INFM 603: Information Technology and Organizational Context

Session 6: Relational Databases

Jimmy Lin
The iSchool
University of Maryland

Thursday, October 11, 2012

Databases Yesterday…

Databases Today...

What’s structured information?

It’s what you put in a database

What’s a database?

It’s what you store structured information in

So what's a database?

An integrated collection of data organized according to some model…

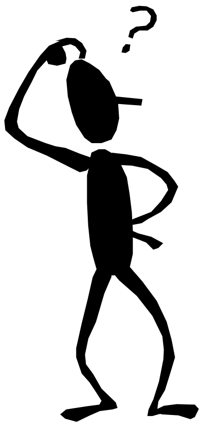
So what's a relational database?

An integrated collection of data organized according to a relational model

Database Management System (DBMS)

Software system designed to store, manage, and facilitate access to databases

Databases (try to) model reality…



- Entities: things in the world
  - Example: airlines, tickets, passengers
- Relationships: how different things are related
  - Example: the tickets each passenger bought
- "Business Logic": rules about the world
  - Example: fare rules

Relational Databases

Source: Microsoft Office Clip Art

Components of a Relational Database

- Field: an "atomic" unit of data
- Record: a collection of related fields
  - Sometimes called a "tuple"
- Table: a collection of related records
  - Each record is a row in the table
  - Each field is a column in the table
- Database: a collection of tables

A Simple Example

Field Name

Table

Name	**DOB**	**SSN**
John Doe	04/15/1970	153-78-9082
Jane Smith	08/31/1985	768-91-2376
Mary Adams	11/05/1972	891-13-3057

Record

Field

Primary Key

Why "Relational"?

- View of the world in terms of entities and relations:
  - Tables represent "relations"
  - Each row (record, tuple) is "about" an entity
  - Fields can be interpreted as "attributes" or "properties" of the entity
- Data is manipulated by "relational algebra":
  - Defines things you can do with tuples
  - Expressed in SQL

The Registrar Example

- What do we need to know?
  - Something about the students
    (e.g., first name, last name, email, department)
  - Something about the courses
    (e.g., course ID, description, enrolled students, grades)
  - Which students are in which courses
- How do we capture these things?

A First Try

Student ID	Last Name	First Name	Dept ID	Dept	Course ID	Course name	Grade	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

Why is this a bad idea?

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

Another Try...

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	Electrical Engineering
HIST	History
CLIS	Information Studies

Course Table

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

Enrollment Table

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

Keys

- "Primary Key" uniquely identifies a record
  - e.g., student ID in the student table
- "Foreign Key" is primary key in the other table
  - It need not be unique in this table

Approaches to Normalization

- For simple problems:
  - Start with the entities you're trying to model
  - Group together fields that "belong together"
  - Add keys where necessary to connect entities in different tables
- For more complicated problems:
  - Entity-relationship modeling

The Data Model

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	Electrical Engineering
HIST	History
CLIS	Information Studies

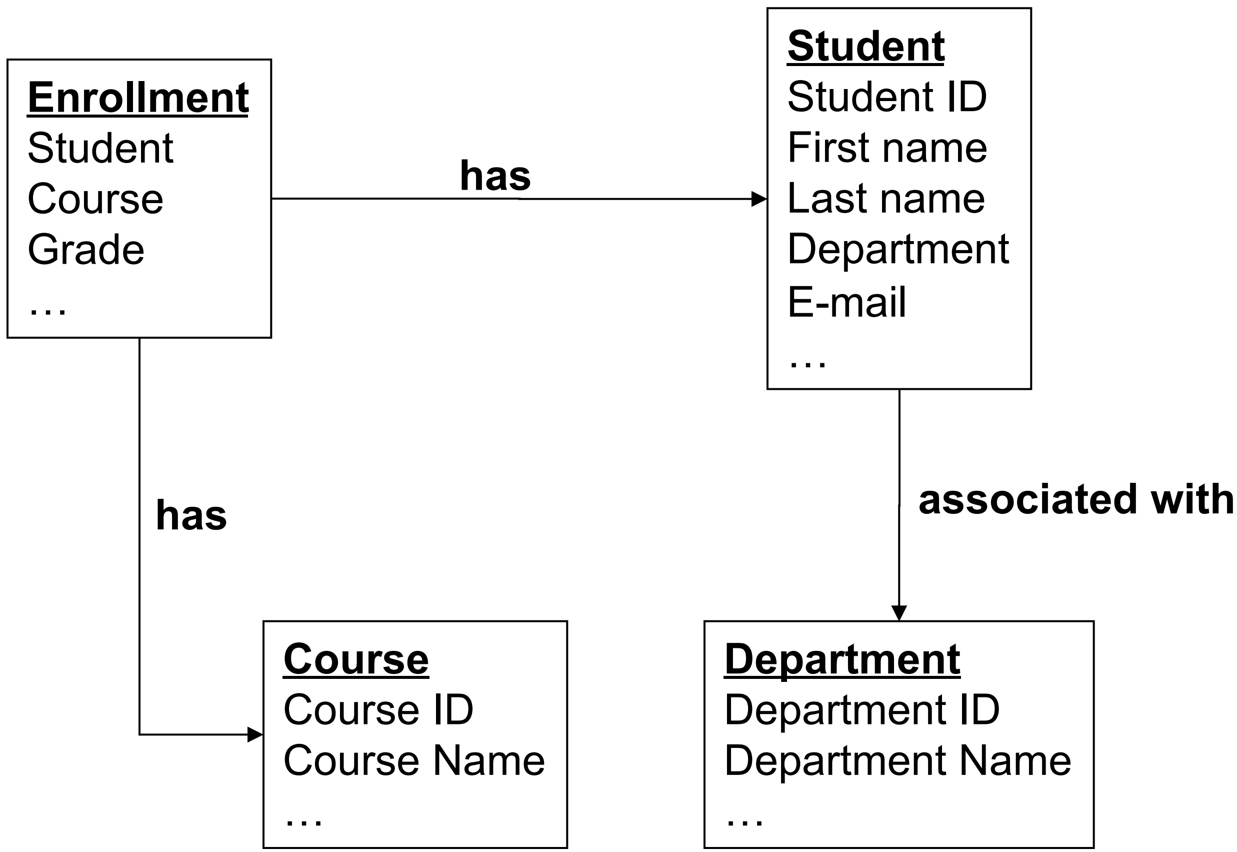
Course Table

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

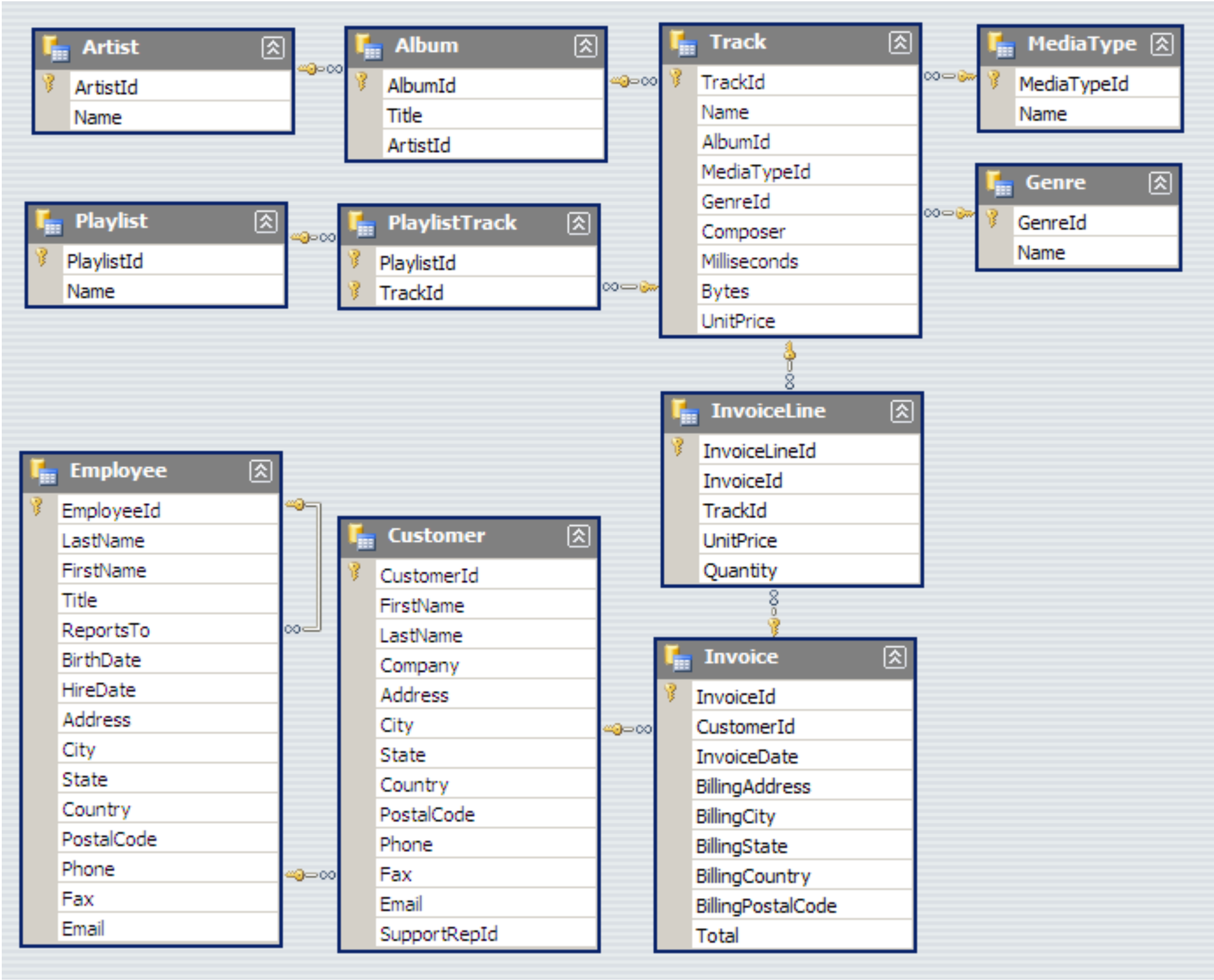
Enrollment Table

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

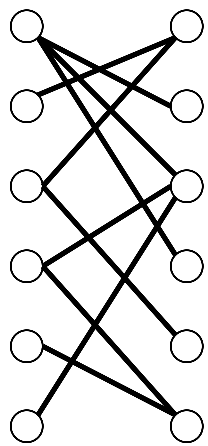
Registrar ER Diagram



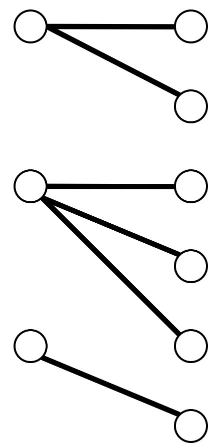
A Real Example



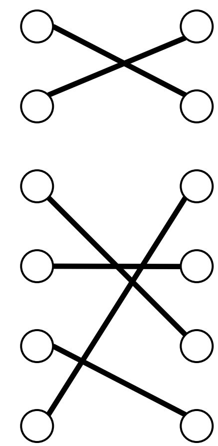
Types of Relationships



Many-to-Many



One-to-Many



One-to-One

Database Integrity

- Registrar database must be internally consistent
  - All enrolled students must have an entry in the student table
  - All 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 be true of the database at any time
  - Specified when the database is designed
  - Checked when the database is modified
- RDBMS ensures that integrity constraints are always kept
  - So that database contents remain faithful to the real world
  - Helps avoid data entry errors
- Where do integrity constraints come from?

SQL

(Don’t Panic!)

Select

Student ID	Last Name	First Name	Dept ID	Department	email
1	Arrows	John	EE	Electrical 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	Electrical Engineering
2	History
3	History
4	Information Stuides

Where

Student ID	Last Name	First Name	Dept ID	Department	email
1	Arrows	John	EE	Electrical 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

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	Electrical Engineering
HIST	History
CLIS	Information Studies

"Joined" Table

Student ID	Last Name	First Name	Dept ID	Department	email
1	Arrows	John	EE	Electrical 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

Relational Operations

- Joining tables: JOIN
- Choosing columns: SELECT
  - Based on their labels (field names)
  - * is a shorthand for saying “all fields”
- Choosing rows: WHERE
  - Based on their contents

    department ID = “HIST”
- These can be specified together

    select Student ID, Dept where Dept = “History”

Use this template!

select [columns in the table]

from [table name]

join [another table name] on [join criterion]

join [another table name] on [join criterion]

…

where [selection criteria]

SQL Tips and Tricks

- Referring to fields (in SELECT statements)
  - Use TableName.FieldName
  - Can drop TableName if FieldName is unambiguous
- Join criterion
  - Most of the time, based on primary key – foreign key relationships
    e.g., Table1.PrimaryKey = Table2.ForeignKey
- Selection criteria
  - Use = instead of ==

Aggregations

- SQL aggregation functions
  - Examples: count, min, max, sum, avg
  - Use in select statements

    select count(*)…
    select min(price)…
    select sum(length)…

  - Tip: when trying to write SQL query with aggregation, do it first without
- Group by [field]
  - Often used in conjunction with aggregation
  - Conceptually, breaks table apart based on the [field]

How do you want your results served?

- Order by [field name]
  - Does exactly what you think it does!
  - Either “asc” or “desc”
- Limit *n*
  - Returns only *n* records
  - Useful to retrieving the top *n* or bottom *n*

So how’s a database more than a spreadsheet?

Database in the “Real World”

- Typical database applications:
  - Banking (e.g., saving/checking accounts)
  - Trading (e.g., stocks)
  - Traveling (e.g., airline reservations)
  - Social media (e.g., Facebook)
  - …
- Characteristics:
  - Lots of data
  - Lots of concurrent operations
  - Must be fast
  - “Mission critical” (well… sometimes)

Operational Requirements

- Must hold a lot of data
- Must be reliable
- Must be fast
- Must support concurrent operations

Must hold a lot of data

Solution: Use lots of machines

(Each machine holds a small slice)

So which machine has your copy?

Must be reliable

Solution: Use lots of machines

(Store multiple copies)

But which copy is the right one?

How do you keep the copies in sync?

Must be fast

Solution: Use lots of machines

(Share the load)

How do you spread the load?

Must support concurrent operations

Solution: this is hard!

(But fortunately doesn't matter
for many applications)

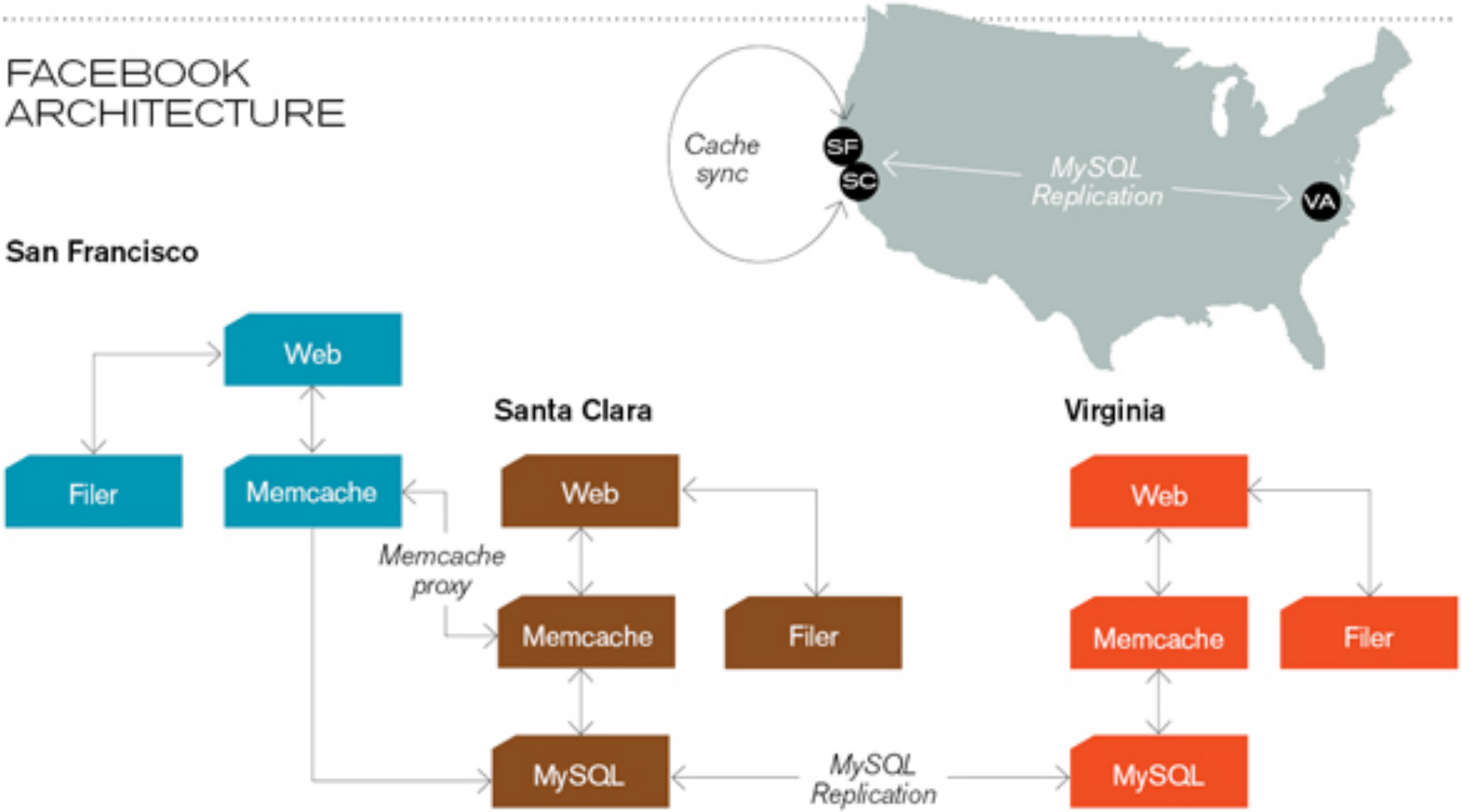
Database Transactions

- Transaction = sequence of database actions grouped together
  - e.g., transfer $500 from checking to savings
- ACID properties:
  - Atomicity: all-or-nothing
  - Consistency: each transaction yield a consistent state
  - 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
  - Trickier than it sounds!



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)

Now you know...

Wait, but these are websites?