College of Information Studies
University of Maryland Hornbake Library Building College Park, MD 20742-4345

Microsoft Access and SQL

Week 9

LBSC 690

Information Technology

Muddiest Points

- Connecting a design to an implementation
- Normalization
- Many-to-many
- Operators

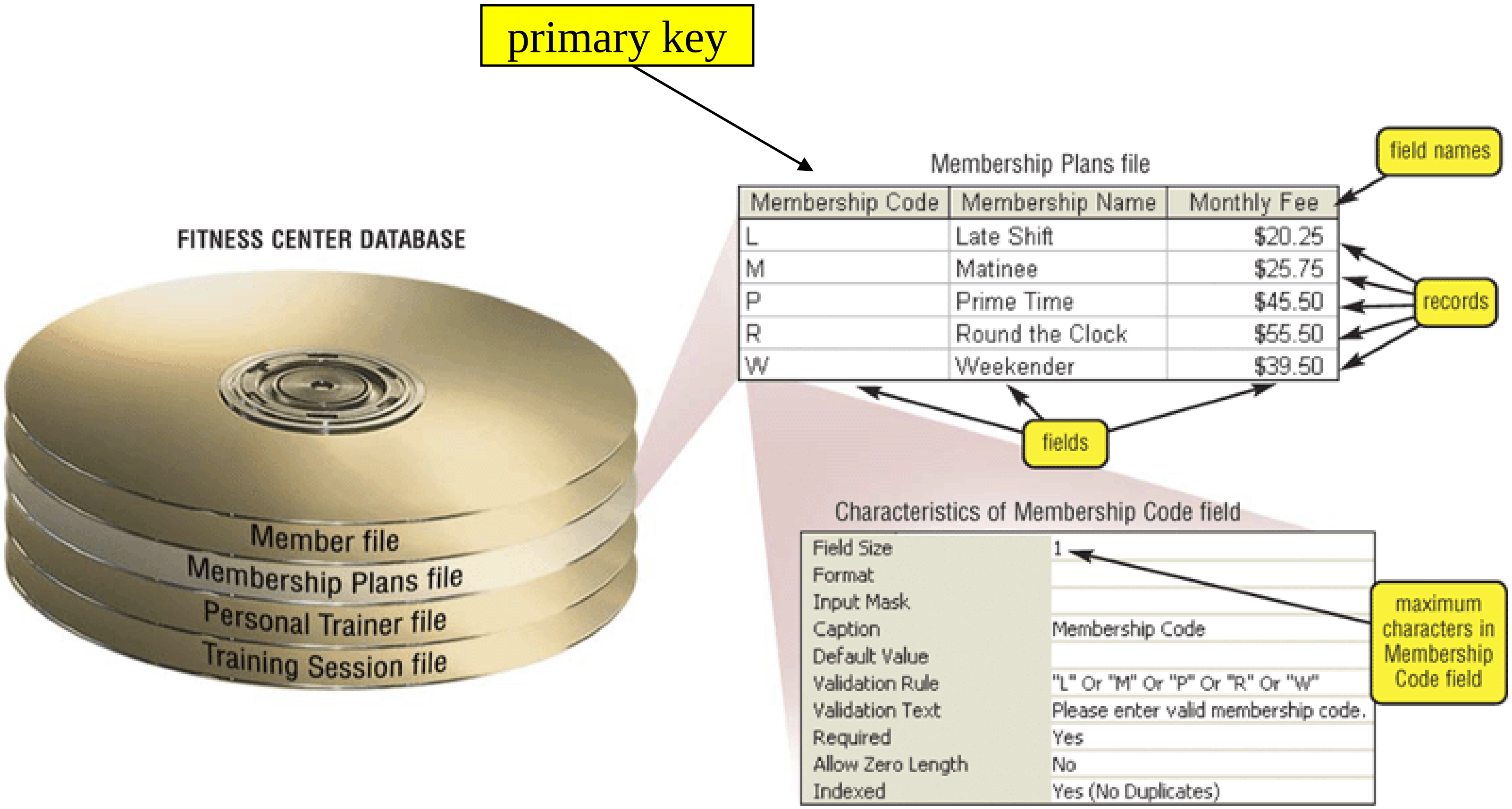
Databases

- Database
  - Collection of data, organized to support access
  - Models some aspects of reality
- DataBase Management System (DBMS)
  - Software to create and access databases
- Relational Algebra
  - Special-purpose programming language

Structured Information

- Field An "atomic" unit of data
  - number, string, true/false, …
- Record A collection of related fields
- Table A collection of related records
  - Each record is one row in the table
  - Each field is one column in the table
- Primary Key The field that identifies a record
  - Values of a primary key must be unique
- Database A collection of tables

A Simple Example



Registrar Example

- Which students are in which courses?

- What do we need to know about the students?
  - first name, last name, email, department

- What do we need to know about the courses?
  - course ID, description, enrolled students, grades

A “Flat File” Solution

Student ID	Last Name	First Name	Department ID	Departmen	Course ID	Course description	Grades	email
1	Arrows	John	EE	EE	lbsc690	Information Technology	90	jarrows@wam
1	Arrows	John	EE	Elec Engin	ee750	Communication	95	ja_2002@yahoo
2	Peters	Kathy	HIST	HIST	lbsc690	Informatino Technology	95	kpeters2@wam
2	Peters	Kathy	HIST	history	hist405	American History	80	kpeters2@wma
3	Smith	Chris	HIST	history	hist405	American History	90	smith2002@glue
4	Smith	John	CLIS	Info Sci	lbsc690	Information Technology	98	js03@wam

Discussion Topic

Why is this a bad approach?

Goals of “Normalization”

- Save space
  - Save each fact only once
- More rapid updates
  - Every fact only needs to be updated once
- More rapid search
  - Finding something once is good enough
- Avoid inconsistency
  - Changing data once changes it everywhere

Relational Algebra

- Tables represent "relations"
  - Course, course description
  - Name, email address, department

- Named fields represent "attributes"

- Each row in the table is called a "tuple"
  - The order of the rows is not important

- Queries specify desired conditions
  - The DBMS then finds data that satisfies them

A Normalized Relational Database

Student Table

Student ID	Last Name	First Name	Department ID	email
1	Arrows	John	EE	jarrows@wam
2	Peters	Kathy	HIST	kpeters2@wam
3	Smith	Chris	HIST	smith2002@glue
4	Smith	John	CLIS	js03@wam

Department Table

Department ID	Department
EE	Electronic Engineering
HIST	History
CLIS	Information Stuides

Course Table

Course ID	Course Description
lbsc690	Information Technology
ee750	Communication
hist405	American History

Enrollment Table

Student ID	Course ID	Grades
1	lbsc690	90
1	ee750	95
2	lbsc690	95
2	hist405	80
3	hist405	90
4	lbsc690	98

Approaches to Normalization

- For simple problems (like the homework)
  - Start with “binary relationships”
    - Pairs of fields that are related
  - Group together wherever possible
  - Add keys where necessary
- For more complicated problems
  - Entity relationship modeling (LBSC 670)

Example of Join

Student Table

Student ID	Last Name	First Name	Department ID	email
1	Arrows	John	EE	jarrows@wam
2	Peters	Kathy	HIST	kpeters2@wam
3	Smith	Chris	HIST	smith2002@glue
4	Smith	John	CLIS	js03@wam

Department Table

Department ID	Department
EE	Electronic Engineering
HIST	History
CLIS	Information Stuides

"Joined" Table

Student ID	Last Name	First Name	Department ID	Department	email
1	Arrows	John	EE	Electronic Engineering	jarrows@wam
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue
4	Smith	John	CLIS	Information Stuides	js03@wam

Problems with Join

- Data modeling for join is complex
  - Useful to start with E-R modeling
- Join are expensive to compute
  - Both in time and storage space
- But it is joins that make databases relational
  - Projection and restriction also used in flat files

Some Lingo

- "Primary Key" uniquely identifies a record
  - e.g. student ID in the student table
- "Compound" primary key
  - Synthesize a primary key with a combination of fields
  - e.g., Student ID + Course ID in the enrollment table
- "Foreign Key" is primary key in the <u>other</u> table
  - Note: it need not be unique in <u>this</u> table

Project

New Table

Student ID	Last Name	First Name	Department ID	Department	email
1	Arrows	John	EE	Electronic Engineering	jarrows@wam
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue
4	Smith	John	CLIS	Information Stuides	js03@wam

SELECT Student ID, Department

Student ID	Department
1	Electronic Engineering
2	History
3	History
4	Information Stuides

Restrict

New Table

Student ID	Last Name	First Name	Department ID	Department	email
1	Arrows	John	EE	Electronic Engineering	jarrows@wam
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue
4	Smith	John	CLIS	Information Stuides	js03@wam

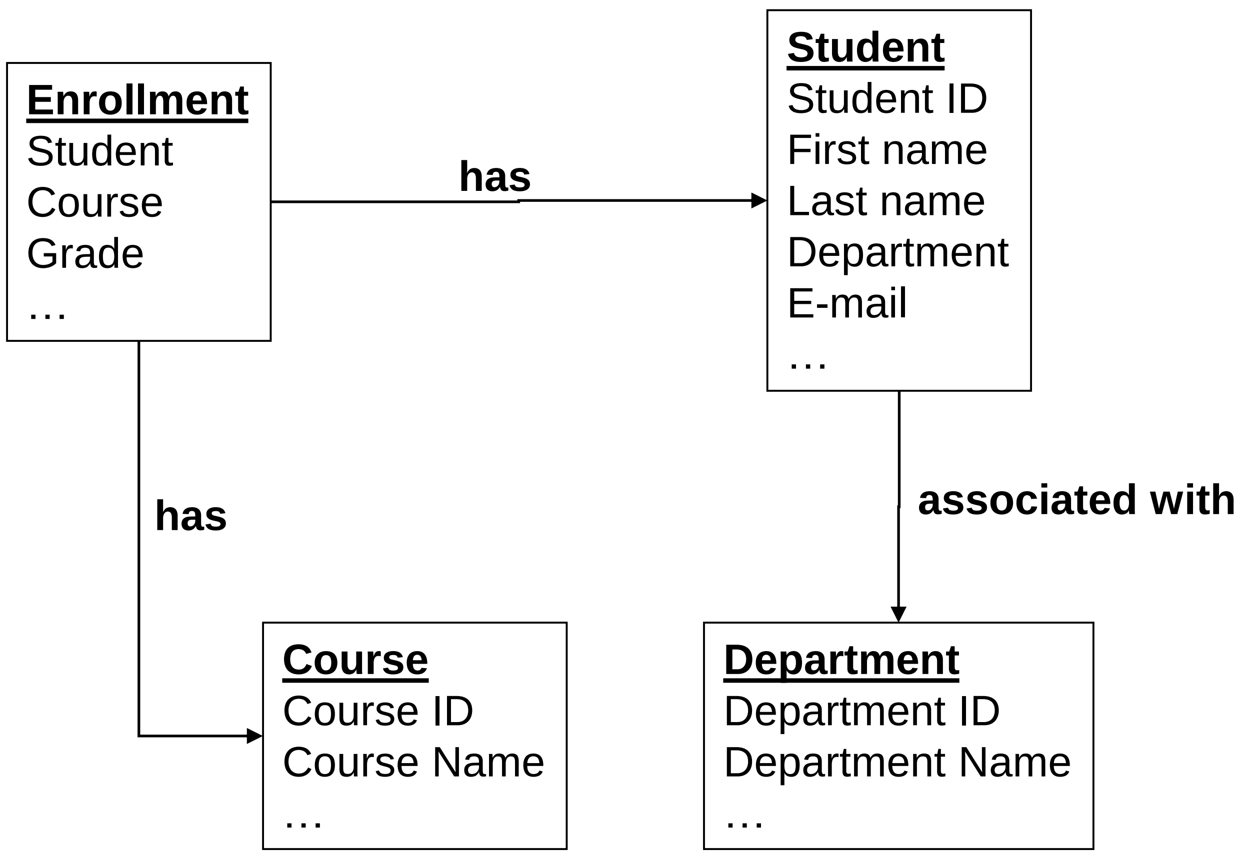
WHERE Department ID = “HIST”

Student ID	Last Name	First Name	Department ID	Department	email
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue

Entity-Relationship Diagrams

- Graphical visualization of the data model
- Entities are captured in boxes
- Relationships are captured using arrows

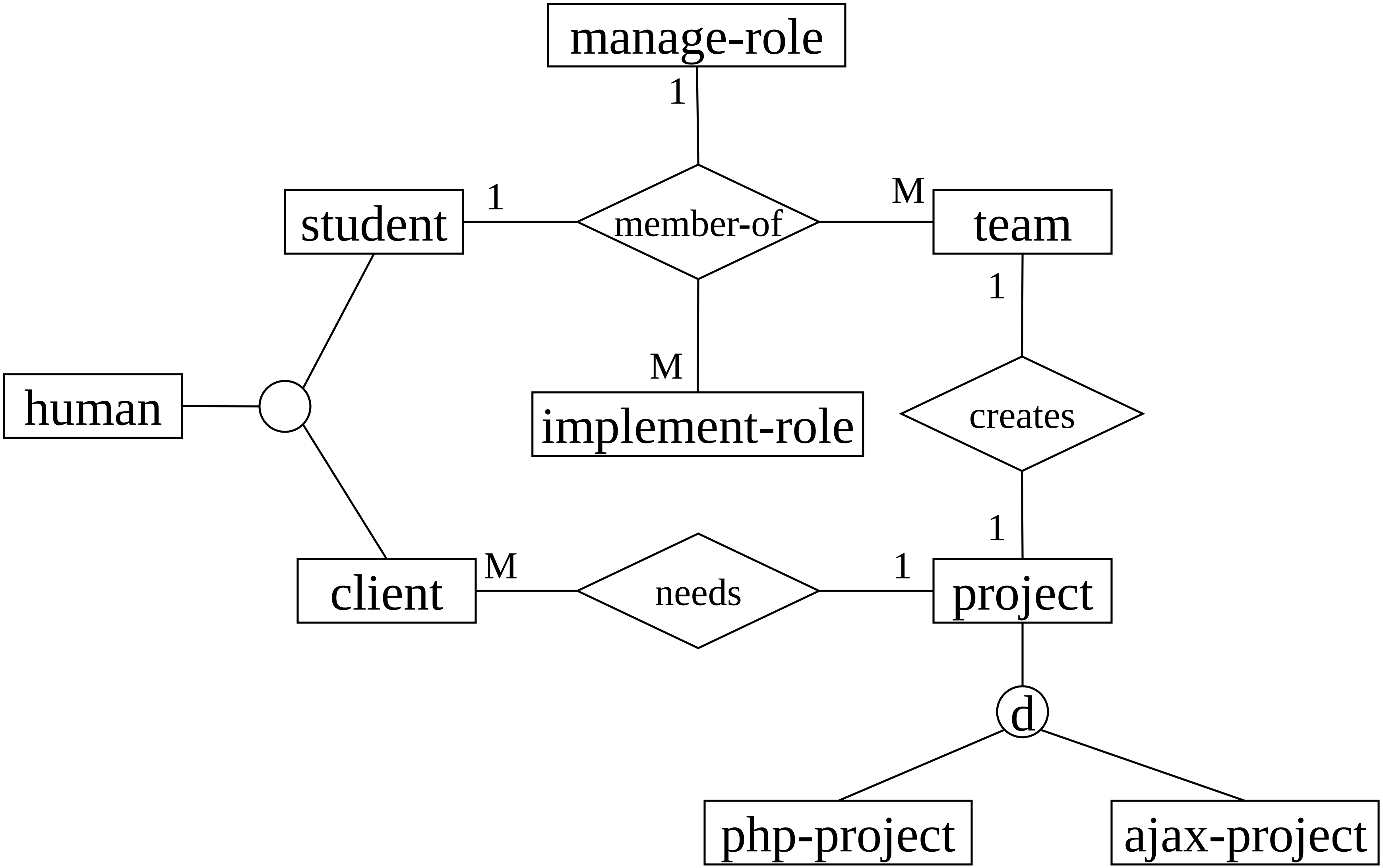
Registrar ER Diagram



Getting Started with E-R Modeling

- What **questions** must you answer?
- What **data** is needed to generate the answers?
  - Entities
    - Attributes of those entities
  - Relationships
    - Nature of those relationships
- How will the user interact with the system?
  - Relating the question to the available data
  - Expressing the answer in a useful form

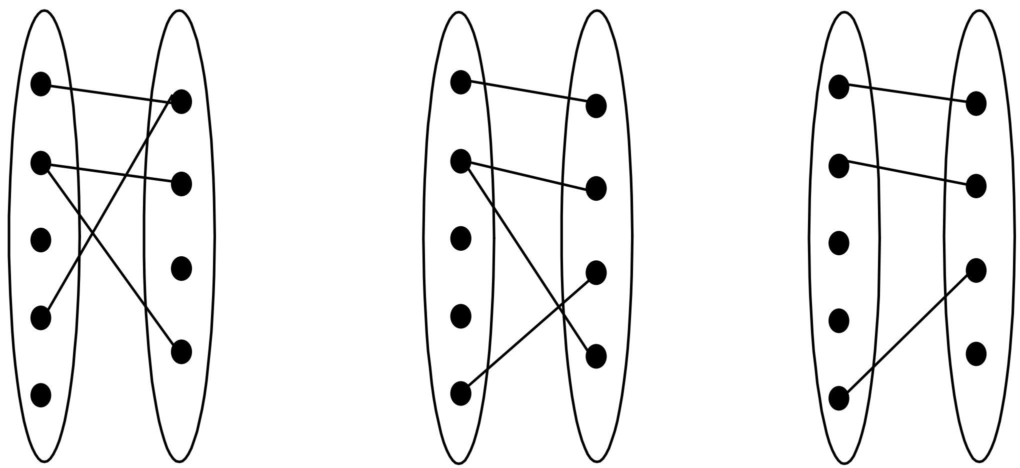
Project Team E-R Example



Components of E-R Diagrams

- Entities
  - Types
    - Subtypes (disjoint / overlapping)
  - Attributes
    - Mandatory / optional
  - Identifier
- Relationships
  - Cardinality
  - Existence
  - Degree

Types of Relationships



Many-to-Many **1-to-Many** **1-to-1**

Making Tables from E-R Diagrams

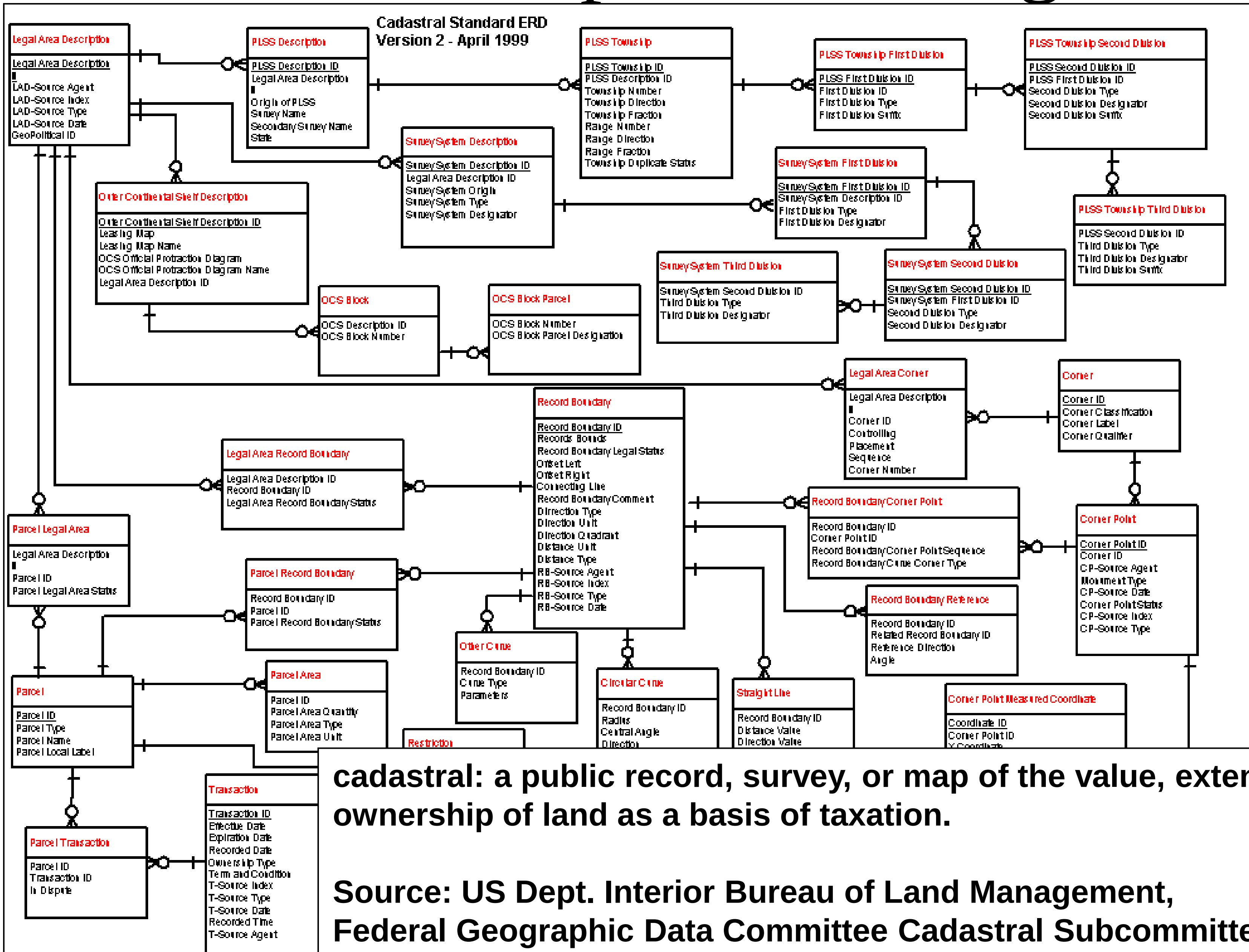
- Pick a primary key for each entity
- Build the tables
  - One per entity
  - Plus one per M:M relationship
  - Choose terse but memorable table and field names
- Check for parsimonious representation
  - Relational "normalization"
  - Redundant storage of computable values
- Implement using a DBMS

- 1NF: <u>Single-valued</u> <u>indivisible</u> (atomic) attributes
  - Split “Doug Oard” to two attributes as (“Doug”, “Oard”)
  - Model M:M implement-role relationship with a table
- 2NF: Attributes depend on <u>complete</u> primary key
  - (<u>id, impl-role</u>, name)->(<u>id</u>, name)+(<u>id, impl-role</u>)
- 3NF: Attributes depend <u>directly</u> on primary key
  - (<u>id</u>, addr, city, state, zip)->(<u>id</u>, addr, zip)+(<u>zip</u>, city, state)
- 4NF: Divide independent M:M tables
  - (<u>id</u>, role, courses) -> (<u>id</u>, role) + (<u>id</u>, courses)
- 5NF: Don’t enumerate derivable combinations

Normalized Table Structure

- Persons: id, fname, lname, userid, password
- Contacts: id, ctype, cstring
- Ctlabels: ctype, string
- Students: id, team, mrole
- Iroles: id, irole
- Rlabels: role, string
- Projects: team, client, pstring

A More Complex ER Diagram



cadastral: a public record, survey, or map of the value, extent, and ownership of land as a basis of taxation.

Source: US Dept. Interior Bureau of Land Management, Federal Geographic Data Committee Cadastral Subcommittee
http://www.fairview-industries.com/standardmodule/cad-erd.htm

Database Integrity

- Registrar database must be internally consistent
  - Enrolled students must have an entry in student table
  - Courses must have a name

- What happens:
  - When a student withdraws from the university?
  - When a course is taken off the books?

Integrity Constraints

- Conditions that must always be true
  - Specified when the database is designed
  - Checked when the database is modified
- RDBMS ensures integrity constraints are respected
  - So database contents remain faithful to real world
  - Helps avoid data entry errors

Referential Integrity

- Foreign key values must exist in other table
  - If not, those records cannot be joined
- Can be enforced when data is added
  - Associate a primary key with each foreign key
- Helps avoid erroneous data
  - Only need to ensure data quality for primary keys

Database “Programming”

- Natural language
  - Goal is ease of use
    - e.g., Show me the last names of students in CLIS
  - Ambiguity sometimes results in errors
- Structured Query Language (SQL)
  - Consistent, unambiguous interface to any DBMS
  - Simple command structure:
    - e.g., SELECT Last name FROM Students WHERE Dept=CLIS
  - Useful standard for inter-process communications
- Visual programming (e.g., Microsoft Access)
  - Unambiguous, and easier to learn than SQL

The SELECT Command

- Project chooses columns
  - Based on their label
- Restrict chooses rows
  - Based on their contents
    - e.g. department ID = “HIST”
- These can be specified together
  - SELECT Student ID, Dept WHERE Dept = “History”

Restrict Operators

- Each SELECT contains a single WHERE
- Numeric comparison
  - <, >, =, <>, …
    - e.g., grade<80
- Boolean operations
  - e.g., Name = “John” AND Dept <> “HIST”

Databases in the Real World

- Some typical database applications:
  - Banking (e.g., saving/checking accounts)
  - Trading (e.g., stocks)
  - Airline reservations

- Characteristics:
  - Lots of data
  - Lots of concurrent access
  - Must have fast access
  - “Mission critical”

Concurrency

- Thought experiment: You and your project partner are editing the same file…
  - Scenario 1: you both save it at the same time
  - Scenario 2: you save first, but before it's done saving, your partner saves

Whose changes survive?
A) Yours B) Partner's C) neither D) both E) ???

Concurrency Example

- Possible actions on a checking account
  - Deposit check (read balance, write new balance)
  - Cash check (read balance, write new balance)
- Scenario:
  - Current balance: $500
  - You try to deposit a $50 check and someone tries to cash a $100 check at the same time
  - Possible sequences: (what happens in each case?)

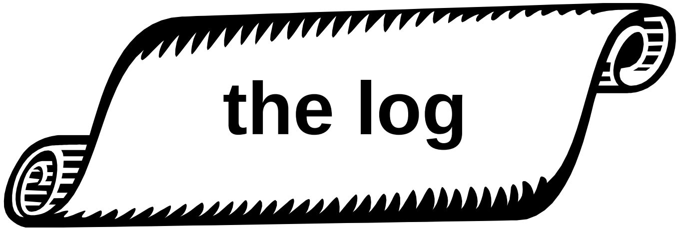
Deposit: read balance	**Deposit: read balance**	**Deposit: read balance**
Deposit: write balance	**Cash: read balance**	**Cash: read balance**
Cash: read balance	**Cash: write balance**	**Deposit: write balance**
Cash: write balance	**Deposit: write balance**	**Cash: write balance**

Database Transactions

- Transaction: sequence of grouped database actions
  - e.g., transfer $500 from checking to savings
- "ACID" properties
  - **Atomicity**
    - All-or-nothing
  - **Consistency**
    - Each transaction must take the DB between consistent states.
  - **Isolation:**
    - Concurrent transactions must appear to run in isolation
  - **Durability**
    - Results of transactions must survive even if systems crash

Making Transactions

- Idea: keep a log (history) of all actions carried out while executing transactions
  - Before a change is made to the database, the corresponding log entry is forced to a safe location



- Recovering from a crash:
  - Effects of partially executed transactions are undone
  - Effects of committed transactions are redone

Key Ideas

- Databases are a good choice when you have
  - Lots of data
  - A problem that contains inherent <u>relationships</u>
- Design before you implement
  - This is just another type of programming
  - The mythical person-month applies!
- Join is the most important concept
  - Project and restrict just remove undesired stuff

RideFinder Exercise

- Design a database to match passengers with available rides for Spring Break
  - Drivers phone in available seats
    - They want to know about interested passengers
  - Passengers call up looking for rides
    - They want to know about available rides
    - No “ride wanted” ads
  - These things happen in no particular order

Exercise Goals

- Identify the tables you will need
  - First decide what data you will need
    - What questions will be asked?
  - Then design normalized tables
    - Start with binary relations if that helps
- Design the queries
  - Using join, project and restrict
  - What happens when a passenger calls?
  - What happens when a driver calls?

Reminder: Starting E-R Modeling

- What **<u>questions</u>** must you answer?
- What **<u>data</u>** is needed to generate the answers?
  - Entities
    - Attributes of those entities
  - Relationships
    - Nature of those relationships
- How will the user interact with the system?
  - Relating the question to the available data
  - Expressing the answer in a useful form

Exercise Logistics

- Work in groups of 3 or 4
- Brainstorm data requirements for 5 minutes
  - Do passengers care about the price?
  - Do drivers care how much luggage there is?
- Develop tables and queries for 15 minutes
  - Don’t get hung up on one thing too long
- Compare you answers with another group
  - Should take about 5 minutes each

Making Tables from E-R Diagrams

- Pick a primary key for each entity
- Build the tables
  - One per entity
  - Plus one per M:M relationship
  - Choose terse but memorable table and field names
- Check for parsimonious representation
  - Relational "normalization"
  - Redundant storage of computable values
- Implement using a DBMS

- 1NF: Single-valued indivisible (atomic) attributes
  - Split “Doug Oard” to two attributes as (“Doug”, “Oard”)
  - Model M:M implement-role relationship with a table
- 2NF: Attributes depend on complete primary key
  - (id, impl-role, name)->(id, name)+(id, impl-role)
- 3NF: Attributes depend directly on primary key
  - (id, addr, city, state, zip)->(id, addr, zip)+(zip, city, state)
- 4NF: Divide independent M:M tables
  - (id, role, courses) -> (id, role) + (id, courses)
- 5NF: Don’t enumerate derivable combinations

Using Microsoft Access

- Create a database called M:\rides.mdb
  - File->New->Blank Database
- Specify the fields (columns)
  - "Create a Table in Design View"
- Fill in the records (rows)
  - Double-click on the icon for the table

Creating Fields

- Enter field name
  - Must be unique, but only within the same table
- Select field type from a menu
  - Use date/time for times
  - Use text for phone numbers
- Designate primary key (right mouse button)
- Save the table
  - That’s when you get to assign a table name

Entering Data

- Open the table
  - Double-click on the icon
- Enter new data in the bottom row
  - A new (blank) bottom row will appear
- Close the table
  - No need to “save” – data is stored automatically

Building Queries

- Copy ride.mdb to your M:\ drive
- "Create Query in Design View"
  - In "Queries"
- Choose two tables, Flight and Company
- Pick each field you need using the menus
  - Unclick "show" to not project
  - Enter a criterion to "restrict"
- Save, exit, and reselect to run the query

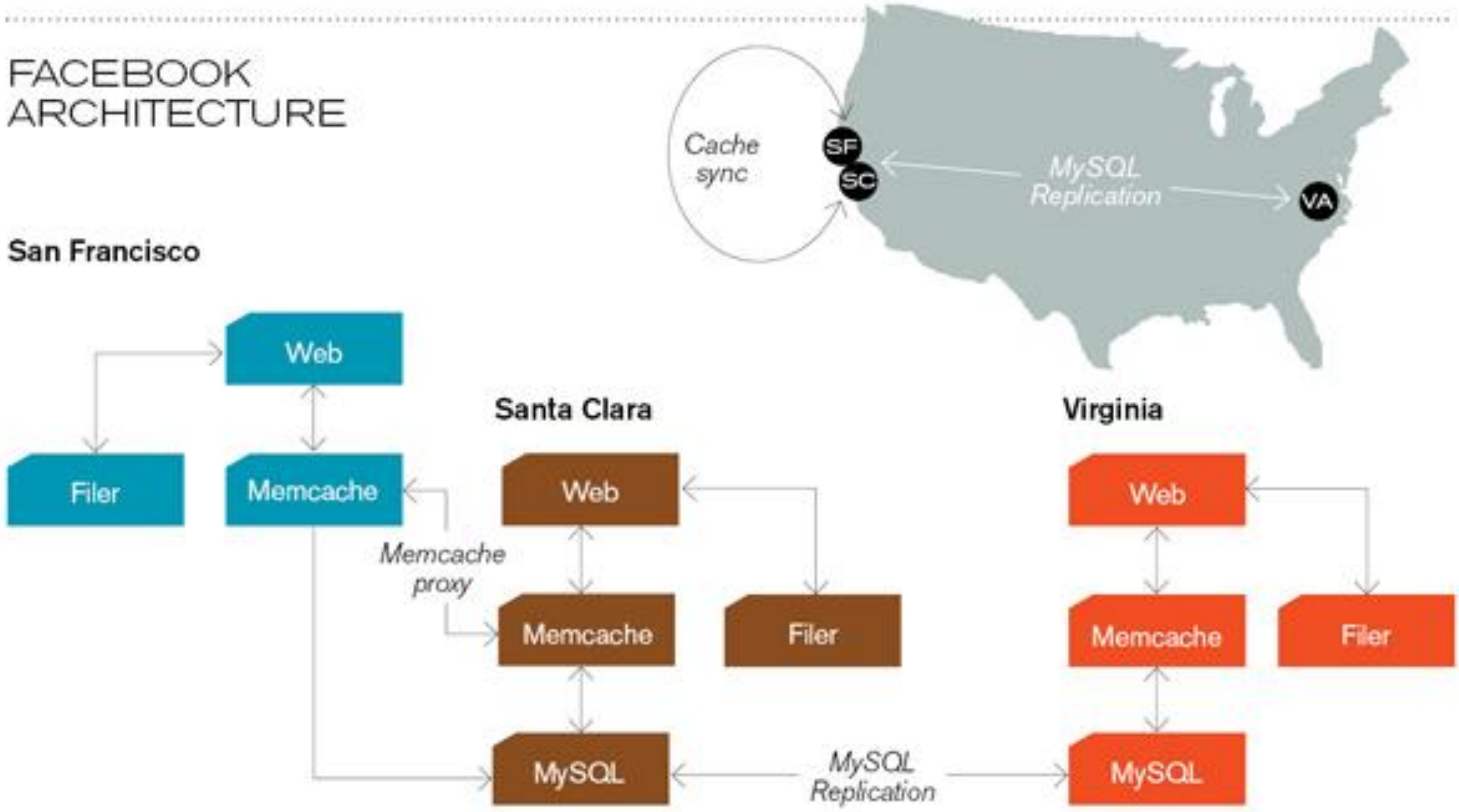
Some Details About Access

- Joins are automatic if field names are same
  - Otherwise, drag a line between the fields
- Sort order is easy to specify
  - Use the menu
- Queries form the basis for reports
  - Reports give good control over layout
  - Use the report wizard - the formats are complex
- Forms manage input better than raw tables
  - Invalid data can be identified when input
  - Graphics can be incorporated

Databases in the Real World

- Some typical database applications:
  - Banking (e.g., saving/checking accounts)
  - Trading (e.g., stocks)
  - Airline reservations

- Characteristics:
  - Lots of data
  - Lots of concurrent access
  - Must have fast access
  - “Mission critical”



Caching servers: 15 million requests per second, 95% handled by memcache (15 TB of RAM)

Database layer: 800 eight-core Linux servers running MySQL (40 TB user data)

Source: Technology Review (July/August, 2008)

Database Integrity

- Registrar database must be internally consistent
  - Enrolled students must have an entry in student table
  - Courses must have a name

- What happens:
  - When a student withdraws from the university?
  - When a course is taken off the books?

Integrity Constraints

- Conditions that must always be true
  - Specified when the database is designed
  - Checked when the database is modified
- RDBMS ensures integrity constraints are respected
  - So database contents remain faithful to real world
  - Helps avoid data entry errors

Referential Integrity

- Foreign key values must exist in other table
  - If not, those records cannot be joined
- Can be enforced when data is added
  - Associate a primary key with each foreign key
- Helps avoid erroneous data
  - Only need to ensure data quality for primary keys

Concurrency

- Thought experiment: You and your project partner are editing the same file…
  - Scenario 1: you both save it at the same time
  - Scenario 2: you save first, but before it's done saving, your partner saves

Whose changes survive?

A) Yours B) Partner's C) neither D) both E) ???

Concurrency Example

- Possible actions on a checking account
  - Deposit check (read balance, write new balance)
  - Cash check (read balance, write new balance)
- Scenario:
  - Current balance: $500
  - You try to deposit a $50 check and someone tries to cash a $100 check at the same time
  - Possible sequences: (what happens in each case?)

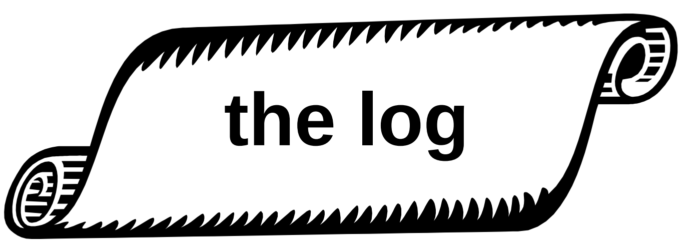
Deposit: read balance	Deposit: read balance	Deposit: read balance
Deposit: write balance	Cash: read balance	Cash: read balance
Cash: read balance	Cash: write balance	Deposit: write balance
Cash: write balance	Deposit: write balance	Cash: write balance

Database Transactions

- Transaction: sequence of grouped database actions
  - e.g., transfer $500 from checking to savings
- "ACID" properties
  - **Atomicity**
    - All-or-nothing
  - **Consistency**
    - Each transaction must take the DB between consistent states.
  - **Isolation:**
    - Concurrent transactions must appear to run in isolation
  - **Durability**
    - Results of transactions must survive even if systems crash

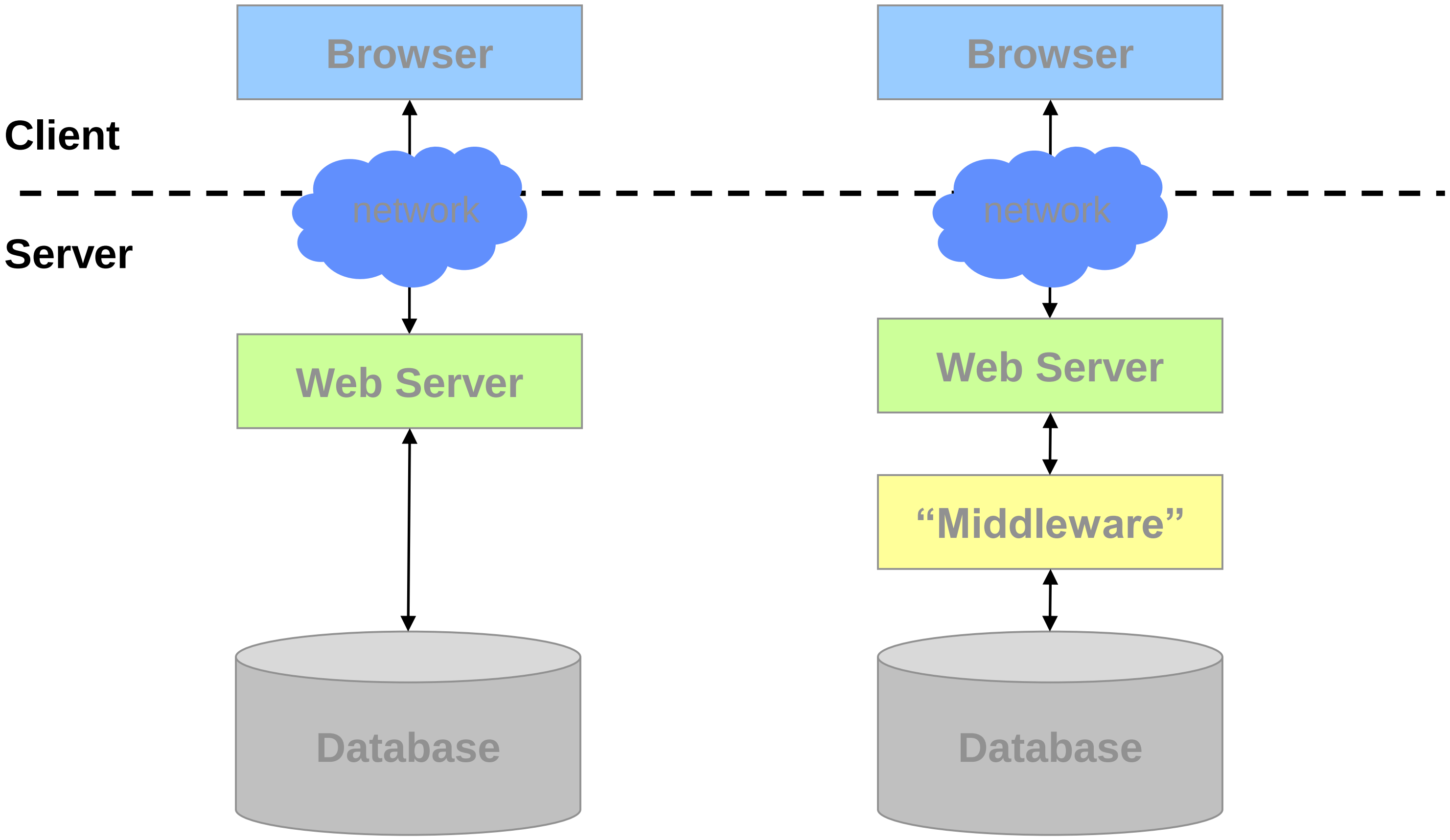
Making Transactions

- Idea: keep a log (history) of all actions carried out while executing transactions
  - Before a change is made to the database, the corresponding log entry is forced to a safe location



- Recovering from a crash:
  - Effects of partially executed transactions are undone
  - Effects of committed transactions are redone

Databases in Web Applications



Why Database-Generated Pages?

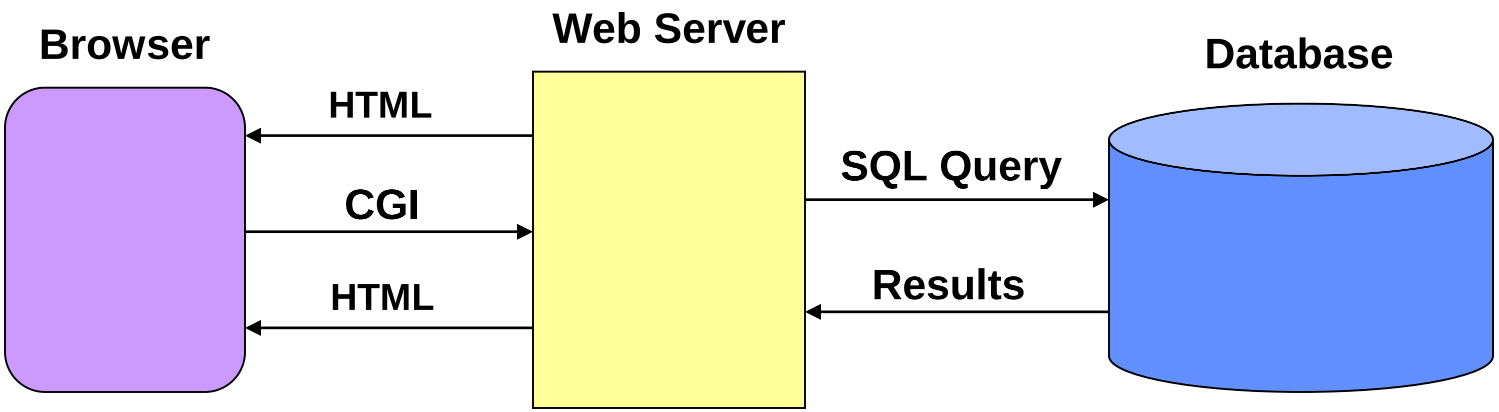
- Remote access to a database
  - Client does not need the database software
- Serve rapidly changing information
  - e.g., Airline reservation systems
- Provide multiple "access points"
  - By subject, by date, by author, …
- Record user responses in the database

Issues to Consider

- Benefits
  - Multiple views
  - Data reuse
  - Scalable
  - Access control

- Costs
  - Formal modeling
  - Complex (learn, design, implement, debug)
  - Brittle (relies on multiple communicating servers)
  - Not crawlable

Putting the Pieces Together



Structured Query Language

DESCRIBE Flight;

Flight : Table

Field Name	Data Type
Flight Number	Text
Origin	Text
Destination	Text
Departure Time	Date/Time
Arrival Time	Date/Time
Available Seats	Number
Company Name	Text
Price	Currency

Structured Query Language

SELECT * FROM Flight;

Flight : Table

	Flight Number	Origin	Destination	Departure Time	Arrival Time	Available Seats	Company Name	Price
▶	CA210	DC	Austin	6:00:00 AM	11:00:00 AM	0	Cal Air	$200.00
	CA345	San Jose	San Diego	9:00:00 AM	10:30:00 AM	20	Cal Air	$100.00
	FT900	Chicago	New York	2:00:00 PM	5:00:00 PM	1	Fancy Trans	$200.00
	GJ405	DC	San Jose	12:30:00 PM	8:45:00 PM	10	Green Jet	$340.00
	GJ908	New York	Austin	8:00:00 AM	12:00:00 PM	2	Green Jet	$250.00
	TP123	New York	San Jose	7:00:00 AM	11:00:00 AM	2	Trans Planet	$400.00
*						0		$0.00

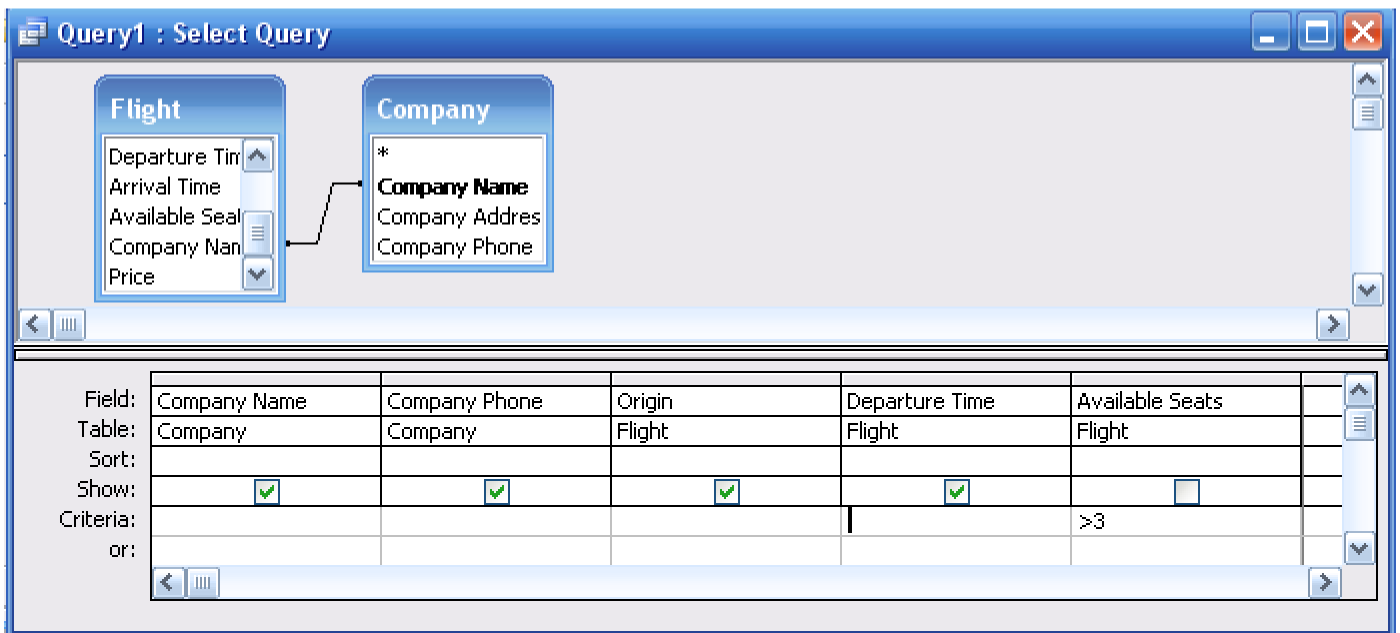
Structured Query Language

SELECT Company.CompanyName, Company.CompanyPhone, Flight.Origin, Flight.DepartureTime

FROM Flight,Company

WHERE Flight.CompanyName=Company.CompanyName

AND Flight.AvailableSeats>3;



field

```
select address
from employee
where employee.surname='Smith' and
employee.forenames='Robert';
```

table

how you want to restrict the rows

field

tables to join

select dname
from employee, department
where employee.depno=department.depno
and surname='Smith' and
forenames='Robert';

how to join

how you want to restrict the rows

Create a MySQL Database

- “root” user creates database + grants permissions
  - Using the WAMP console (or mysql –u root –p)
    - root has no initial password; just hit <enter> when asked
  - By the system administrator on OTAL (otal.umd.edu)

```
CREATE DATABASE project;
GRANT SELECT, INSERT, UPDATE, DELETE, INDEX, ALTER, CREATE, DROP ON project.* TO ‘foo’@’localhost’ IDENTIFIED BY ‘bar’;
FLUSH PRIVILEGES;
```

- Start mysql
  - MySQL console for WAMP, ssh for OTAL

```
mysql –u foo –p bar
```

- Connect to your database

```
USE project;
```

Creating Tables

```
CREATE TABLE contacts (
  ckey      MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
  id        MEDIUMINT UNSIGNED NOT NULL,
  ctype     SMALLINT UNSIGNED NOT NULL,
  cstring   VARCHAR(40) NOT NULL,
  FOREIGN KEY (id) REFERENCES persons(id) ON DELETE CASCADE,
  FOREIGN KEY (ctype) REFERENCES ctlabels(ctype) ON DELETE RESTRICT,
  PRIMARY KEY (ckey)
) ENGINE=INNODB;
```

- To delete: DROP TABLE contacts;

Populating Tables

```
INSERT INTO ctlabels
  (string) VALUES
  ('primary email'),
  ('alternate email'),
  ('home phone'),
  ('cell phone'),
  ('work phone'),
  ('AOL IM'),
  ('Yahoo Chat'),
  ('MSN Messenger'),
  ('other');
```

- To empty a table: DELETE FROM ctlabels;

“Looking Around” in MySQL

- SHOW DATABASES;
- SHOW TABLES;
- DESCRIBE tablename;
- SELECT * FROM tablename;

SQL Utility Service Desk Exercise

- Design a database to keep track of service calls for a utility company:
  - Customers call to report problems
  - Call center manages “tickets” to assign workers to jobs
    - Must match skills and service location
    - Must balance number of assignments
  - Workers call in to ask where their next jobs are
- In SQL, you can do the following operations:
  - Count the number of rows in a result set
  - Sort the result set according to a field
  - Find the maximum and minimum value of a field

Before You Go

On a sheet of paper, answer the following (ungraded) question (no names, please):

What was the muddiest point in today's class?