one” for non-key columns.
- **Make sure that each non-key column depends on the whole key and nothing but the key.**
  - No hidden dependencies.

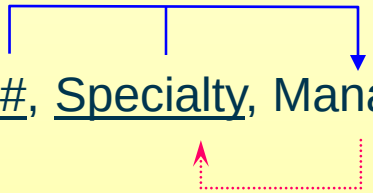
D A T A B A S E

Checking Your Work (Quality Control)

- Look for one-to-many relationships.
  - Many side should be keyed (underlined).
  - e.g., VideosRented(TransID, VideoID, . . .).
  - Check each column and ask if it should be 1 : 1 or 1: M.
  - If add a key, renormalize.
- Verify no repeating sections (1NF)
- Check 3NF
  - Check each column and ask:
  - Does it depend on the whole key and nothing but the key?
- Verify that the tables can be reconnected (joined) to form the original tables (*draw lines*).
- Each table represents one object.
- Enter sample data--look for replication.

DATABASE

Boyce-Codd Normal Form (BCNF)



- Hidden dependency
- Example:
  - Employee-Specialty(E#, Specialty, Manager)
  - Is in 3NF now.
- Business rules.
  - Employee may have many specialties.
  - Each specialty has many managers.
  - Each manager has only one specialty.
  - Employee has only one manager for each specialty.
- Problem is hidden relationship between manager and specialty.
  - Need separate table for manager.
  - But then we don't need to repeat specialty.
- In real life, probably accept the duplication (specialty listed in both tables).

Employee(E#, Manager)

Manager(Manager, Specialty)

Employee(E#, Specialty, Manager)

Manager(Manager, Specialty)

acceptable

Fourth Normal Form (Keys)

EmployeeTasks(EID, Specialty, ToolID)

- Technically, if you keyed every column, any table would be in 3NF, which does not solve any problems.
- In some cases, there are hidden relationships between key properties.
- Example:
  - EmployeeTasks(EID, Specialty, ToolID)
  - In 3NF (BCNF) now.
- Business Rules
  - Each employee has many specialties.
  - Each employee has many tools.
  - Tools and specialties are unrelated

EmployeeSpecialty(EID, Specialty)

EmployeeTools(EID, ToolID)

Domain-Key Normal Form (DKNF)

- DKNF is ultimate goal: table will always be in 4NF, etc.
- Drawbacks
  - No mechanical method to get to DKNF
  - No guarantee a table can be converted to DKNF
- Rules
  - Table => one topic
  - All business rules explicitly written as domain constraints and key relationships.
  - No hidden relationships.

Employee(EID, Name, speciality)

Business rule: *An employee can have many specialties.*
So example is **not** in DKNF, since EID is not unique.

DATABASE

DKNF Examples

Employee(EID, Name, Speciality)

Business rule: *An employee can have many specialties.*
Example is **not** in DKNF: EID is not unique.

Employee(EID, Name, Speciality)

Business rule: *An employee has one name.*
Example is **not** DKNF: hidden relationship between EID and name.

Employee(EID, Name)

EmployeeSpecialty(EID, Speciality)

DKNF Examples

Student(SID, Name, Major, Advisor)

Advisor(FID, Name, Office, Discipline)

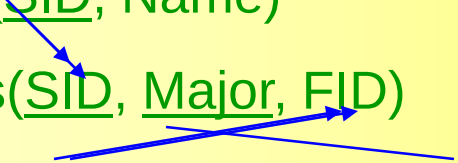
Business rules: *A student can have many advisors, but only one for each major. Faculty can only be advisors for their discipline.*

Not in DKNF: Primary key and hidden rule.

Student(SID, Name)

Advisors(SID, Major, FID)

Faculty(FID, Name, Office, Discipline)



DKNF: Foreign key (Major <--> Discipline) makes advisor rule explicit.

DATABASE

No Hidden Dependencies

- The simple normalization rules:
- Remove repeating sections
- Each non-key column must depend on the whole key and nothing but the key.
- There must be no hidden dependencies.
- Solution: Split the table.
- Make sure you can rejoin the two pieces to recreate the original data relationships.
- For some hidden dependencies within keys, double-check the business assumption to be sure that it is realistic. Sometimes you are better off with a more flexible assumption.

DATABASE

Data Rules and Integrity

- Simple business rules
  - Limits on data ranges
    - Price > 0
    - Salary < 100,000
    - DateHired > 1/12/1995
  - Choosing from a set
    - Gender = M, F, Unknown
    - Jurisdiction=City, County, State, Federal
- Referential Integrity
  - Foreign key values in one table must exist in the master table.
  - Order(O#, Odate, C#,…)
  - C# must exist in the customer table.

Order

O#	Odate	C#	…
1173	1-4-97	321	
1174	1-5-97	938	
1185	1-8-97	337	
1190	1-9-97	321	
1192	1-9-97	776	

No data for this customer yet!

Customer

C#	Name	Phone	…
321	Jones	9983-	
337	Sanchez	7738-	
938	Carson	8738-	

SQL Foreign Key (Oracle, SQL Server)

```
CREATE TABLE Order
 (   OID          NUMBER(9) NOT NULL,
     Odate        DATE,
     CID          NUMBER(9),
          CONSTRAINT pk_Order PRIMARY KEY (OID),
          CONSTRAINT fk_OrderCustomer
                FOREIGN KEY (CID)
                REFERENCES Customer (CID)
                ON DELETE CASCADE
)
```

DATABASE

Effect of Business Rules

Location Date Played					Referee Name Phone Number, Address				
Team 1 Name Sponsor		Score			Team 2 Name Sponsor		Score		
Player Name	Phone	Age	Points	Penalties	Player Name	Phone	Age	Points	Penalties

Key business rules:

- A player can play on only one team.
- There is one referee per match.

Business Rules 1

There is one referee per match.
A player can play on only one team.

Match(MatchID, DatePlayed, Location, RefID)
Score(MatchID, TeamID, Score)
Referee(RefID, Phone, Address)
Team(TeamID, Name, Sponsor)
Player(PlayerID, Name, Phone, DoB, TeamID)
PlayerStats(MatchID, PlayerID, Points, Penalties)

*RefID and TeamID are **not** keys in the Match and Team tables, because of the one-to-one rules.*

DATABASE

Business Rules 2

There can be several referees per match.
A player can play on only several teams (substitute), but only on one team per match.

Match(MatchID, DatePlayed, Location, RefID)
Score(MatchID, TeamID, Score)
Referee(RefID, Phone, Address)
Team(TeamID, Name, Sponsor)
Player(PlayerID, Name, Phone, DoB, TeamID)
PlayerStats(MatchID, PlayerID, Points, Penalties)

*To handle the many-to-many relationship, we need to make RefID and TeamID keys. But if you leave them in the same tables, the tables are **not** in 3NF. DatePlayed does not depend on RefID. Player Name does not depend on TeamID.*

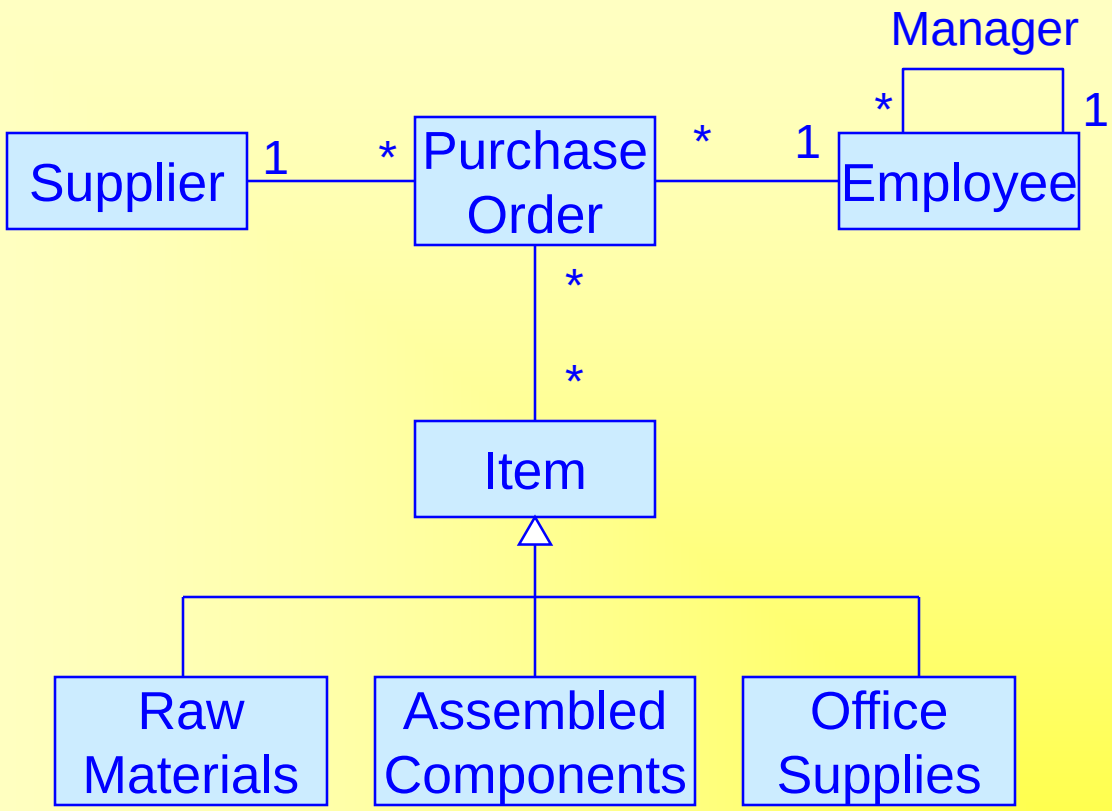
DATABASE

Business Rules 2: Normalized

There can be several referees per match.
A player can play on only several teams (substitute), but only on one team per match.

Match(MatchID, DatePlayed, Location)
RefereeMatch(MatchID, RefID)
Score(MatchID, TeamID, Score)
Referee(RefID, Phone, Address)
Team(TeamID, Name, Sponsor)
Player(PlayerID, Name, Phone, DoB)
PlayerStats(MatchID, PlayerID, TeamID, Points, Penalties)

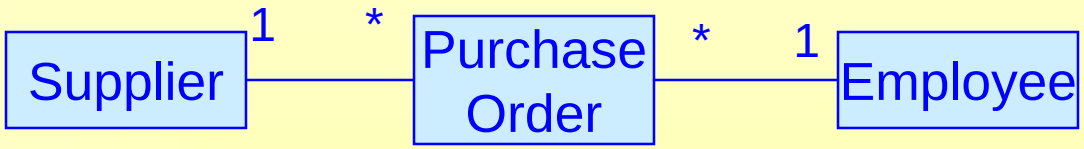
Converting a Class Diagram to Normalized Tables



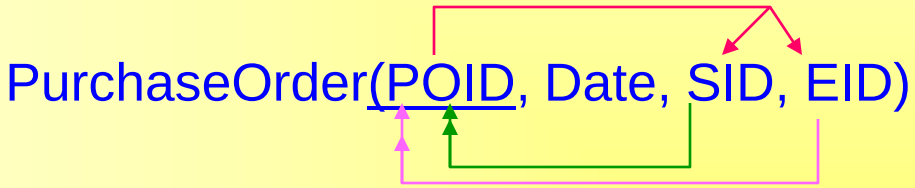
subtypes

DATABASE

One-to-Many Relationships



Supplier(SID, Name, Address, City, State, Zip, Phone)
Employee(EID, Name, Salary, Address, …)



The many side becomes a key (underlined).

Each PO has one supplier and employee.
(Do not key SID or EID)
Each supplier can receive many POs. *(Key PO)*
Each employee can place many POs. *(Key PO)*

DATABASE

One-to-Many Sample Data

Supplier

ID	Name	Address	City	State	Zip	Phone
5676	Jones	123 Elm	Ames	IA	50010	515-777-8988
6731	Markle	938 Oak	Boston	MA	02109	617-222-9999
7831	Paniche	873 Hickory	Jackson	MS	39205	601-333-9932
8872	Swensen	773 Poplar	Wichita	KS	67209	316-999-3312

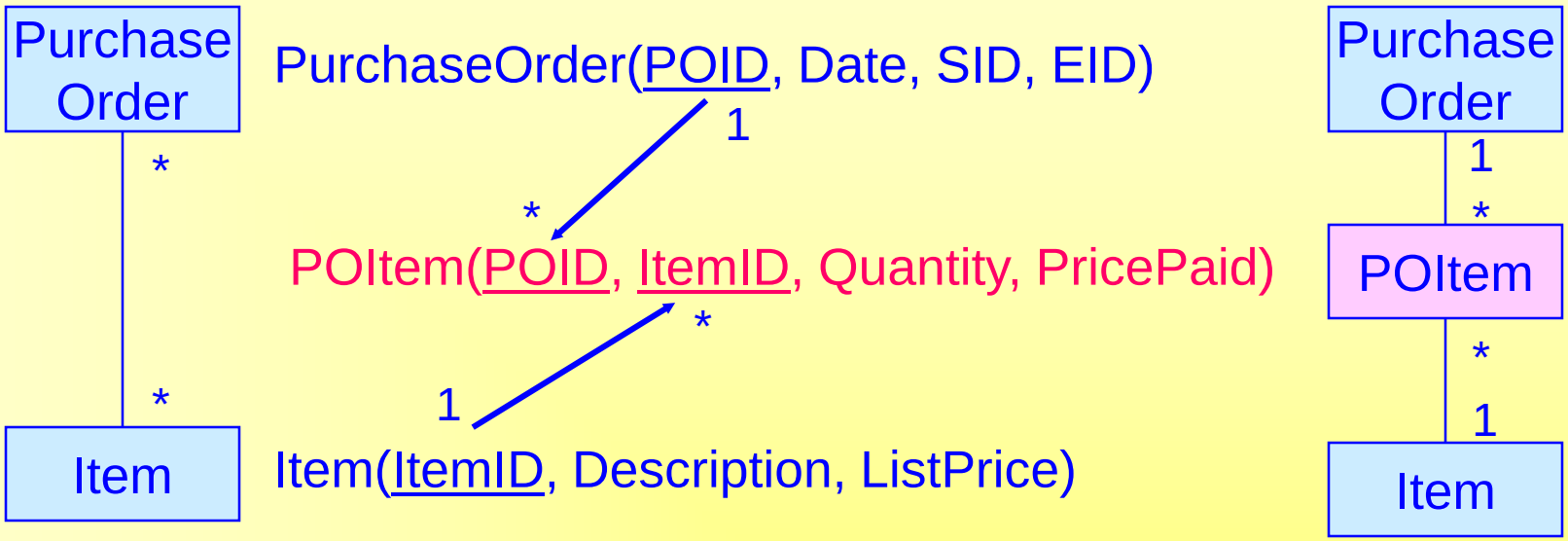
Purchase Order

POID	Date	SID	EID
22234	9-9-2004	5676	221
22235	9-10-2004	5676	554
22236	9-10-2004	7831	221
22237	9-11-2004	8872	335

Employee

EID	Name	Salary	Address
221	Smith	67,000	223 W. 2300
335	Sanchez	82,000	37 W. 7200
554	Johnson	35,000	440 E. 5200

Many-to-Many Relationships



Each POID can have many Items (key/underline ItemID).
Each ItemID can be on many POIDs (key POID).

Need the new intermediate table (POItem) because:
You cannot put ItemID into PurchaseOrder because Date, SID, and EID do not depend on the ItemID.
You cannot put POID into Item because Description and ListPrice do not depend on POID.

DATABASE

Many-to-Many Sample Data

Purchase Order

POID	Date	SID	EID
22234	9-9-2004	5676	221
22235	9-10-2004	5676	554
22236	9-10-2004	7831	221
22237	9-11-2004	8872	335

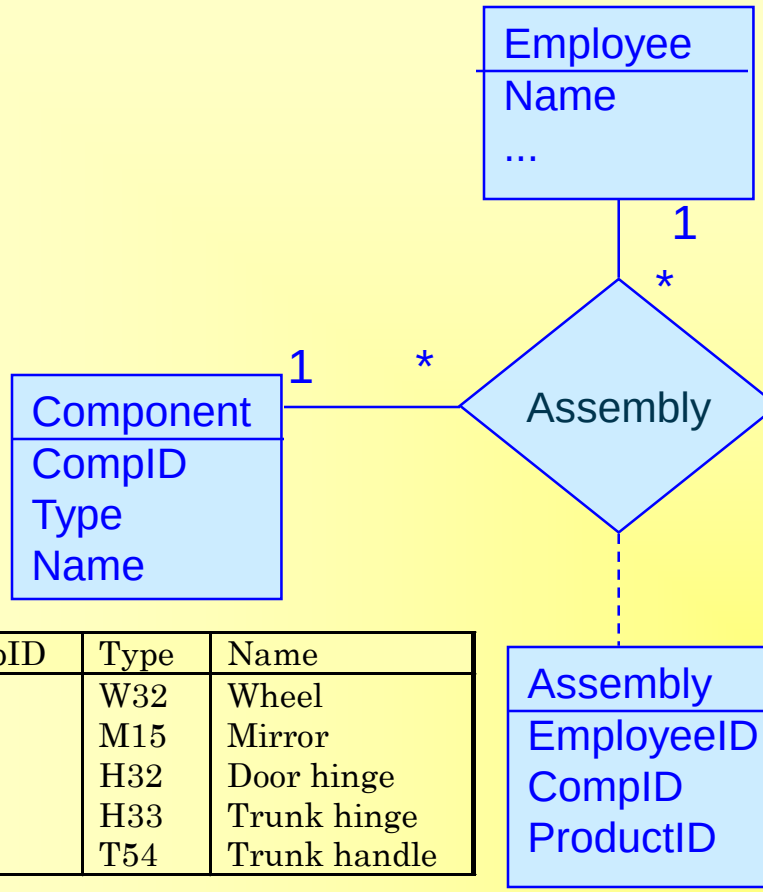
POItem

POID	ItemID	Quantity	PricePaid
22234	444098	3	2.00
22234	444185	1	25.00
22235	444185	4	24.00
22236	555828	10	150.00
22236	555982	1	5800.00

Item

ItemID	Description	ListPrice
444098	Staples	2.00
444185	Paper	28.00
555828	Wire	158.00
555982	Sheet steel	5928.00
888371	Brake assembly	152.00

N-ary Associations



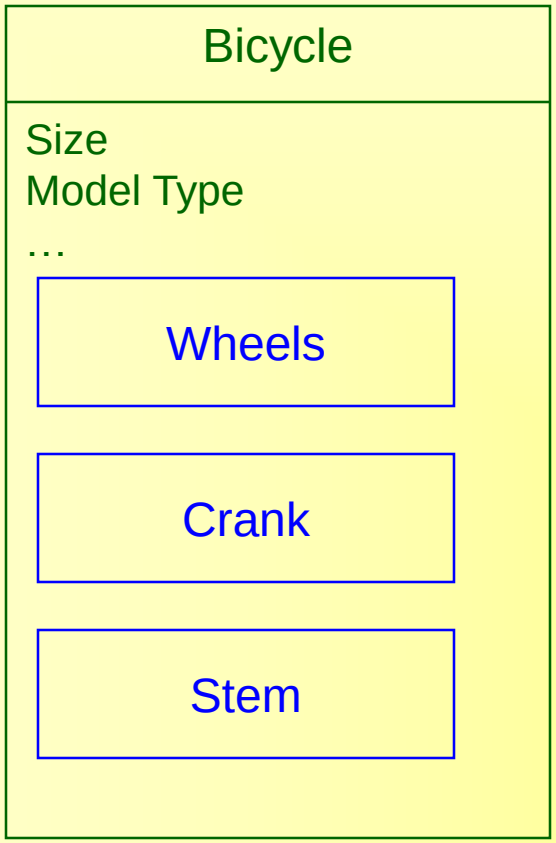
EmployeeID	Name	...
11	Joe Jones	...
12	Maria Rio	...

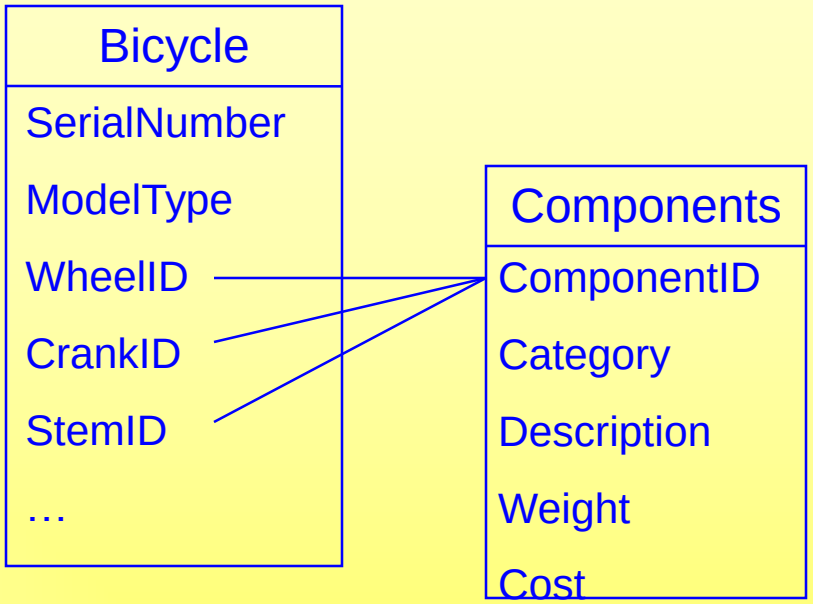
ProductID	Type	Name
A3222	X32	Corvette
A5411	B17	Camaro

CompID	Type	Name
563	W32	Wheel
872	M15	Mirror
882	H32	Door hinge
883	H33	Trunk hinge
888	T54	Trunk handle

EmployeeID	CompId	ProductID
11	563	A3222
11	872	A3222
11	563	A5411
11	872	A5411
12	563	A3222
12	882	A3222
12	888	A3222
12	883	A5411

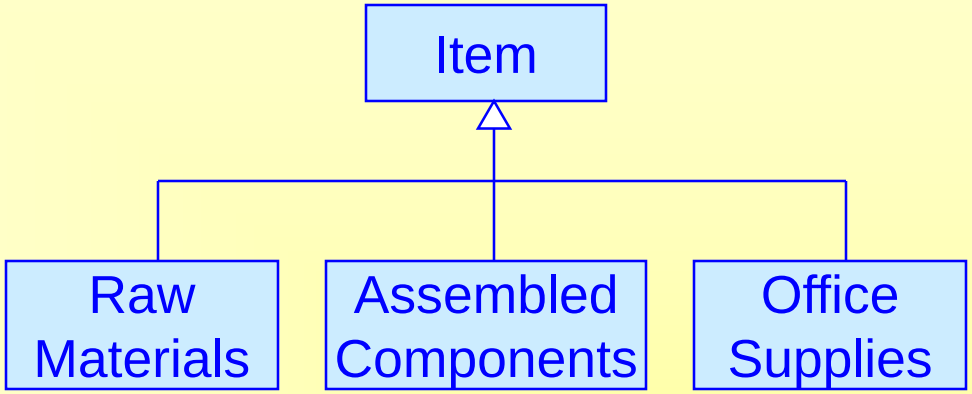
Composition





DATABASE

Generalization or Subtypes



Item(ItemID, Description, ListPrice)

RawMaterials(ItemID, Weight, StrengthRating)

AssembledComponents(ItemID, Width, Height, Depth)

OfficeSupplies(ItemID, BulkQuantity, Discount)

Add new tables for each subtype.
Use the same key as the generic type (ItemID)--one-to-one relationship.
Add the attributes specific to each subtype.

Subtypes Sample Data

Item

ItemID	Description	ListPrice
444098	Staples	2.00
444185	Paper	28.00
555828	Wire	158.00
555982	Sheet steel	5928.00
888371	Brake assembly	152.00

RawMaterials

ItemID	Weight	StrengthRating
555828	57	2000
555982	2578	8321

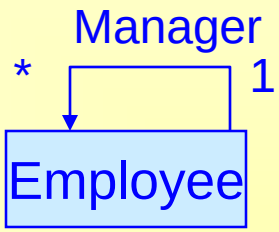
AssembledComponents

ItemID	Width	Height	Depth
888371	1	3	1.5

OfficeSupplies

ItemID	BulkQuantity	Discount
444098	20	10%
444185	10	15%

Recursive Relationships

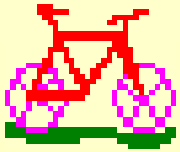


Employee(EID, Name, Salary, Address, Manager)

Employee

EID	Name	Salary	Address	Manager
221	Smith	67,000	223 W. 2300	335
335	Sanchez	82,000	37 W. 7200	
554	Johnson	35,000	440 E. 5200	335

Add a manager column that contains Employee IDs.
An employee can have only one manager. *(Manager is not a key.)*
A manager can supervise many employees. *(EID is a key.)*



Normalization Examples

- Possible topics
  - Auto repair
  - Auto sales
  - Department store
  - Hair stylist
  - HRM department
  - Law firm
  - Manufacturing
  - National Park Service
  - Personal stock portfolio
  - Pet shop
  - Restaurant
  - Social club
  - Sports team

DATABASE

Multiple Views & View Integration

- Collect multiple views
  - Documents
  - Reports
  - Input forms
- Create normalized tables from each view
- Combine the views into one complete model.
- Keep meta-data in a data dictionary
  - Type of data
  - Size
  - Volume
  - Usage
- Example
  - Federal Emergency Management Agency (FEMA). Disaster planning and relief.
  - Make business assumptions as necessary, but try to keep them simple.

DATABASE

The Pet Store: Sales Form

Sales

Sales# Date

Customer
Name
Address
City, State, Zip

Employee ID
Name

Animal Sale

ID	Name	Category	Breed	DoB	Gender	Reg.	Color	ListPrice	SalePrice
								Animal SubTotal	

Merchandise Sale

Item	Description	Category	ListPrice	SalePrice	Quantity	Extended
				Merchandise SubTotal		
				Subtotal		
				Tax		
				Total		

Sales(SaleID, Date, CustomerID, Name, Address, City, State, Zip, EmployeeID, Name, (AnimalID, Name, Category, Breed, DateOfBirth, Gender, Registration, Color, ListPrice, SalePrice), (ItemID, Description, Category, ListPrice, SalePrice, Quantity))

DATABASE

The Pet Store: Purchase Animals

Purchase Order for Animals

Order# Date Ordered
Date Received

Supplier	Employee ID
Name	Name
Contact	Home Phone
Phone	Date Hired
Address	
City, State, ZIPcode	

Animal Descriptions

Name	Category	Breed	Gender	Registration	Price

Subtotal
Shipping Cost
Total

AnimalOrder(OrderID, OrderDate, ReceiveDate, SupplierID, Name, Contact, Phone, Address, City, State, Zip, EmployeeID, Name, Phone, DateHired, (AnimalID, Name, Category, Breed, Gender, Registration, Cost), ShippingCost)

DATABASE

The Pet Store: Purchase Merchandise

Purchase Order for Merchandise	
Order#	Date Ordered Date Received
Supplier Name Contact Phone Address City, State, ZIPcode	Employee ID Name Home Phone

Items Ordered

ItemID	Description	Category	Price	Quantity	Ext.	QOH
				Subtotal		
				Shipping Cost		
				Total		

MerchandiseOrder(PONumber, OrderDate, ReceiveDate, SupplierID, Name, Contact, Phone, Address, City, State, Zip, EmployeeID, Name, HomePhone, (ItemID, Description, Category, Price, Quantity, QuantityOnHand), ShippingCost)

DATABASE

Pet Store Normalization

Sale(SaleID, Date, CustomerID, EmployeeID)
SaleAnimal(SaleID, AnimalID, SalePrice)
SaleItem(SaleID, ItemID, SalePrice, Quantity)
Customer(CustomerID, Name, Address, City, State, Zip)
Employee(EmployeeID, Name)
Animal(AnimalID, Name, Category, Breed, DateOfBirth,
Gender, Registration, Color, ListPrice)
Merchandise(ItemID, Description, Category, ListPrice)

AnimalOrder(OrderID, OrderDate, ReceiveDate, SupplierID, EmpID, ShipCost)
AnimalOrderItem(OrderID, AnimalID, Cost)
Supplier(SupplierID, Name, Contact, Phone, Address, City, State, Zip)
Employee(EmployeeID, Name, Phone, DateHired)
Animal(AnimalID, Name, Category, Breed, Gender, Registration, Cost)

MerchandiseOrder(PONumber, OrderDate, ReceiveDate, SID, EmpID, ShipCost)
MerchandiseOrderItem(PONumber, ItemID, Quantity, Cost)
Supplier(SupplierID, Name, Contact, Phone, Address, City, State, Zip)
Employee(EmployeeID, Name, Phone)
Merchandise(ItemID, Description, Category, QuantityOnHand)

DATABASE

Pet Store View Integration

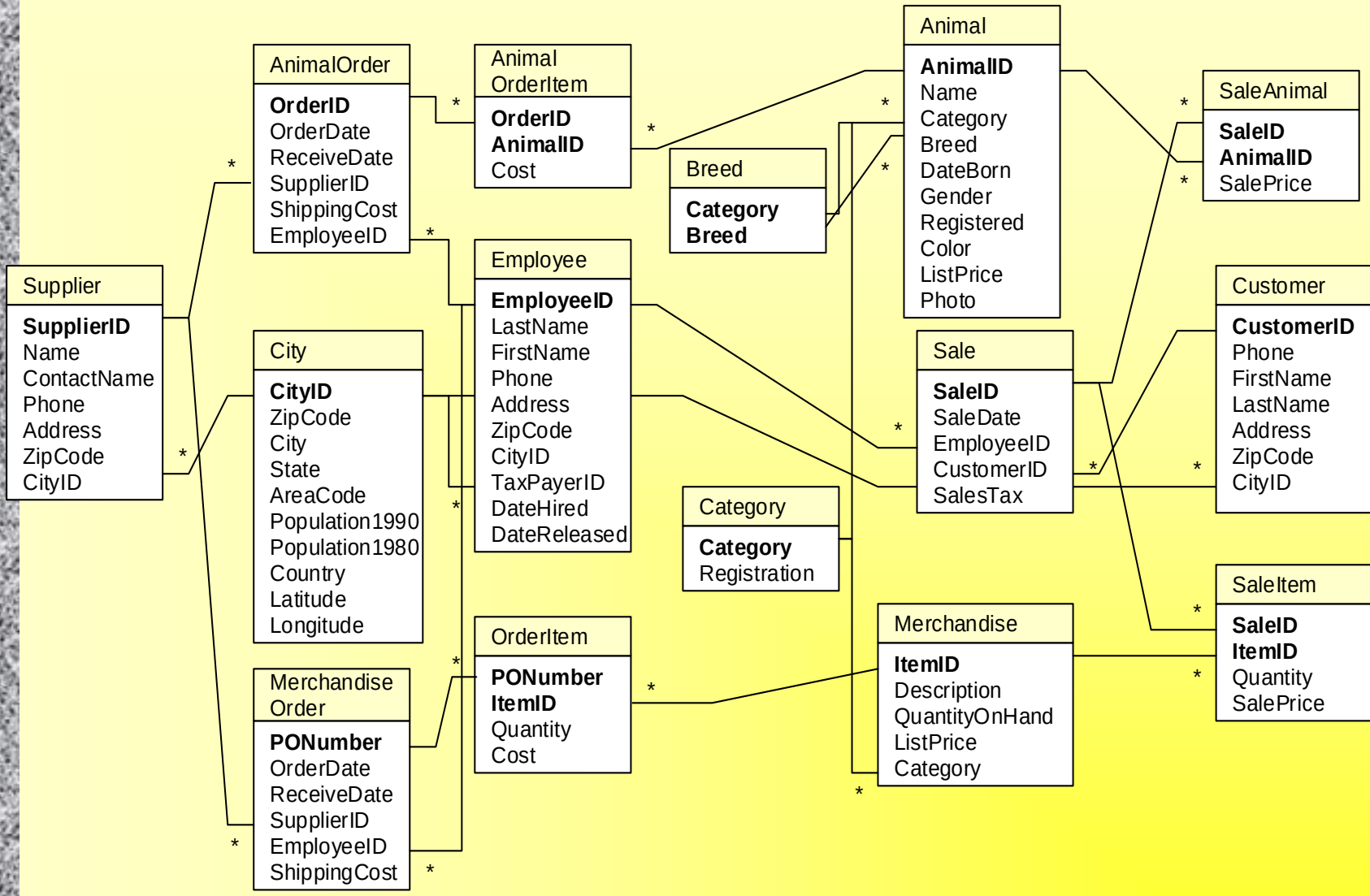
Sale(SaleID, Date, CustomerID, EmployeeID)
SaleAnimal(SaleID, AnimalID, SalePrice)
SaleItem(SaleID, ItemID, SalePrice, Quantity)
Customer(CustomerID, Name, Address, City, State, Zip)
Employee(EmployeeID, Name, Phone, DateHired)
Animal(AnimalID, Name, Category, Breed, DateOfBirth,
Gender, Registration, Color, ListPrice, Cost)
Merchandise(ItemID, Description, Category, ListPrice, QuantityOnHand)

AnimalOrder(OrderID, OrderDate, ReceiveDate, SupplierID, EmpID, ShipCost)
AnimalOrderItem(OrderID, AnimalID, Cost)
Supplier(SupplierID, Name, Contact, Phone, Address, City, State, Zip)
~~Employee(EmployeeID, Name, Phone, DateHired)~~
~~Animal(AnimalID, Name, Category, Breed, Gender, Registration, Cost)~~

MerchandiseOrder(PONumber, OrderDate, ReceiveDate, SID, EmpID, ShipCost)
MerchandiseOrderItem(PONumber, ItemID, Quantity, Cost)
~~Supplier(SupplierID, Name, Contact, Phone, Address, City, State, Zip)~~
Employee(EmployeeID, Name, Phone)
~~Merchandise(ItemID, Description, Category, QuantityOnHand)~~

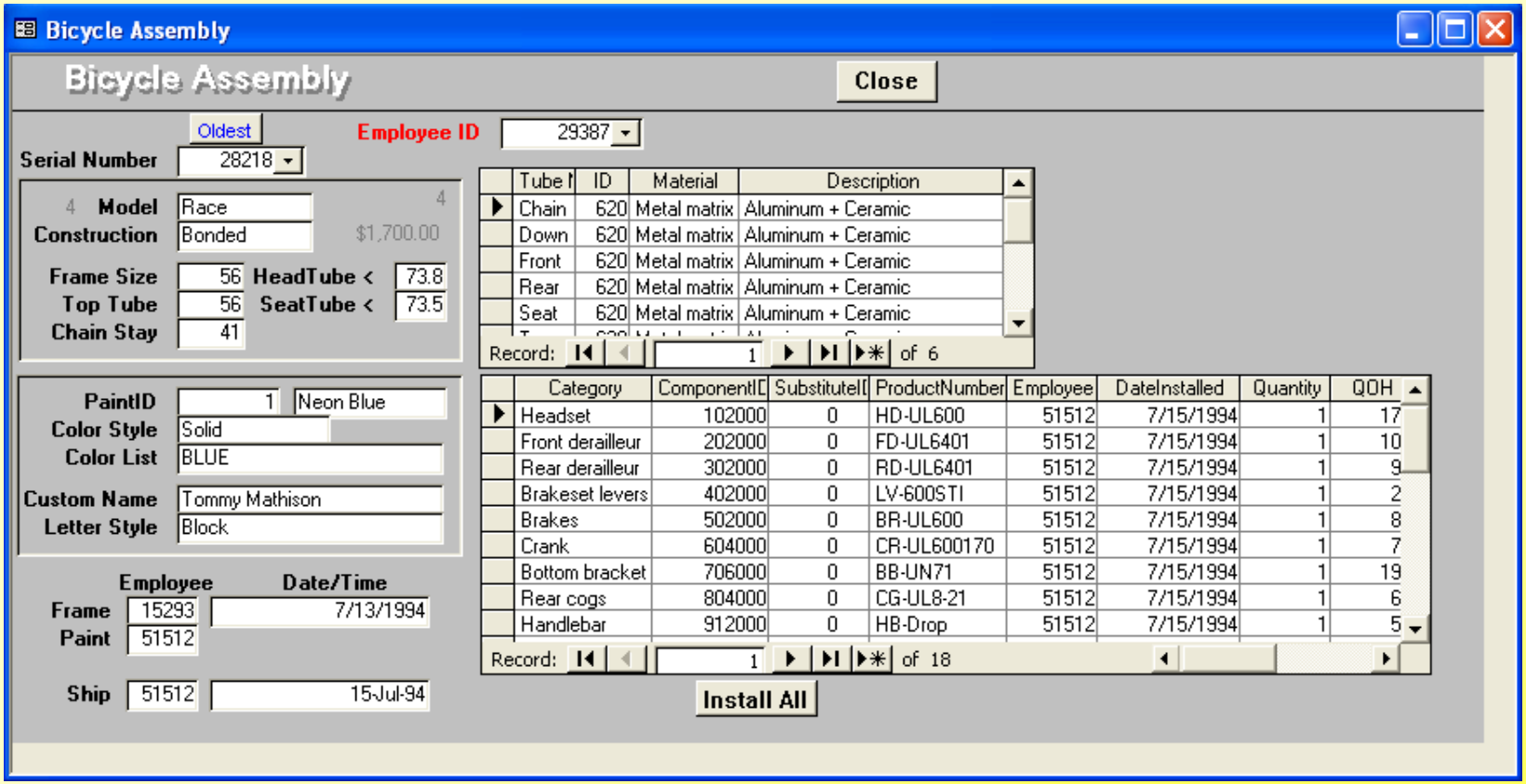
DATABASE

Pet Store Class Diagram



DATABASE

Rolling Thunder Integration Example



Bicycle Assembly form. The main EmployeeID control is not stored directly, but the value is entered in the assembly column when the employee clicks the column.

DATABASE

Initial Tables for Bicycle Assembly

BicycleAssembly(
SerialNumber, Model, Construction, FrameSize, TopTube, ChainStay, HeadTube, SeatTube,
PaintID, PaintColor, ColorStyle, ColorList, CustomName, LetterStyle, EmpFrame, EmpPaint,
BuildDate, ShipDate,
(Tube, TubeType, TubeMaterial, TubeDescription),
(CompCategory, ComponentID, SubstID, ProdNumber, EmpInstall, DateInstall, Quantity, QOH))

Bicycle(SerialNumber, Model, Construction, FrameSize, TopTube, ChainStay, HeadTube, SeatTube,
PaintID, ColorStyle, CustomName, LetterStyle, EmpFrame, EmpPaint, BuildDate, ShipDate)

Paint(PaintID, ColorList)

BikeTubes(SerialNumber, TubeID, Quantity)

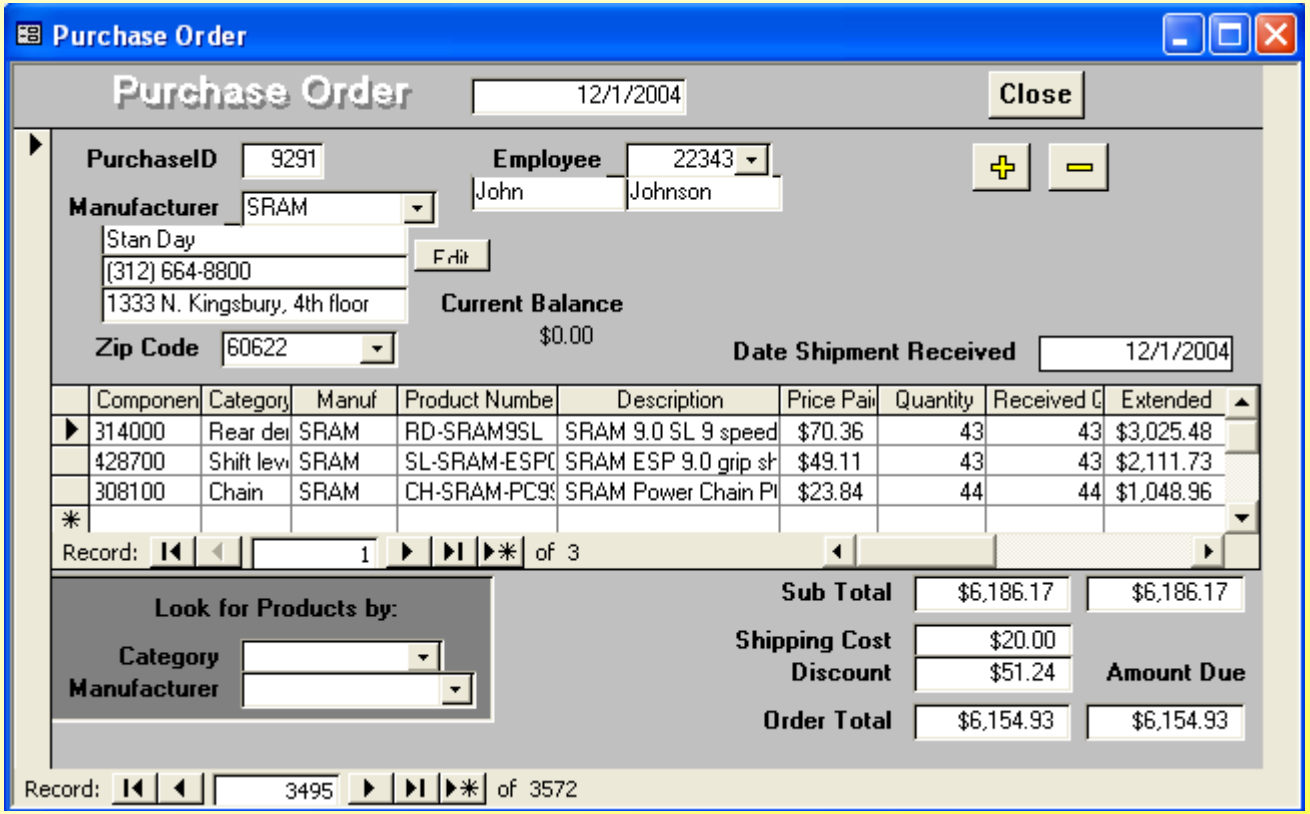
TubeMaterial(TubeID, Type, Material, Description)

BikeParts(SerialNumber, ComponentID, SubstID, Quantity, DateInstalled, EmpInstalled)

Component(ComponentID, ProdNumber, Category, QOH)

DATABASE

Rolling Thunder: Purchase Order



DATABASE

RT Purchase Order: Initial Tables

PurchaseOrder(PurchaseID, PODate, EmployeeID, FirstName, LastName, ManufacturerID, MfgName, Address, Phone, CityID, CurrentBalance, ShipReceiveDate, (ComponentID, Category, ManufacturerID, ProductNumber, Description, PricePaid, Quantity, ReceiveQuantity, ExtendedValue, QOH, ExtendedReceived), ShippingCost, Discount)

PurchaseOrder(PurchaseID, PODate, EmployeeID, ManufacturerID, ShipReceiveDate, ShippingCost, Discount)

Employee(EmployeeID, FirstName, LastName)

Manufacturer(ManufacturerID, Name, Address, Phone, Address, CityID, CurrentBalance)

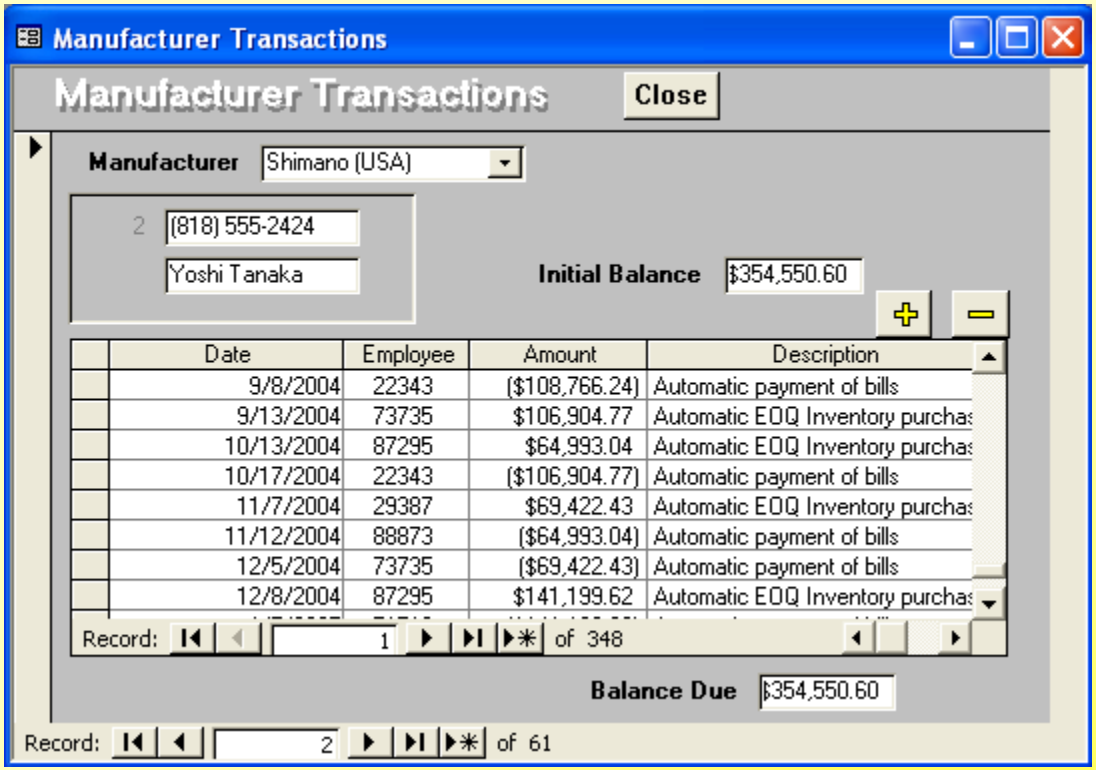
City(CityID, Name, ZipCode)

PurchaseItem(PurchaseID, ComponentID, Quantity, PricePaid, ReceivedQuantity)

Component(ComponentID, Category, ManufacturerID, ProductNumber, Description, QOH)

DATABASE

Rolling Thunder: Transactions



DATABASE

RT Transactions: Initial Tables

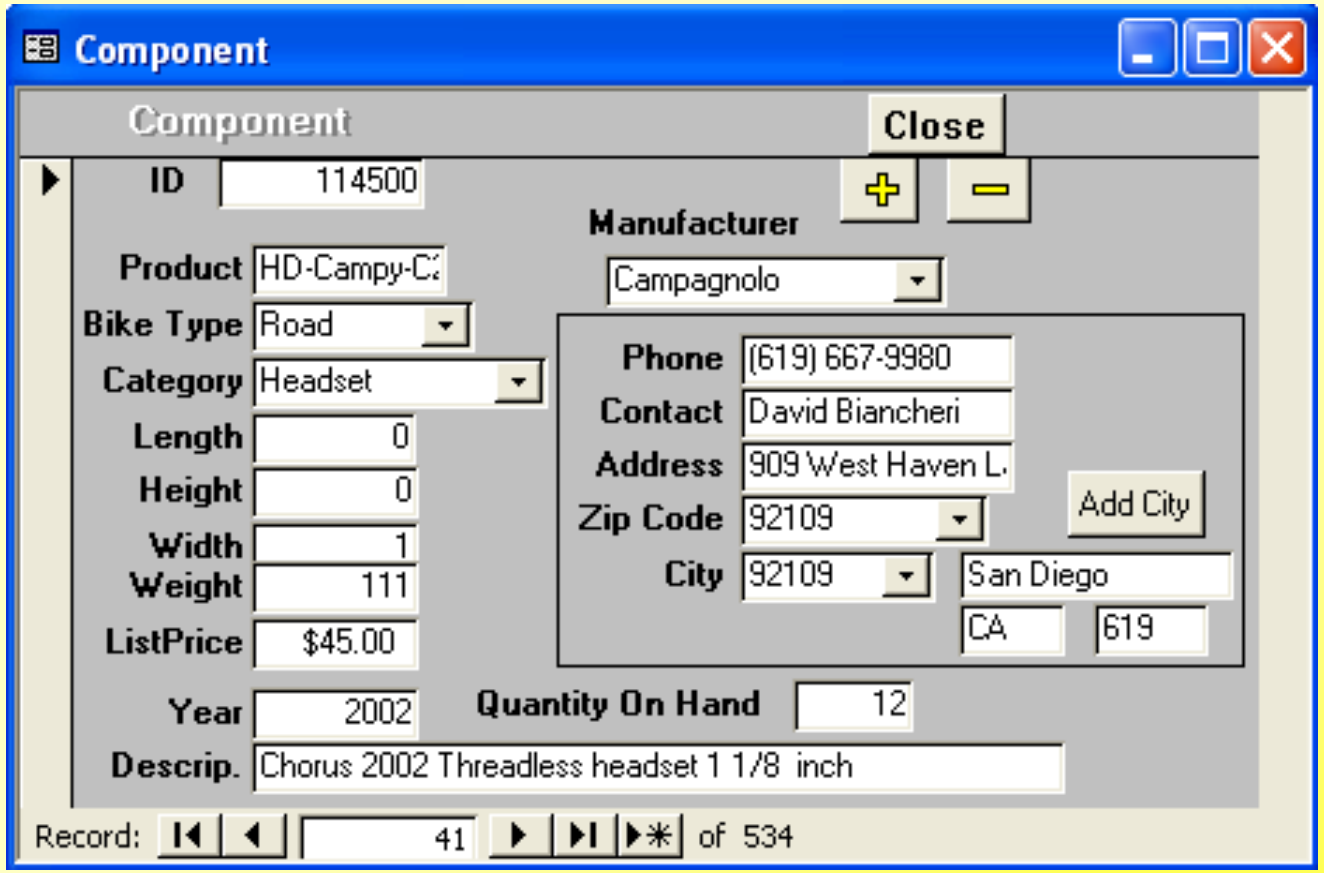
ManufacturerTransactions(ManufacturerID, Name, Phone, Contact, BalanceDue,
(TransDate, Employee, Amount, Description))

Manufacturer(ManufacturerID, Name, Phone, Contact, BalanceDue)

ManufacturerTransaction(ManufacturerID, TransactionDate, EmployeeID,
Amount, Description)

DATABASE

Rolling Thunder: Components



DATABASE

RT Components: Initial Tables

ComponentForm(ComponentID, Product, BikeType, Category, Length, Height, Width, Weight, ListPrice,Description, QOH, ManufacturerID, Name, Phone, Contact, Address, ZipCode, CityID, City, State, AreaCode)

Component(ComponentID, ProductNumber, BikeType, Category, Length, Height, Width,Weight, ListPrice, Description, QOH, ManufacturerID)

Manufacturer(ManufacturerID, Name, Phone, Contact, Address, ZipCode, CityID)

City(CityID, City, State, ZipCode, AreaCode)

DATABASE

RT: Integrating Tables

Duplicate Manufacturer tables:

PO Mfr(ManufacturerID, Name, Address, Phone, CityID, CurrentBalance)
Mfg Mfr(ManufacturerID, Name, Phone, **Contact**, BalanceDue)
Comp Mfr(ManufacturerID, Name, Phone, Contact, Address, **ZipCode**, CityID)

Note that each form can lead to duplicate tables.
Look for tables with the same keys, but do not expect them to be named exactly alike.
Find all of the data and combine it into one table.

Manufacturer(ManufacturerID, Name, Contact, Address, Phone,
Address, CityID, |ZipCode, CurrentBalance)

DATABASE

RT Example: Integrated Tables

Bicycle(SerialNumber, Model, Construction, FrameSize, TopTube, ChainStay, HeadTube, SeatTube, PaintID, ColorStyle, CustomName, LetterStyle, EmpFrame, EmpPaint, BuildDate, ShipDate)

Paint(PaintID, ColorList)

BikeTubes(SerialNumber, TubeID, Quantity)

TubeMaterial(TubeID, Type, Material, Description)

BikeParts(SerialNumber, ComponentID, SubstID, Quantity, DateInstalled, EmpInstalled)

Component(ComponentID, ProductNumber, BikeType, Category, Length, Height, Width, Weight, ListPrice, Description, QOH, ManufacturerID)

PurchaseOrder(PurchaseID, PODate, EmployeeID, ManufacturerID, ShipReceiveDate, ShippingCost, Discount)

PurchaseItem(PurchaseID, ComponentID, Quantity, PricePaid, ReceivedQuantity)

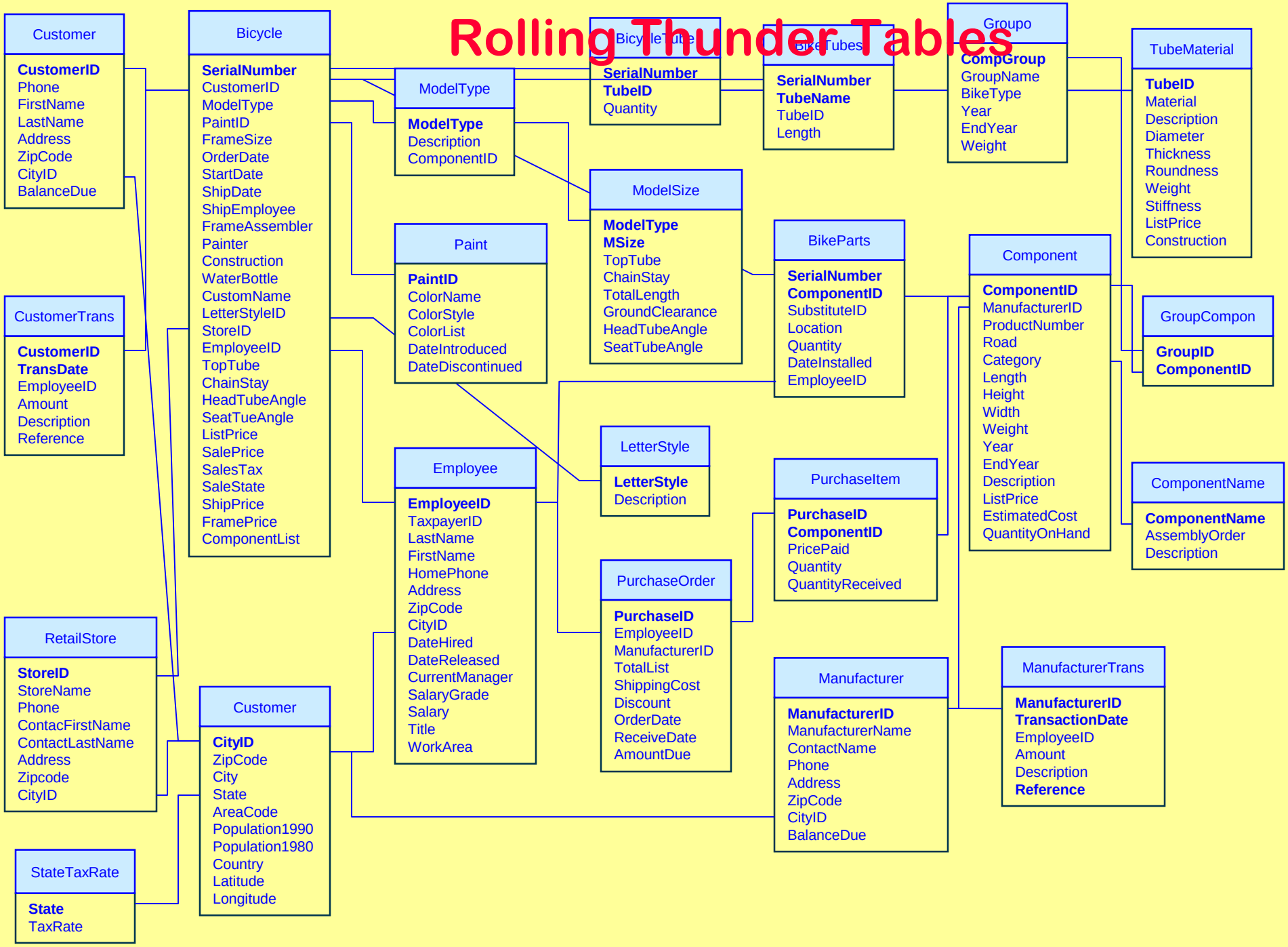
Employee(EmployeeID, FirstName, LastName)

Manufacturer(ManufacturerID, Name, Contact, Address, Phone, CityID, ZipCode, CurrentBalance)

ManufacturerTransaction(ManufacturerID, TransactionDate, EmployeeID, Amount, Description, Reference)

City(CityID, City, State, ZipCode, AreaCode)

Rolling Thunder Tables



DATABASE

View Integration (FEMA Example 1)

Team Roster

Team# Date Formed
Home Base
Response time (days)

Leader
Name Fax Phone
Address, C,S,Z Home phone

Team Members/Crew

ID	Name	Home phone	Specialty	DoB	SSN	Salary

Total Salary

- This first form is kept for each team that can be called on to help in emergencies.

DATABASE

View Integration (FEMA Example 2)

Disaster Name
Local Agency
Political Contact

HQ Location
Commander

On-Site Problem Report

Date Reported
Problem Description

Assigned Problem#

Severity

Reported By: Specialty Specialty Rating
Verified By: Specialty Specialty Rating

SubProblem Details

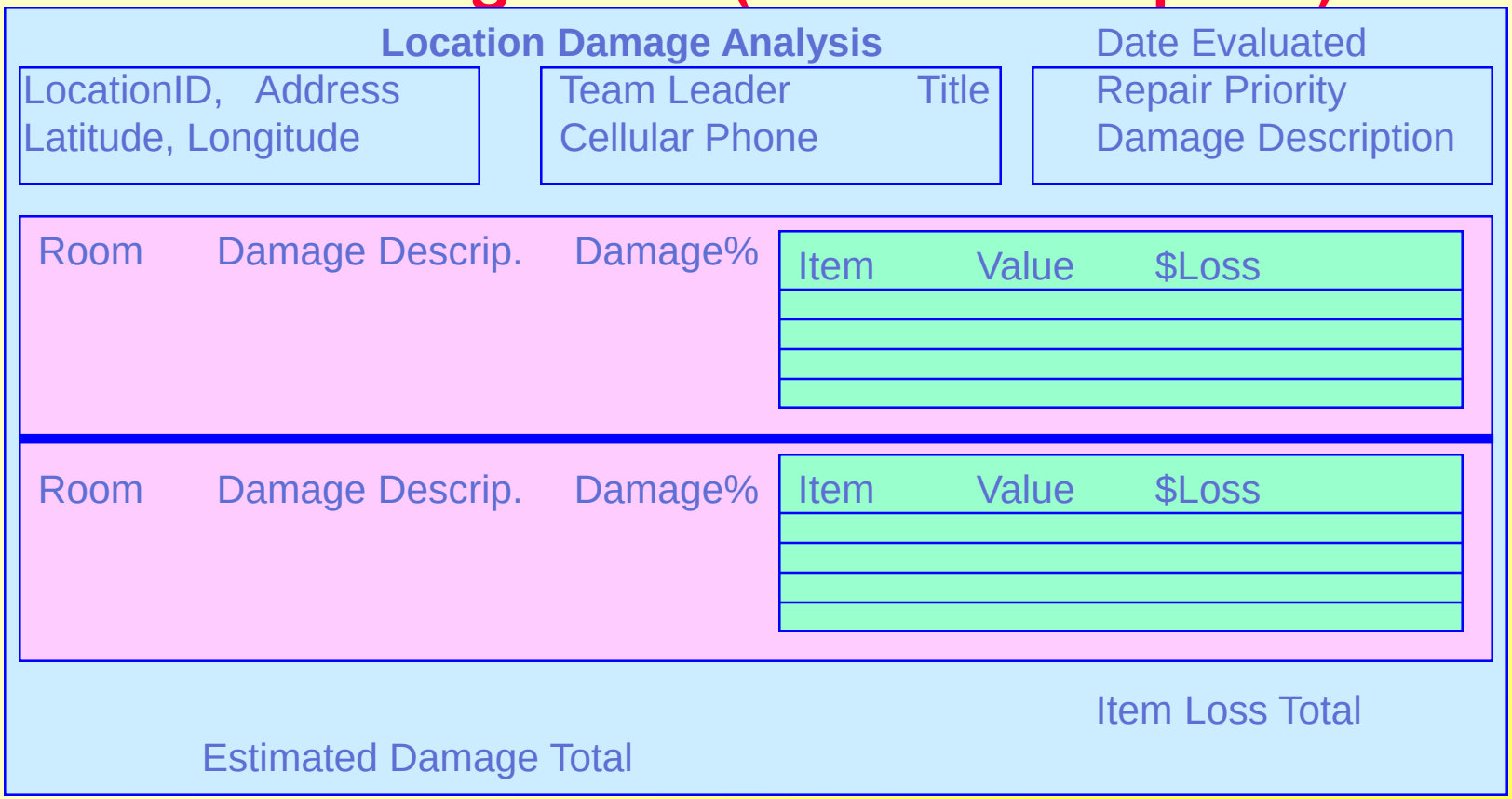
Sub Prob#	Category	Description	Action	Est. Cost

Total Est. Cost

- Major problems are reported to HQ to be prioritized and scheduled for correction.

DATABASE

View Integration (FEMA Example 3)



- On-site teams examine buildings and file a report on damage at that location.

View Integration (FEMA Example 3a)

- Location Analysis(LocationID, MapLatitude, MapLongitude, Date, Address, Damage, PriorityRepair, Leader, LeaderPhone, LeaderTitle, (Room, Description, PercentDamage, (Item, Value, Loss)))

DATABASE

View Integration (FEMA Example 4)

Task Completion Report

Disaster Name Disaster Rating HQ Phone Date

Problem# Supervisor Date

SubProblem	Team#	Team Specialty	CompletionStatus	Comment	Expenses

Total Expenses

Problem# Supervisor Date

SubProblem	Team#	Team Specialty	CompletionStatus	Comment	Expenses

Total Expenses

- Teams file task completion reports. If a task is not completed, the percentage accomplished is reported as the completion status.

View Integration (FEMA Example 4a)

- TasksCompleted(Date, DisasterName, DisasterRating, HQPhone, (Problem#, Supervisor, (SubProblem, Team#, CompletionStatus, Comments, Expenses))

DATABASE

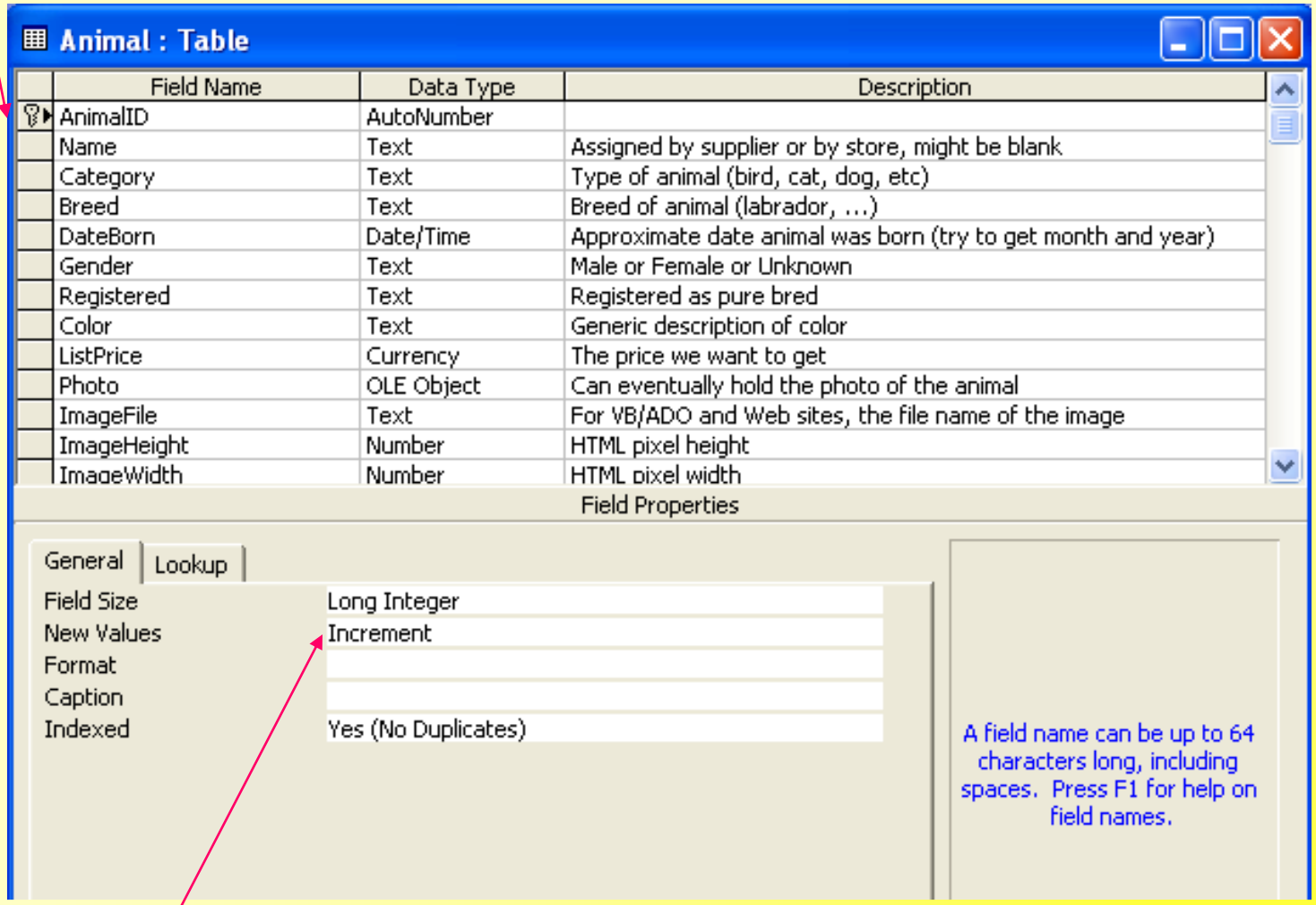
DBMS Table Definition

- Enter Tables
  - Columns
  - Keys
  - Data Types
    - Text
    - Memo
    - Number
      - Byte
      - Integer, Long
      - Single, Double
    - Date/Time
    - Currency
    - AutoNumber (Long)
    - Yes/No
    - OLE Object
  - Descriptions
- Column Properties
  - Format
  - Input Mask
  - Caption
  - Default
  - Validation Rule
  - Validation Text
  - Required & Zero Length
  - Indexed
- Relationships
  - One-to-One
  - One-to-Many
  - Referential Integrity
  - Cascade Update/Delete
  - Define **before** entering data

DATABASE

Key

Table Definition in Access

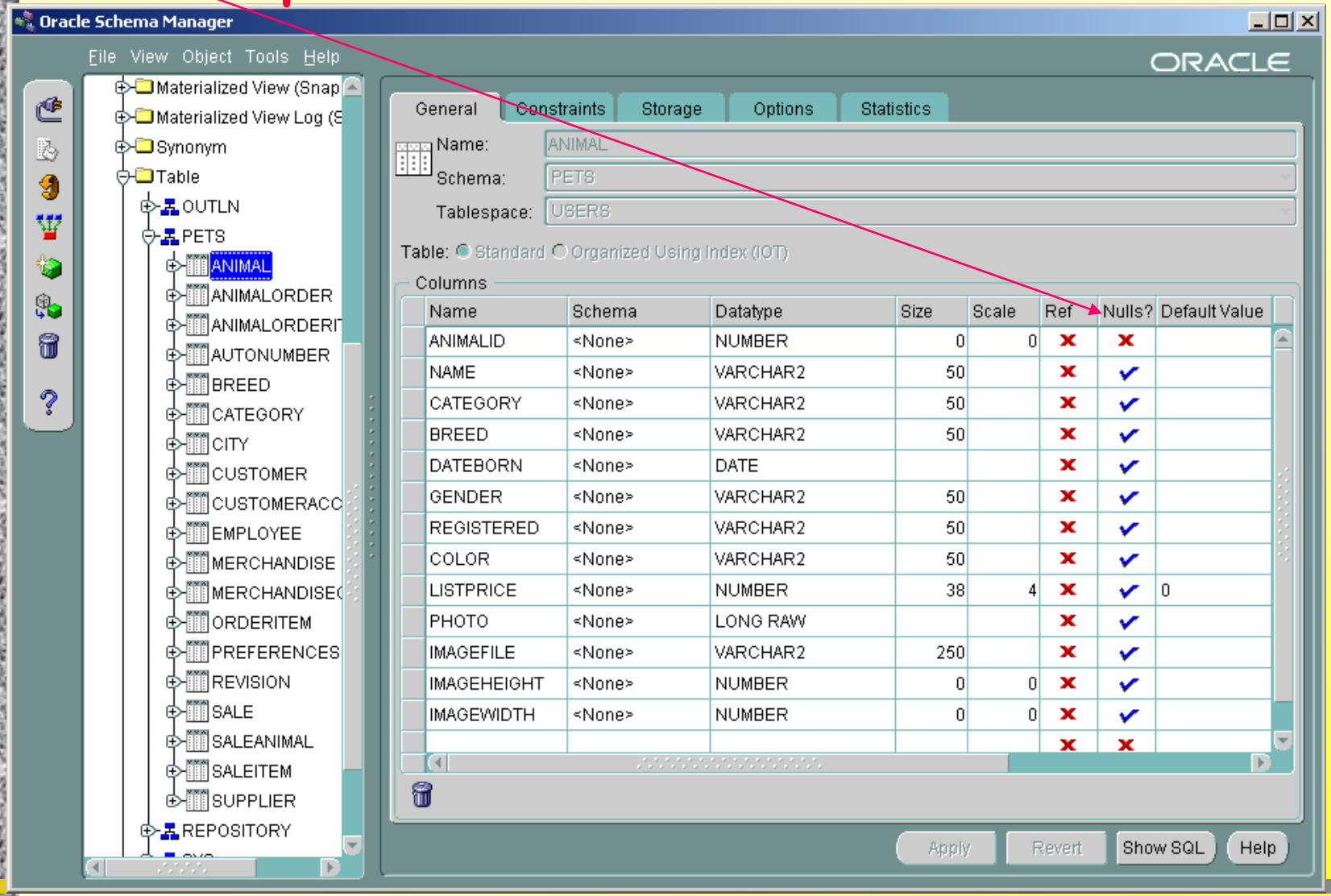


Numeric Subtypes or text length

DATABASE

Key

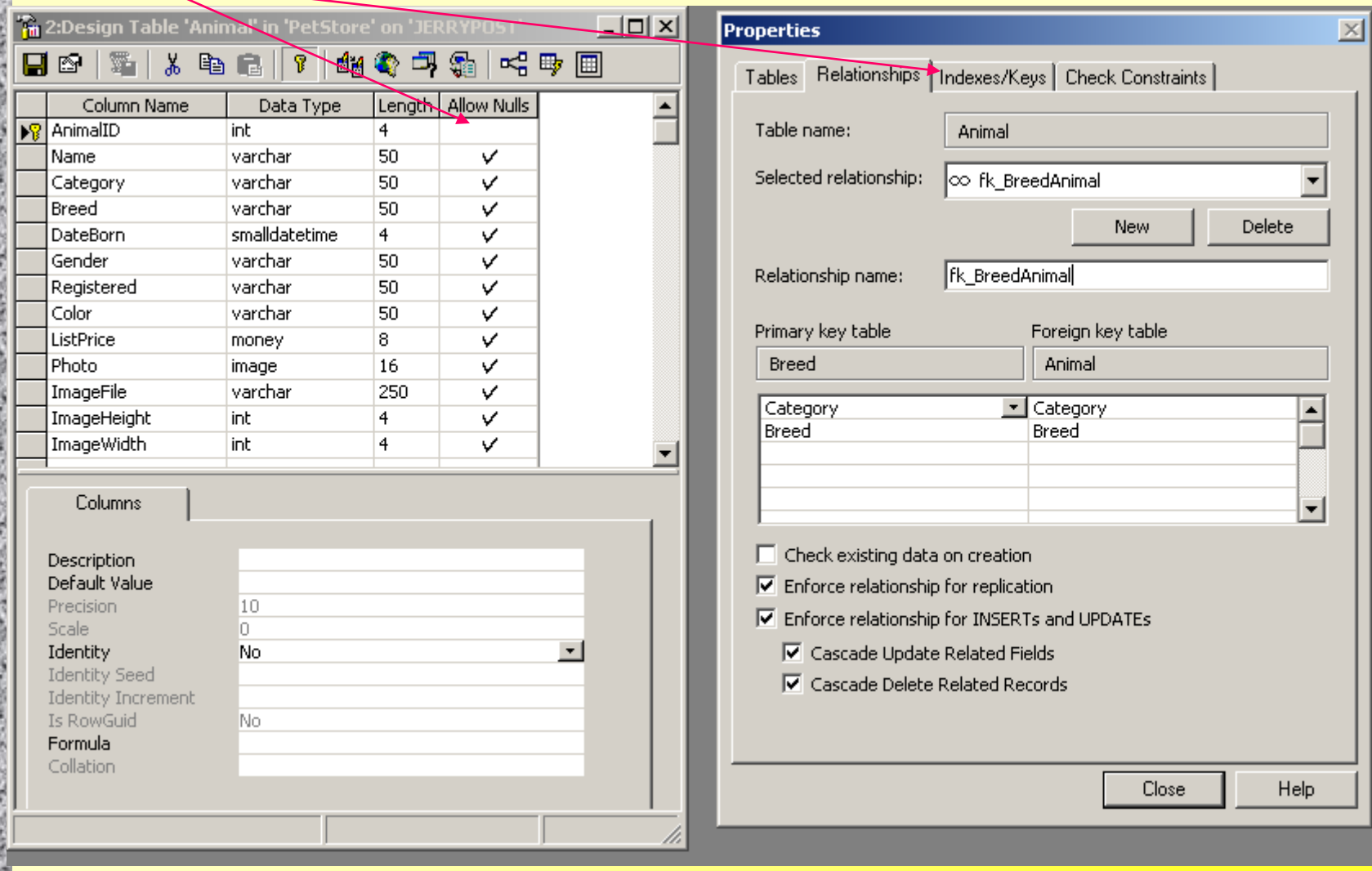
Graphical Table Definition in Oracle



Minimal ability to change the table definition later.

Graphical Table Definition in SQL Server

Keys



DATABASE

SQL Table Definition

```
CREATE TABLE Animal
(
AnimalID        INTEGER,
Name            VARCHAR2(50),
Category        VARCHAR2(50),
Breed           VARCHAR2(50),
DateBorn        DATE,
Gender          VARCHAR2(50)    CHECK (Gender='Male' Or
       Gender='Female' Or Gender='Unknown' Or Gender Is Null),
Registered      VARCHAR2(50),
Color           VARCHAR2(50),
ListPrice       NUMBER(38,4)        DEFAULT 0,
Photo           LONG RAW,
ImageFile       VARCHAR2(250),
ImageHeight     INTEGER,
ImageWidth      INTEGER,
     CONSTRAINT pk_Animal PRIMARY KEY (AnimalID),
     CONSTRAINT fk_BreedAnimal FOREIGN KEY (Category,Breed)
        REFERENCES Breed(Category,Breed)
        ON DELETE CASCADE,
     CONSTRAINT fk_CategoryAnimal FOREIGN KEY (Category)
        REFERENCES Category(Category)
        ON DELETE CASCADE
);
```

Oracle Databases

- For Oracle and SQL Server, it is best to create a text file that contains all of the SQL statements to create the table.
  - It is usually easier to modify the text table definition.
  - The text file can be used to recreate the tables for backup or transfer to another system.
  - To make major modifications to the tables, you usually create a new table, then copy the data from the old table, then delete the old table and rename the new one. It is much easier to create the new table using the text file definition.
  - Be sure to specify Primary Key and Foreign Key constraints.
  - Be sure to create tables in the correct order—any table that appears in a Foreign Key constraint must first be created. For example, create Customer before creating Order.
  - In Oracle, to substantially improve performance, issue the following command once all tables have been created:
    - `Analyze table Animal compute statistics;`

Data Volume

- Estimate the total size of the database.
  - Current.
  - Future growth.
  - Guide for hardware and software purchases.
- For each table.
  - Use data types to estimate the number of bytes used for each row.
  - Multiply by the estimated number of rows.
- Add the value for each table to get the total size.
- For concatenated keys (and similar tables).
  - OrderItems(O#, Item#, Qty)
  - Hard to “know” the total number of items ordered.
    - Start with the total number of orders.
    - Multiply by the average number of items on a typical order.
- Need to know time frame or how long to keep data.
  - Do we store all customer data forever?
  - Do we keep all orders in the active database, or do we migrate older ones?

DATABASE

Data Volume Example

Customer(C#, Name, Address, City, State, Zip)
Row: 4 + 15 + 25 + 20 + 2 + 10 = 76

Order(O#, C#, Odate)
Row: 4 + 4 + 8 = 16

OrderItem(O#, P#, Quantity, SalePrice)
Row: 4 + 4 + 4 + 8 = 20

$$Orders\,in\,3\,yrs = 1000\,Customers * \frac{10\,Orders}{Customer} * 3\,yrs = 30{,}000$$

$$OrderLines = 30{,}000\,Orders * \frac{5\,Lines}{Order} = 150{,}000$$

- Business rules
  - Three year retention.
  - 1000 customers.
  - Average 10 orders per customer per year.
  - Average 5 items per order.

Customer	76 * 1000	76,000
Order	16 * 30,000	480,000
OrderItem	20 * 150,000	3,000,000
Total		**3,556,000**

DATABASE

Appendix: Formal Definitions: Terms

Formal	Definition	Informal
Relation	A set of attributes with data that changes over time. Often denoted R.	Table
Attribute	Characteristic with a real-world domain. Subsets of attributes are multiple columns, often denoted X or Y.	Column
Tuple	The data values returned for specific attribute sets are often denoted as t[X]	Row of data
Schema	Collection of tables and constraints/relationships	
Functional dependency	X → Y	Business rule dependency

Appendix: Functional Dependency

Derives from a real-world relationship/constraint.

Denoted X → Y for sets of attributes X and Y

Holds when any rows of data that have identical values for X attributes also have identical values for their Y attributes:

If t1[X] = t2[X], then t1[Y] = t2[Y]

X is also known as a **determinant** if X is non-trivial (not a subset of Y).

DATABASE

Appendix: Keys

Keys are attributes that are ultimately used to identify rows of data.

A **key** K (sometimes called candidate key) is a set of attributes

(1) With FD K → U where U is all other attributes in the relation

(2) If K' is a subset of K, then there is no FD K' → U

A set of key attributes functionally determines all other attributes in the relation, and it is the smallest set of attributes that will do so (there is no smaller subset of K that determines the columns.)

DATABASE

Appendix: First Normal Form

A relation is in **first normal form (1NF)** if and only if all attributes are atomic.

Atomic attributes are single valued, and cannot be composite, multi-valued or nested relations.

Example:

Customer(CID, Name: First + Last, Phones, Address)

CID	Name: First + Last	Phones	Address
111	Joe Jones	111-2223 111-3393 112-4582	123 Main

DATABASE

Appendix: Second Normal Form

A relation is in **second normal form (2NF)** if it is in 1NF and each non-key attribute is fully functionally dependent on the primary key.

K → Ai for each non-key attribute Ai

That is, there is no subset K' such that K' → Ai

Example:

OrderProduct(OrderID, ProductID, Quantity, Description)

OrderID	ProductID	Quantity	Description
32	15	1	Blue Hose
32	16	2	Pliers
33	15	1	Blue Hose

Appendix: Transitive Dependency

Given functional dependencies: X → Y and Y → Z, the **transitive dependency** X → Z must also hold.

Example:

There is an FD between OrderID and CustomerID. Given the OrderID key attribute, you always know the CustomerID.

There is an FD between CustomerID and the other customer data, because CustomerID is the primary key. Given the CustomerID, you always know the corresponding attributes for Name, Phone, and so on.

Consequently, given the OrderID (X), you always know the corresponding customer data by transitivity.

DATABASE

Appendix: Third Normal Form

A relation is in **third normal form** if and only if it is in 2NF and no non-key attributes are transitively dependent on the primary key.

That is, K → Ai for each attribute, (2NF) and

There is no subset of attributes X such that K → X → Ai

Example:

Order(OrderID, OrderDate, CustomerID, Name, Phone)

OrderID	OrderDate	CustomerID	Name	Phone
32	5/5/2004	1	Jones	222-3333
33	5/5/2004	2	Hong	444-8888
34	5/6/2004	1	Jones	222-3333

DATABASE

Appendix: Boyce-Codd Normal Form

A relation is in Boyce-Codd Normal Form (BCNF) if and only if it is in 3NF and every determinant is a candidate key (or K is a superkey).

That is, K → Ai for every attribute, and there is no subset X (key or nonkey) such that X → Ai where X is different from K.

Example: Employees can have many specialties, and many employees can be within a specialty. Employees can have many managers, but a manager can have only one specialty: Mgr → Specialty

EmpSpecMgr(EID, Specialty, ManagerID)

EID	Speciality	ManagerID
32	Drill	1
33	Weld	2
34	Drill	1

FD ManagerID → Specialty is not currently a key.

Appendix: Multi-Valued Dependency

A multi-valued dependency (MVD) exists when there are at least three attributes in a relation (A, B, and C; and they could be sets), and one attribute (A) determines the other two (B and C) but the other two are independent of each other.

That is, A →B and A → C but B and C have no FDs

Example:

Employees have many specialties and many tools, but tools and specialties are not directly related.

Appendix: Fourth Normal Form

A relation is in **fourth normal form 4NF** if and only if it is in BCNF and there are no multi-valued dependencies.

That is, all attributes of R are also functionally dependent on A.

If A →→ B, then all attributes of R are also functionally dependent on A: A → Ai for each attribute.

Example:

EmpSpecTools(EID, Specialty, ToolID)

EmpSpec(EID, Specialty)

EmpTools(EID, ToolID)

Database Management Systems

Chapter 4
Queries

Jerry Post

DATABASE

Why do we Need Queries

- Natural languages (English) are too vague
  - With complex questions, it can be hard to verify that the question was interpreted correctly, and that the answer we received is truly correct.
  - Consider the question: Who are our best customers?
- We need a query system with more structure
- We need a standardized system so users and developers can learn one method that works on any (most) systems.
  - Query By Example (QBE)
  - SQL

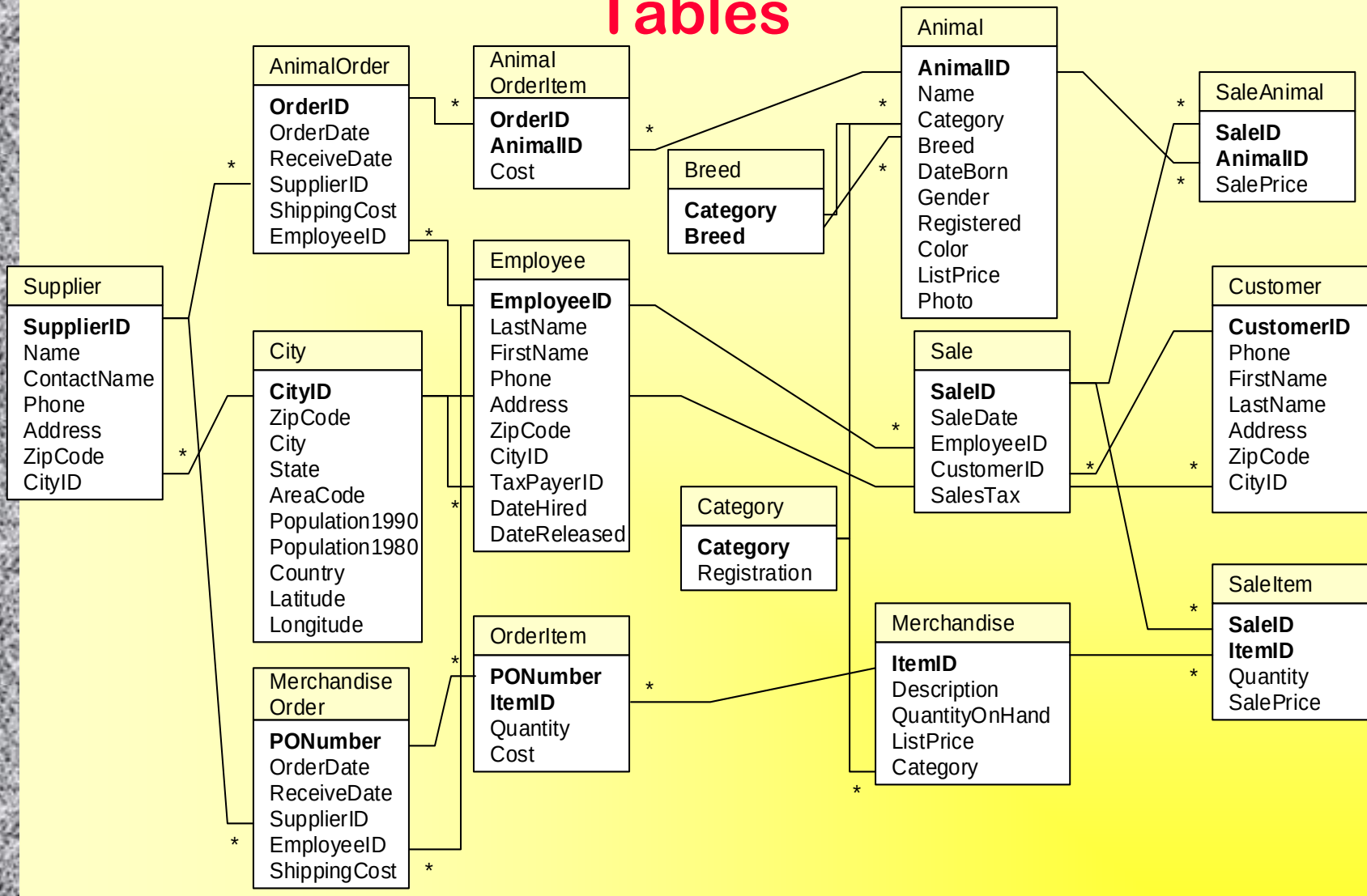
DATABASE

Four Questions to Create a Query

- What output do you want to see?
- What do you already know (or what constraints are given)?
- What tables are involved?
- How are the tables joined together?

DATABASE

Tables



DATABASE

Organization

- Single table
- Constraints
- Computations
- Groups/Subtotals
- Multiple Tables

DATABASE

Sample Questions

- List all animals with yellow in their color.
- List all dogs with yellow in their color born after 6/1/04.
- List all merchandise for cats with a list price greater than $10.
- List all dogs who are male and registered or who were born before 6/1/04 and have white in their color.
- What is the average sale price of all animals?
- What is the total cost we paid for all animals?
- List the top 10 customers and total amount they spent.
- How many cats are in the animal list?
- Count the number of animals in each category.
- List the CustomerID of everyone who bought something between 4/1/04 and 5/31/04.
- List the first name and phone of every customer who bought something between 4/1/04 and 5/31/04.
- List the last name and phone of anyone who bought a registered white cat between 6/1/04 and 12/31/04.
- Which employee has sold the most items?

DATABASE

Query By Example & SQL

What tables?

```
SELECT AnimalID, Category, Breed, Color
FROM   Animal
WHERE (Color LIKE '%Yellow%');
```

Animal
AnimalID
Name
Category
Breed
DateBorn
Gender

What to see?

What conditions?

Field	AnimalID	Category	Breed	Color
Table	Animal	Animal	Animal	Animal
Sort				
Criteria				Like '%Yellow%'
Or				

List all animals with yellow in their color

DATABASE

Basic SQL SELECT

SELECT	columns	What do you want to see?
FROM	tables	What tables are involved?
JOIN	conditions	How are the tables joined?
WHERE	criteria	What are the constraints?

DATABASE

ORDER BY

SELECT	columns
FROM	tables
JOIN	join columns
WHERE	conditions
ORDER BY	columns (ASC DESC)

Animal
AnimalID
Name
Category
Breed
DateBorn
Gender

SELECT Name, Category, Breed
FROM Animal
ORDER BY Category, Breed;

Field	Name	Category	Breed
Table	Animal	Animal	Animal
Sort		Ascending	Ascending
Criteria			
Or			

Name	Category	Breed
Cathy	Bird	African Grey
	Bird	Canary
Debbie	Bird	Cockatiel
	Bird	Cockatiel
Terry	Bird	Lovebird
	Bird	Other
Charles	Bird	Parakeet
Curtis	Bird	Parakeet
Ruby	Bird	Parakeet
Sandy	Bird	Parrot
Hoyt	Bird	Parrot
	Bird	Parrot

DATABASE

DISTINCT

SELECT Category
FROM Animal;

Category
Fish
Dog
Fish
Cat
Cat
Dog
Fish
Dog
Dog
Dog
Fish
Cat
Dog
. . .

SELECT DISTINCT Category
FROM Animal;

Category
Bird
Cat
Dog
Fish
Mammal
Reptile
Spider

DATABASE

Constraints: And

Animal
AnimalID
Name
Category
Breed
DateBorn
Gender

```
SELECT AnimalID, Category, DateBorn
FROM   Animal
WHERE ((Category='Dog')
       AND (Color Like '%Yellow%')
       AND (DateBorn>'01-Jun-2004'));
```

Field	AnimalID	Category	DateBorn	Color
Table	Animal	Animal	Animal	Animal
Sort				
Criteria	'Dog'		>'01-Jun-2004'	Like '%Yellow%'
Or				

List all dogs with yellow in their color born after 6/1/04.

DATABASE

Boolean Algebra

And: Both must be true.
Or: Either one is true.
Not: Reverse the value.

a	b	a AND b	a OR b
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

a = 3

b = -1

c = 2

(a > 4) And (b < 0)

F T

F

(a > 4) Or (b < 0)

F T

T

NOT (b < 0)

T

F

DATABASE

Boolean Algebra

The result is affected by the order of the operations.
Parentheses indicate that an operation should be performed first.
With no parentheses, operations are performed left-to-right.

a = 3

b = -1

c = 2

((a > 4) AND (b < 0)) OR (c > 1)

F T T

F T

T

(a > 4) AND ((b < 0) OR (c > 1))

F T T

F T

F

Always use parentheses, so other people can read and understand your query.

DATABASE

DeMorgan's Law Example

Customer: "I want to look at a cat, but I don't want any cats that are registered or that have red in their color."

Animal
AnimalID Name Category Breed DateBorn Gender

```
SELECT AnimalID, Category, Registered, Color
FROM Animal
WHERE (Category='Cat') AND
        NOT ((Registered is NOT NULL)
                OR (Color LIKE '%Red%')).
```

Field	AnimalID	Category	Registered	Color
Table	Animal	Animal	Animal	Animal
Sort				
Criteria		'Cat'	Is Null	Not Like '%Red%'
Or				

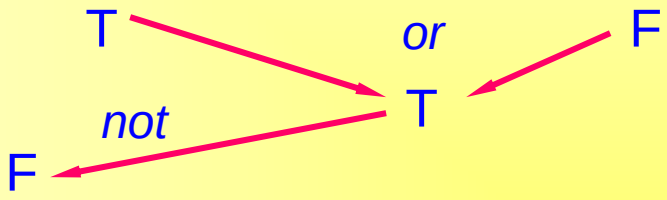
DATABASE

DeMorgan's Law

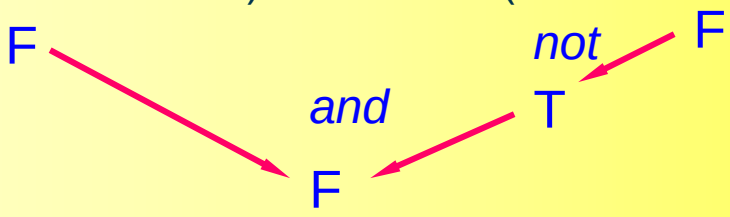
- Negation of clauses
  - Not (A And B) becomes
    Not A Or Not B
  - Not (A Or B) becomes
    Not A And Not B

Registered=ASCF
Color=Black

NOT ((Registered is NOT NULL) OR (Color LIKE '%Red%'))



(Registered is NULL) AND NOT (Color LIKE '%Red%')



Query04_03

Conditions: AND, OR

```
SELECT AnimalID, Category, Gender, Registered, DateBorn, Color
FROM Animal
WHERE (( Category='Dog') AND
                    ( ( (Gender='Male') AND (Registered Is Not Null) ) OR
                      ( (DateBorn<'01-Jun-2004') AND (Color Like '%White%') ) ) );
```

Animal
AnimalID Name Category Breed DateBorn Gender

Field	AnimalID	Category	Gender	Registered	DateBorn	Color
Table	Animal	Animal	Animal	Animal	Animal	Animal
Sort						
Criteria		'Dog'	'Male'	Is Not Null		
Or		'Dog'			< '01-Jun-2004'	Like '%White%'

List all dogs who are male and registered or who were born before 6/1/2004 and have white in their color.

DATABASE

Useful *Where* Conditions

Comparisons	Examples
Operators	<, =, >, <>, BETWEEN, LIKE, IN
Numbers	AccountBalance > 200
Text	
Simple	Name > ‘Jones’
Pattern match one	License LIKE ‘A_ _82_’
Pattern match any	Name LIKE ‘J%’
Dates	SaleDate BETWEEN ’15-Aug-2004’ AND ’31-Aug-2004’
Missing Data	City IS NULL
Negation	Name IS NOT NULL
Sets	Category IN (‘Cat’, ‘Dog’, ‘Hamster’)

Simple Computations

SaleItem(OrderID, ItemID, SalePrice, Quantity)

```
Select OrderID, ItemID, SalePrice, Quantity,
            SalePrice*Quantity As Extended
From SaleItem;
```

OrderID	ItemID	Price	Quantity	Extended
151	9764	19.50	2	39.00
151	7653	8.35	3	25.05
151	8673	6.89	2	13.78

Basic computations (+ - * /) can be performed on numeric data.
The new display column should be given a meaningful name.

DATABASE

Computations: Aggregation--Avg

```
SELECT Avg(SalePrice) AS AvgOfSalePrice
FROM SaleAnimal;
```

SaleAnimal
SaleID AnimalID SalePrice

Field	SalePrice
Table	SaleAnimal
Total	Avg
Sort	
Criteria	
Or	

Sum
Avg
Min
Max
Count
StDev or StdDev
Var

What is the average sale price of all animals?

Computations (Math Operators)

OrderItem
PONumber ItemID Quantity Cost

```
SELECT Sum(Quantity*Cost) AS OrderTotal
FROM OrderItem
WHERE (PONumber=22);
```

Field	PONumber	OrderTotal: Quantity*Cost
Table	OrderItem	OrderItem
Total		
Sort		
Criteria	=22	
Or		

OrderTotal
1798.28

- What is the total value of the order for PONumber 22?
  - Use any common math operators on numeric data.
  - Operate on data in **one row at a time**.

DATABASE

SQL Differences

Task	Access	SQL Server	Oracle
Strings			
Concatenation	FName & " " & LName	FName + ' ' + LName	FName || ' ' || LName
Length	Len(LName)	Length(LName)	LENGTH(LName)
Upper case	UCase(LName)	Upper(LName)	UPPER(LName)
Lower case	LCase(LName)	Lower(LName)	LOWER(LName)
Partial string	MID(LName,2,3)	Substring(LName,2,3)	SUBSTR(LName,2,3)
Dates			
Today	Date( ), Time( ), Now( )	GetDate()	SYSDATE
Month	Month(myDate)	DateName(month, myDate)	TRUNC(myDate, 'mm')
Day	Day(myDate)	DatePart(day, myDate)	TRUNC (myDate, 'dd')
Year	Year(myDate)	DatePart(year, myDate)	TRUNC (myDate, 'yyyy')
Date arithmetic	DateAdd	DateAdd	ADD_MONTHS
	DateDiff	DateDiff	MONTHS_BETWEEN
			LAST_DAY
Formatting	Format(item, format)	Str(item, length, decimal)	TO_CHAR(item, format)
		Cast, Convert	TO_DATE(item, format)
Numbers			
Math functions	Cos, Sin, Tan, Sqrt	Cos, Sin, Tan, Sqrt	COS, SIN, TAN, SQRT
Exponentiation	2 ^ 3	Power(2, 3)	POWER(2,3)
Aggregation	Min, Max, Sum, Count,	Min, Max, Sum, Count,	MIN, MAX, SUM, COUNT,
Statistics	Avg StDev, Var	Avg, StDev, Var, LinReqSlope, Correlation	AVG, STDDEV, VARIANCE, REGR, CORR

DATABASE

Subtotals (Where)

Animal
AnimalID Name Category Breed DateBorn Gender

```
SELECT  Count(AnimalID) AS CountOfAnimalID
FROM    Animal
WHERE   (Category = 'Cat');
```

Field	AnimalID	Category
Table	Animal	Animal
Total	Count	Where
Sort		
Criteria		'Cat'
Or		

How many cats are in the Animal list?

DATABASE

Groups and Subtotals

```
SELECT    Category, Count(AnimalID) AS CountOfAnimalID
FROM      Animal
GROUP BY  Category
ORDER BY  Count(AnimalID) DESC;
```

Animal
AnimalID
Name
Category
Breed
DateBorn
Gender

Field	Category	AnimalID
Table	Animal	Animal
Total	Group By	Count
Sort		Descending
Criteria		
Or		

Category	CountOfAnimalID
Dog	100
Cat	47
Bird	15
Fish	14
Reptile	6
Mammal	6
Spider	3

- Count the number of animals in each category.
  - You could type in each WHERE clause, but that is slow.
  - And you would have to know all of the Category values.

DATABASE

Conditions on Totals (Having)

Animal
AnimalID Name Category Breed DateBorn Gender

```
SELECT    Category, Count(AnimalID) AS CountOfAnimalID
FROM      Animal
GROUP BY  Category
HAVING    Count(AnimalID) > 10
ORDER BY  Count(AnimalID) DESC;
```

Field	Category	AnimalID
Table	Animal	Animal
Total	Group By	Count
Sort		Descending
Criteria		>10
Or		

Category	CountOfAnimalID
Dog	100
Cat	47
Bird	15
Fish	14

Count number of Animals in each Category, but only list them if more than 10.

Where (Detail) v Having (Group)

Animal
AnimalID
Name
Category
Breed
DateBorn
Gender

```
SELECT    Category, Count(AnimalID) AS CountOfAnimalID
FROM      Animal
WHERE     DateBorn > '01-Jun-2004'
GROUP BY  Category
HAVING    Count(AnimalID) > 10
ORDER BY  Count(AnimalID) DESC;
```

Field	Category	AnimalID	DateBorn
Table	Animal	Animal	Animal
Total	Group By	Count	Where
Sort		Descending	
Criteria		>10	>'01-Jun-2004'
Or			

Category	CountOfAnimalID
Dog	30
Cat	18

Count Animals born after 6/1/2004 in each Category, but only list Category if more than 10.

DATABASE

Multiple Tables (Intro & Distinct)

Sale
SaleID
SaleDate
EmployeeID
CustomerID
SalesTax

```
SELECT DISTINCT CustomerID
FROM Sale
WHERE (SaleDate Between '01-Apr-2004'
        And '31-May-2004')
ORDER BY CustomerID;
```

Field	CustomerID	SaleDate
Table	Sale	Sale
Sort	Ascending	
Criteria		Between '01-Apr-2004' And '31-May-2004'
Or		

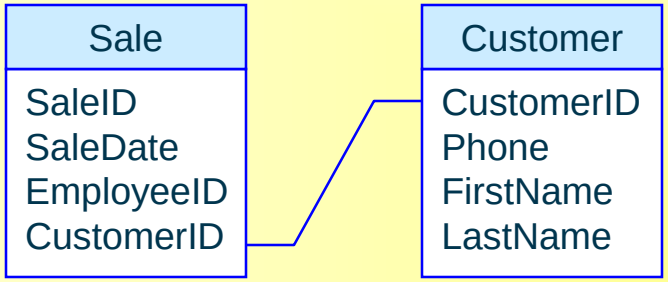
CustomerID
6
8
14
19
22
24
28
36
37
38
39
42
50
57
58
63
74
80
90

List the CustomerID of everyone who bought something between 01-Apr-2004 and 31-May-2004.

DATABASE

Joining Tables

```
SELECT DISTINCT Sale.CustomerID, Customer.LastName
FROM Customer
INNER JOIN Sale ON Customer.CustomerID = Sale.CustomerID
WHERE (SaleDate Between '01-Apr-2004' And '31-May-2004')
ORDER BY Customer.LastName;
```



Field	CustomerID	LastName	SaleDate
Table	Sale	Customer	Sale
Sort		Ascending	
Criteria			Between '01-Apr-2004' And '31-May-2004'
Or			

CustomerID	LastName
22	Adkins
57	Carter
38	Franklin
42	Froedge
63	Grimes
74	Hinton
36	Holland
6	Hopkins
50	Lee
58	McCain
…	

List LastNames of Customers who bought between 4/1/2004 and 5/31/2004.

SQL JOIN

```
FROM table1
INNER JOIN table2
ON table1.column = table2.column
```

SQL 92 syntax (Access and SQL Server)

```
FROM table1, table2
WHERE table1.column = table2.column
```

SQL 89 syntax (Oracle)

```
FROM table1, table2
JOIN table1.column = table2.column
```

Informal syntax

DATABASE

Syntax for Three Tables

SQL '92 syntax to join three tables

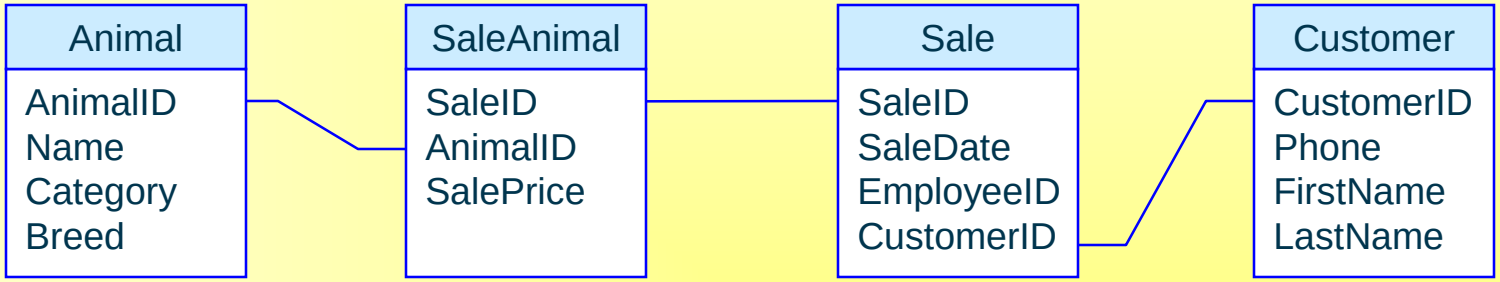
```
FROM Table1
    INNER JOIN (Table2 INNER JOIN Table3
        ON Table2.ColA = Table3.ColA)
    ON Table1.ColB = Table2.ColB
```

Easier notation, but not correct syntax

```
FROM Table1, Table2, Table3
JOIN    Table1.ColB = Table2.ColB
        Table2.ColA = Table3.ColA
```

Multiple Tables (Many)

```
SELECT DISTINCTROW Customer.LastName, Customer.Phone
FROM Customer INNER JOIN (Sale INNER JOIN (Animal INNER JOIN SaleAnimal
 ON Animal.AnimalID = SaleAnimal.AnimalID)  ON Sale.SaleID = SaleAnimal.SaleID)
 ON Customer.CustomerID = Sale.CustomerID
WHERE ((Animal.Category='Cat') AND (Animal.Registered Is Not Null)
 AND (Color Like '%White%')  AND (SaleDate Between '01-Jun-2004' And '31-Dec-2004'));
```



Field	LastName	Phone	Category	Registered	Color	SaleDate
Table	Customer	Customer	Animal	Animal	Animal	Sale
Sort	Ascending					
Criteria			'Cat'	Is Not Null	Like '%White%'	Between '01-Jun-2004' And '31-Dec-2004'
Or						

- List the Last Name and Phone of anyone who bought a registered White cat between 6/1/2004 and 12/31/2004.

DATABASE

Oracle

```
select lastname, phone
from customer inner join sale on customer.customerid = sale.customerid
  inner join saleanimal on sale.saleid = saleanimal.saleid
  inner join animal on saleanimal.animalid = animal.animalid
where (category = 'Cat') and (Registered is not null) and (color like
'%White%')
  AND (saledate between '01-Jun-2004' and '31-Dec-2004')
 ;
```

DATABASE

Building a Query

- List the Last Name and Phone of anyone who bought a registered White cat between 6/1/04 and 12/31/04.
- Identify the tables involved.
  - Look at the columns you want to see.
    - LastName, Phone: Customer
  - Look at the columns used in the constraints.
    - Registered, Color, Category: Animal
    - Sale Date: Sale
  - Find connector tables.
    - To connect Animal to Sale: SaleAnimal
- Select the desired columns and test the query.
- Enter the constraints.
- Set Order By columns.
- Add Group By columns.
- Add summary computations to the SELECT statement.

DATABASE

Joining Tables (Hints)

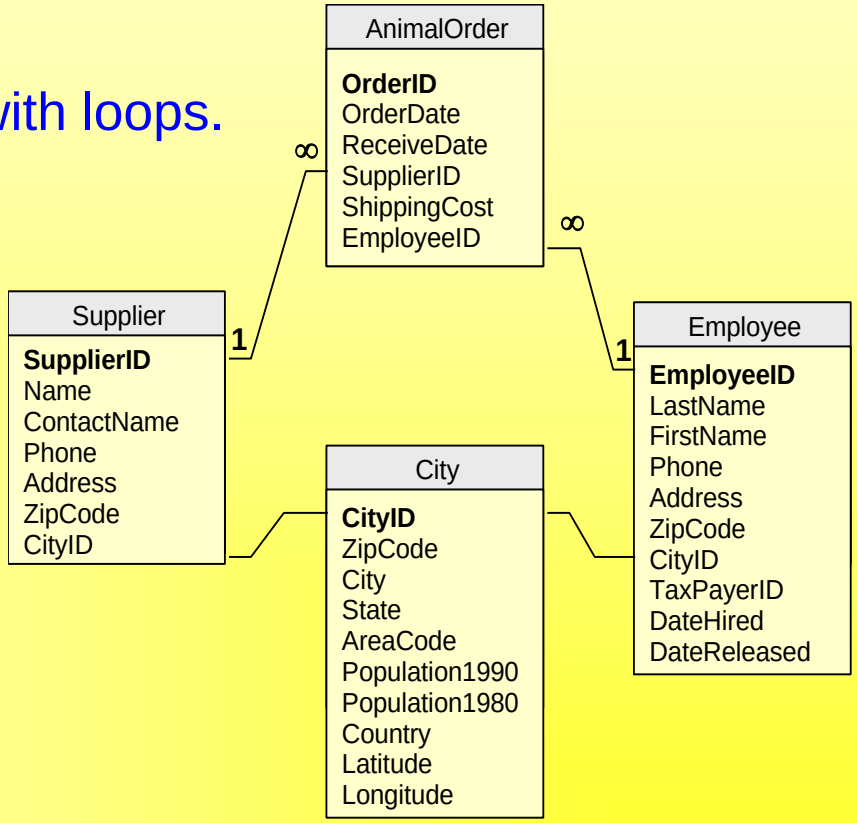
- Build Relationships First
  - Drag and drop
  - From one side to many side
- Avoid multiple ties between tables
- SQL
  - FROM Table1
  - INNER JOIN Table2
  - ON Table1.ColA = Table2.ColB
- Join columns are often keys, but they can be any columns--as long as the domains (types of data) match.
- Multiple Tables
  - FROM (Table1
  - INNER JOIN Table2
  - ON T1.ColA = T2.ColB)
  - INNER JOIN Table3
  - ON T3.ColC = T3.ColD
- Shorter Notation
  - FROM T1, T2, T3
  - JOIN T1.ColA = T2.ColB
  - T1.ColC = T3.ColD
- Shorter Notation is **not** correct syntax, but it is easier to write.

DATABASE

Tables with Multiple Joins

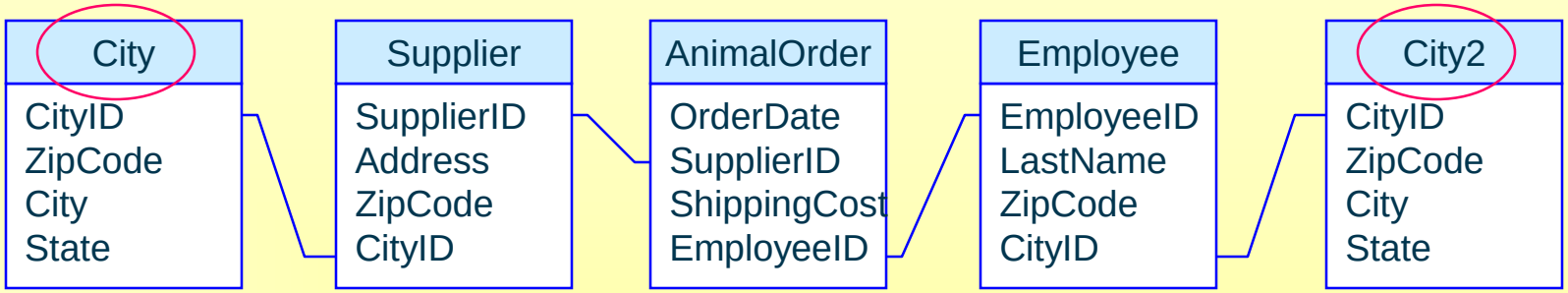
- Potential problem with three or more tables.
- Access uses predefined relationships to automatically determine JOINs.
- JOINS might loop.
- Most queries will not work with loops.

A query with these four tables with four JOINS would only return rows where the Employee had the same ZipCode as the Supplier. If you only need the Supplier city, just delete the JOIN between Employee and ZipCode. If you want both cities, add the ZipCode table **again** as a fifth table.



DATABASE

Table Alias




```
SELECT Supplier.SID, Supplier.CityID, City.City, Employee.EID,
Employee.LastName, Employee.CityID, City2.City
FROM (City INNER JOIN Supplier ON City.CityID = Supplier.CityID) INNER JOIN
((City AS City2 INNER JOIN Employee ON City2.CityID = Employee.CityID)
INNER JOIN AnimalOrder ON Employee.EmployeeID = AnimalOrder.EmployeeID)
ON Supplier.SupplierID = AnimalOrder.SupplierID;
```

SID	Supplier.CityID	City.City	EID	LastName	Employee.CityID	City2.City
4	7972	Middlesboro	5	James	7083	Orlando
2	10896	Springfield	1	Reeves	9201	Lincoln
4	7972	Middlesboro	3	Reasoner	8313	Springfield
9	10740	Columbia	8	Carpenter	10592	Philadelphia
5	10893	Smyrna	3	Reasoner	8313	Springfield

DATABASE

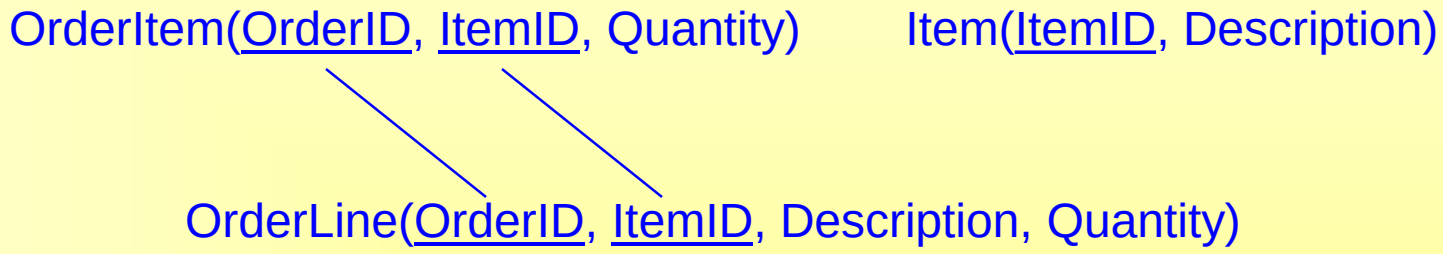
Saved Query: Create View

- Save a query
  - Faster: only enter once
  - Faster: only analyze once
- Any SELECT statement
- Can use the View within other SQL queries.

```
CREATE VIEW Kittens AS
SELECT *
FROM Animal
WHERE (Category = 'Cat') AND
(Today - DateBorn < 180);
```

```
SELECT Avg(ListPrice)
FROM Kittens
WHERE (Color LIKE '%Black%');
```

Updateable Views



- To be updateable, a view must focus on one primary table. (OrderItem)
  - Goal is to change data in only one table. (OrderItem)
  - Data can be displayed from other tables. (Item)
  - Never include or attempt to change primary keys from more than one table. (Item.ItemID)

Non Updateable View

OrderItem(OrderID, ItemID, Quantity)

121	57	3
121	82	2
122	57	1

Item(ItemID, Description)

57	Cat food
58	Dog food
59	Bird food

OrderLine(OrderID, Item.ItemID, Description, Quantity)

121	57	Cat food	3
121	82	Bird feeder	2
122	~~57~~ 32	Cat food	1

If you attempt to change the Item.ItemID in the OrderLineView:

You will simply change the primary key value in the Item table.

It will **not** add a new row to the OrderItem table.

SQL Syntax: ALTER TABLE

ALTER TABLE table
  ADD COLUMN column datatype (size)
  DROP COLUMN column

See also:

CREATE TABLE

DROP TABLE

SQL Syntax: COMMIT

COMMIT WORK

See also: ROLLBACK

SQL Syntax: CREATE INDEX

CREATE [UNIQUE] INDEX index
ON table (column1, column2, …)
WITH {PRIMARY | DISALLOW NULL | IGNORE NULL}

See also: CREATE TABLE

DATABASE

SQL Syntax: CREATE TABLE

```
CREATE TABLE table
(
   column1      datatype (size) [NOT NULL] [index1] ,
   column2      datatype (size) [NOT NULL] [index2],
   … ,
   CONSTRAINT pkname PRIMARY KEY (column, …),
   CONSTRAINT fkname FOREIGN KEY (column)
         REFERENCES existing_table (key_column)
         ON DELETE CASCASDE
)
```

See also:

- ALTER TABLE
- DROP TABLE

SQL Syntax: CREATE VIEW

CREATE VIEW viewname AS
SELECT …

See also: SELECT

SQL Syntax: DELETE

```
DELETE
FROM table
WHERE condition
```

See also: DROP

DATABASE

SQL Syntax: DROP

DROP INDEX index ON table

DROP TABLE

DROP VIEW

See also: DELETE

DATABASE

SQL Syntax: INSERT

INSERT INTO table (column1, column2, …)
VALUES (value1, value2, …)

INSERT INTO newtable (column1, column2, …)
SELECT …

See also: SELECT

SQL Syntax: GRANT

GRANT privilege
ON object
TO user | PUBLIC

privileges
ALL, ALTER, DELETE, INDEX,
INSERT, SELECT, UPDATE

See also: REVOKE

SQL Syntax: REVOKE

REVOKE privilege
ON object
FROM user | PUBLIC

privileges
ALL, ALTER, DELETE, INDEX,
INSERT, SELECT, UPDATE

See also: GRANT

SQL Syntax: ROLLBACK

SAVEPOINT savepoint {optional}

ROLLBACK WORK
TO savepoint

See also: COMMIT

SQL Syntax: SELECT

SELECT DISTINCT table.column {AS alias} , . . .
FROM table/query
INNER JOIN table/query ON T1.ColA = T2.ColB
WHERE (condition)
GROUP BY column
HAVING (group condition)
ORDER BY table.column
{ UNION, INTERSECT, EXCEPT … }

SQL Syntax: SELECT INTO

```
SELECT column1, column2, …
INTO newtable
FROM tables
WHERE condition
```

See also: SELECT

DATABASE

SQL Syntax: UPDATE

```
UPDATE TABLE table
   SET column1 = value1, column2 = value2, …
   WHERE condition
```

See also: DELETE