

CSE 544

Principles of Database Management Systems

Magdalena Balazinska (magda)

Winter 2015

Lecture 1 - Class Introduction and
Review of Relational Model

Outline

- Introductions
- Class overview
- What is the point of a db management system (DBMS)?
- Review of relational model

Course Staff

- **Instructor: Magda** (`magda@cs.washington.edu`)
 - Office hours Thursday 4:30pm-5:20pm
 - Location: CSE 584
- **TA: Jingjing Wang** (`jwang@cs.washington.edu`)
 - Graduate student in the database group
 - Office hours and location: to be announced on course website
- **TA: Jennifer Ortiz** (`jortiz16@cs.washington.edu`)
 - Graduate student in the database group
 - Office hours and location: to be announced on course website

Who is Magda?

- Associate Professor. At UW since January 2006
- PhD from MIT, February 2006

- Area of interest: database management systems
- Current research focus
 - Efficiency and usability of database systems
 - Topics of interest
 - Cloud computing
 - Big data management
 - Scientific data management
 - Pricing and data management

Goals of the Class/Class Content

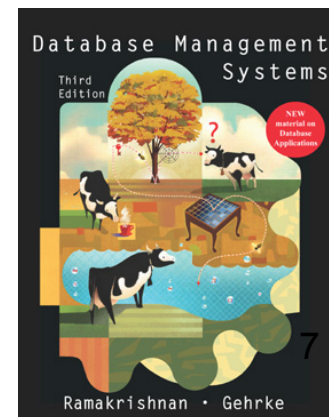
- Study principles of data management
 - Data models, data independence, declarative query language, etc.
- Study key DBMS design issues
 - Storage, query execution and optimization, transactions
 - Parallel data processing, data warehousing, streaming data, etc.
- Study some current, hot research topics
- Ensure that
 - You are comfortable using a DBMS locally and in the cloud
 - You have an idea about how to build a DBMS
 - You know a bit about current research topics in data management
 - You get to explore in depth a data management problem (project)

Class Format

- Two lectures per week: MW @ 1:30pm
- Mix of lecture and discussion
 - Mostly based on papers
 - Must read papers before lecture and submit a paper review
 - Come prepared to discuss the papers assigned for the class
 - Class participation counts for a non-negligible part of your grade

Readings and Notes

- Readings are based on papers
 - Mix of old seminal papers and new papers
 - **Papers available online on class website**
 - Some come from the “red book” [no need to get it]
 - Three types of readings
 - Mandatory, additional resources, and optional
- Background readings from the following book
 - Database Management Systems. **Third Ed.** Ramakrishnan and Gehrke. McGraw-Hill. [recommended]
- Lecture notes (the pptx slides)
 - Posted on class website after each lecture



Class Resources

- Website: lectures, assignments, projects
<http://www.cs.washington.edu/544>
List of all the **deadlines**
- Mailing list on course website: Make sure you register
 - Your @uw.edu email address is already on the list
- Discussion board: Discuss assignments, papers, other

Evaluation

- **Class participation & paper reviews 20%**
 - Paper reviews are due by the beginning of each lecture
 - Reading questions are posted on class website
 - You need to speak up in class from time to time
- **Assignments 35%: **Individual****
 - HW1: Using a DBMS / data analysis pipeline
 - HW2: Building a simple DBMS
 - HW3: Analyzing a large dataset with a cloud DBMS
- **Project 45%: Groups of up to three students**
 - Small research or engineering. Start to think about it now!

Class Participation

- An important part of your grade
- Because
 - We want you to read & think about papers throughout quarter
 - Important to learn to discuss papers
- Expectations
 - Ask questions, raise issues, think critically
 - Learn to express your opinion
 - Respect other people's opinions

Paper reviews

- Between 1/2 page and 1 page in length
 - Summary of the main points of the paper
 - Critical discussion of the paper
- Reading questions
 - For some papers, we will **post reading questions** to help you figure out what to focus on when reading the paper
 - Please address these questions in your reviews
- Grading: credit/no-credit
 - You can skip one review without penalty
 - **MUST** submit review **BEFORE** lecture
 - Individual assignments (but feel free to discuss paper with others)

Assignments

- **Goals:**
 - Hands-on experience using a DBMS, building a simple DBMS, and using a cloud DBMS for data analysis
- **HW1:** Use a DBMS
- **HW2:** Build a simple DBMS
- **HW3:** Large-scale data analysis with a cloud DBMS
- See course calendar for deadlines
- HW1 will be posted on Monday
- We will accept late assignments with valid excuse

Project Overview

- Topic
 - Choose from a list of mini-research topics
 - Or come up with your own
 - Can be related to your ongoing research
 - Can be related to a project in another course
 - Must be related to databases
 - Must involve either research or significant engineering
 - Open ended
- **Final deliverables**
 - Short conference-style paper (6 pages)
 - Conference-style presentation or posters depending on nb groups

Amazon AWS
credits available!

Project Goals

- Study **in depth** a data management problem that you find interesting
 - Understand and model the problem
 - Research and understand related work (no need to be exhaustive)
 - Propose some new approach or some interesting eng. problem
 - Implement some parts
 - Evaluate your solution
 - Write-up and present your results
- Amount of work may vary widely between groups

Project Milestones

- Dates are posted on course website
 - Project proposal
 - Milestone report
 - Final report
 - Final presentation
- More details on the website, including ideas & examples
- We will provide feedback throughout the quarter

Let's get started

- What is a database?
- Give examples of databases

Let's get started

- What is a database?
 - A collection of files storing related data
- Give examples of databases
 - Accounts database; payroll database; UW's students database; Amazon's products database; airline reservation database

Data Management

- Data is valuable but hard and costly to manage
- Example: Store database
 - **Entities:** employees, positions (ceo, manager, cashier), stores, products, sells, customers.
 - **Relationships:** employee positions, staff of each store, inventory of each store.
- What operations do we want to perform on this data?
- What functionality do we need to manage this data?

Required Functionality

1. Describe real-world entities in terms of stored data
2. Create & persistently store large datasets
3. Efficiently query & update
 1. Must handle complex questions about data
 2. Must handle sophisticated updates
 3. Performance matters
4. Change structure (e.g., add attributes)
5. Concurrency control: enable simultaneous updates
6. Crash recovery
7. Access control, security, integrity

Difficult and costly to implement all these features

Database Management System

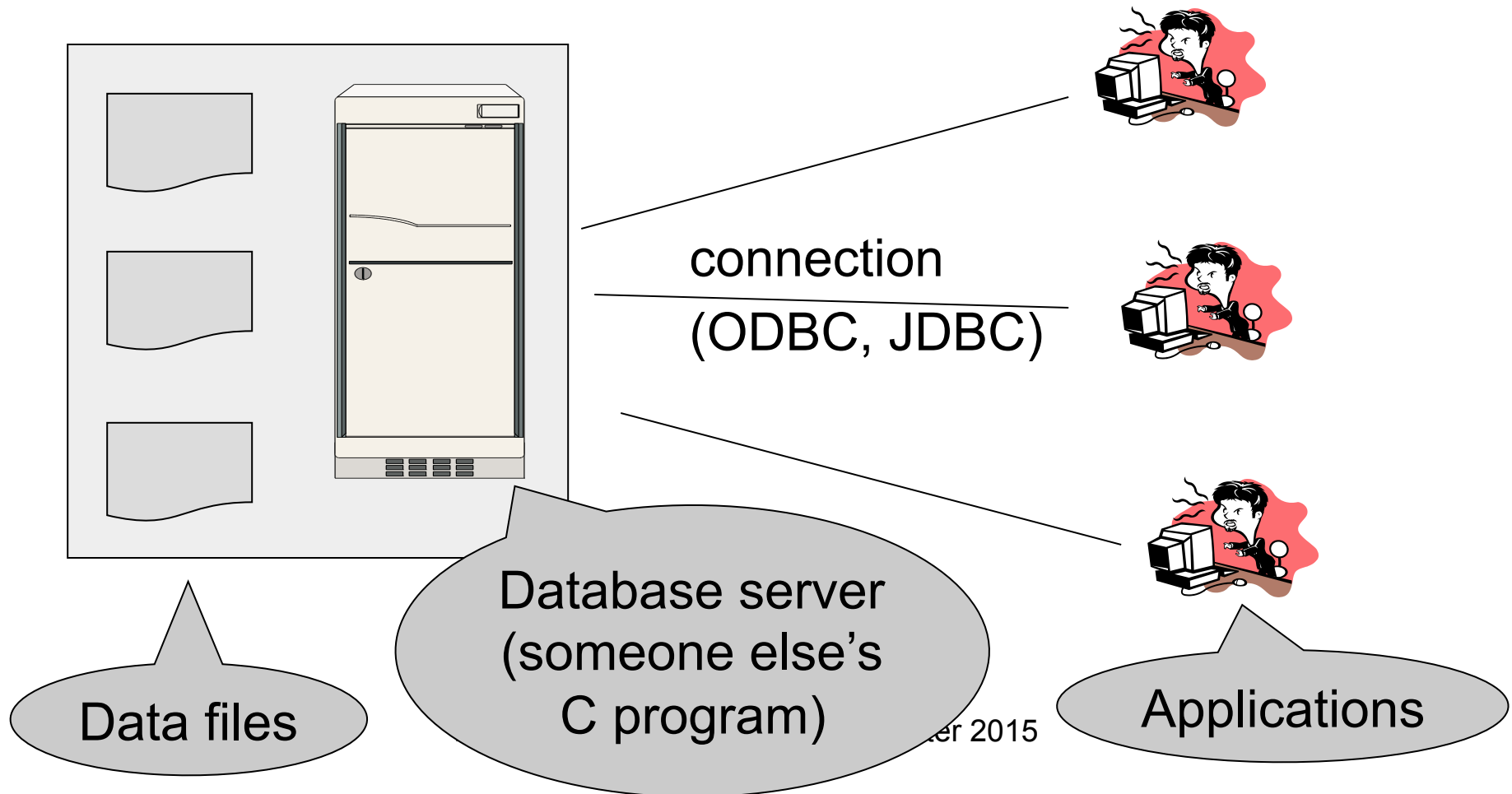
- A DBMS is a software system designed to provide data management services
- Examples of DBMS
 - Oracle, DB2 (IBM), SQL Server (Microsoft),
 - PostgreSQL, MySQL,...

Market Shares

- From 2006 Gartner report:
 - IBM: 21% market with \$3.2BN in sales
 - Oracle: 47% market with \$7.1BN in sales
 - Microsoft: 17% market with \$2.6BN in sales

Typical System Architecture

“Two tier system” or “client-server”



Main DBMS Features

- Data independence
 - Data model
 - Data definition language
 - Data manipulation language
- Efficient data access
- Data integrity and security
- Data administration
- Concurrency control
- Crash recovery
- Reduced application development time

How to decide what features should go into the DBMS?

When not to use a DBMS?

- DBMS is optimized for a certain workload
- Some applications may need
 - A completely different data model
 - Completely different operations
 - A few time-critical operations
- Example
 - Highly optimized scientific simulations

Outline

- Introductions
- Class overview
- What is the point of a db management system (DBMS)?
- Review of relational model

Relation Definition

- **Database is collection of relations**
- **Relation R is subset of $S_1 \times S_2 \times \dots \times S_n$**
 - Where S_i is the domain of attribute i
 - n is number of attributes of the relation
- Relation is basically a table with rows & columns
 - SQL uses the term “table” to refer to a relation

Properties of a Relation

- Each row represents an n-tuple of R
- Ordering of rows is immaterial
- All rows are distinct
- Ordering of columns is significant
 - Because two columns can have same domain
 - But columns are labeled so
 - Applications need not worry about order
 - They can simply use the names
- Domain of each column is a primitive type
- Relation consists of a **relation schema** and **instance**

More Definitions

- **Relation schema**: describes column heads
 - Relation name
 - Name of each field (or column, or attribute)
 - Domain of each field
- **Degree (or arity) of relation**: nb attributes
- **Database schema**: set of all relation schemas

Even More Definitions

- **Relation instance**: concrete table content
 - Set of tuples (also called records) matching the schema
- **Cardinality of relation instance**: nb tuples
- **Database instance**: set of all relation instances

Example

- Relation schema

Supplier(sno: integer, sname: string, scity: string, sstate: string)

- Relation instance

sno	sname	scity	sstate
1	s1	city 1	WA
2	s2	city 1	WA
3	s3	city 2	MA
4	s4	city 2	MA

Integrity Constraints

- **Integrity constraint**
 - Condition specified on a database schema
 - Restricts data that can be stored in db instance
- DBMS enforces integrity constraints
 - Ensures only legal database instances exist
- Simplest form of constraint is domain constraint
 - Attribute values must come from attribute domain

Key Constraints

- **Key constraint:** “certain minimal subset of fields is a unique identifier for a tuple”
- **Candidate key**
 - Minimal set of fields
 - That uniquely identify each tuple in a relation
- **Primary key**
 - One candidate key can be selected as primary key

Foreign Key Constraints

- A relation can refer to a tuple in another relation
- **Foreign key**
 - Field that refers to tuples in another relation
 - Typically, this field refers to the primary key of other relation
 - Can pick another field as well

Key Constraint SQL Examples

```
CREATE TABLE Part (  
    pno integer,  
    pname varchar(20),  
    psize integer,  
    pcolor varchar(20),  
    PRIMARY KEY (pno)  
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(  
    sno integer,  
    pno integer,  
    qty integer,  
    price integer  
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(  
    sno integer,  
    pno integer,  
    qty integer,  
    price integer,  
    PRIMARY KEY (sno,pno)  
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(  
    sno integer,  
    pno integer,  
    qty integer,  
    price integer,  
    PRIMARY KEY (sno,pno) ,  
    FOREIGN KEY (sno) REFERENCES Supplier,  
    FOREIGN KEY (pno) REFERENCES Part  
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(  
    sno integer,  
    pno integer,  
    qty integer,  
    price integer,  
    PRIMARY KEY (sno,pno) ,  
    FOREIGN KEY (sno) REFERENCES Supplier  
        ON DELETE NO ACTION,  
    FOREIGN KEY (pno) REFERENCES Part  
        ON DELETE CASCADE  
);
```

General Constraints

- Table constraints serve to express complex constraints over a single table

```
CREATE TABLE Part (  
    pno integer,  
    pname varchar(20),  
    psize integer,  
    pcolor varchar(20),  
    PRIMARY KEY (pno),  
    CHECK ( psize > 0 )  
);
```

- It is also possible to create constraints over many tables

Relational Queries

- **Query inputs and outputs are relations**
- Query evaluation
 - Input: instances of input relations
 - Output: instance of output relation

Relational Algebra

- **Query language** associated with relational model
- **Queries specified in an operational manner**
 - A query gives a step-by-step procedure
- **Relational operators**
 - Take one or two relation instances as argument
 - Return one relation instance as result
 - Easy to **compose** into **relational algebra expressions**

Relational Operators

- **Selection**: $\sigma_{\text{condition}}(S)$
 - Condition is Boolean combination (\wedge, \vee) of terms
 - Term is: attr. op constant, attr. op attr.
 - Op is: $<$, \leq , $=$, \neq , \geq , or $>$
- **Projection**: $\pi_{\text{list-of-attributes}}(S)$
- **Union** (\cup), **Intersection** (\cap), **Set difference** ($-$),
- **Cross-product** or **cartesian product** (\times)
- **Join**: $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$
- **Division**: R/S , **Rename** $\rho(R(F), E)$

Selection & Projection Examples

Patient

no	name	zip	disease
1	p1	98125	flu
2	p2	98125	heart
3	p3	98120	lung
4	p4	98120	heart

$\pi_{\text{zip,disease}}(\text{Patient})$

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

$\sigma_{\text{disease}='heart'}(\text{Patient})$

no	name	zip	disease
2	p2	98125	heart
4	p4	98120	heart

$\pi_{\text{zip}}(\sigma_{\text{disease}='heart'}(\text{Patient}))$

zip
98120
98125

Relational Operators

- **Selection**: $\sigma_{\text{condition}}(S)$
 - Condition is Boolean combination (\wedge, \vee) of terms
 - Term is: attr. op constant, attr. op attr.
 - Op is: $<$, \leq , $=$, \neq , \geq , or $>$
- **Projection**: $\pi_{\text{list-of-attributes}}(S)$
- **Union** (\cup), **Intersection** (\cap), **Set difference** ($-$),
- **Cross-product** or **cartesian product** (\times)
- **Join**: $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$
- **Division**: R/S , **Rename** $\rho(R(F), E)$

Cross-Product Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

P x V

P.age	P.zip	disease	name	V.age	V.zip
54	98125	heart	p1	54	98125
54	98125	heart	p2	20	98120
20	98120	flu	p1	54	98125
20	98120	flu	p2	20	98120

Different Types of Join

- **Theta-join:** $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$
 - Join of R and S with a join condition θ
 - Cross-product followed by selection θ
- **Equijoin:** $R \bowtie_{\theta} S = \pi_A (\sigma_{\theta}(R \times S))$
 - Join condition θ consists only of equalities
 - Projection π_A drops all redundant attributes
- **Natural join:** $R \bowtie S = \pi_A (\sigma_{\theta}(R \times S))$
 - Equijoin
 - Equality on **all** fields with same name in R and in S

Theta-Join Example

AnonPatient P

age	zip	disease
50	98125	heart
19	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie_{P.zip = V.zip \text{ and } P.age \leq V.age + 1 \text{ and } P.age \geq V.age - 1} V$

P.age	P.zip	disease	name	V.age	V.zip
19	98120	flu	p2	20	98120

Equijoin Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie_{P.age=V.age} V$

age	P.zip	disease	name	V.zip
54	98125	heart	p1	98125
20	98120	flu	p2	98120

Natural Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie V$

age	zip	disease	name
54	98125	heart	p1
20	98120	flu	p2

More Joins

- **Outer join**
 - Include tuples with no matches in the output
 - Use NULL values for missing attributes
- Variants
 - Left outer join
 - Right outer join
 - Full outer join

Outer Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu
33	98120	lung

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie V$

age	zip	disease	name
54	98125	heart	p1
20	98120	flu	p2
33	98120	lung	null

Example of Algebra Queries

Q1: Names of patients who have heart disease

$\pi_{\text{name}}(\text{Voter} \bowtie (\sigma_{\text{disease}=\text{'heart'}}(\text{AnonPatient})))$

More Examples

Relations

Supplier(sno, sname, scity, sstate)

Part(pno, pname, psize, pcolor)

Supply(sno, pno, qty, price)

Q2: Name of supplier of parts with size greater than 10

$\pi_{\text{sname}}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{\text{psize} > 10}(\text{Part})))$

Q3: Name of supplier of red parts or parts with size greater than 10

$\pi_{\text{sname}}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{\text{psize} > 10}(\text{Part}) \cup \sigma_{\text{pcolor} = \text{'red'}}(\text{Part})))$

(Many more examples in the book)

Logical Query Plans

An RA expression but represented as a tree

