CSE 544
Principles of Database Management Systems

Fall 2016

Lecture 1 - Introduction and the Relational Model

Outline

- Introduction
- Class overview
- Why database management systems (DBMS)?
- The relational model

Course Staff

- Instructor: Dan Suciu
  - Office hours: Wednesday 3:30pm-4:20pm (or by appointment)
  - Location: CSE 662
- TA: Shrainik Jain
  - Graduate student in the database group
  - Office hours and location: Fridays 1:30-2:20, CSE 218
- Use cse544-staff@cs.washington.edu to reach us

About Me

- PhD from UPenn
- Bell Labs / AT&T Labs
- UW (since 2000)

- I like to combine theory with database systems:
  - New data models and their theory: semistructured data model
  - Data privacy ("does a view leak anything about a query?")
  - Probabilistic databases (next week: workshop at Simon's)
  - Optimal query processing (non-traditional algorithms)
  - Data pricing ("How much $$ for select * from R join S join … where ...")

Goals of the Class/Class Content

- Study principles of data management
  - Data models, data independence, declarative query language.
- Study key Database Mgmt Systems (DBMS) design issues
  - Storage, query execution and optimization, transactions
  - Parallel data processing, column-oriented db etc.
- Theory of data management
  - Query equivalence, optimal query processing
- 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)

A Note for Non-Majors

- For the Data Science option it is recommended that you take 414
- For the Advanced Data Science option you need to take 544

- 544 is an ***advanced*** class, intended as an introduction to data management research
- Does poor job at covering fundamentals systematically, yet there is a midterm testing those fundamentals

- Unsure? Look at the short quiz on the website.

Class Format

- Two lectures per week: Tues & Thurs @ 11am
- Mix of lecture and discussion

Class Topics

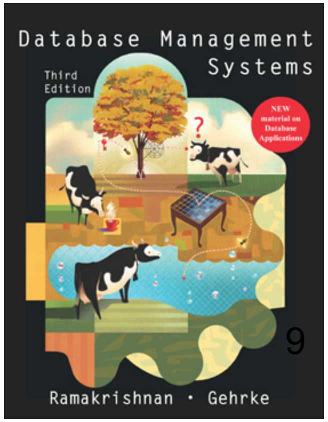
- Fundamentals
  - Query language (SQL, RA, datalog)
  - Data models
- Systems:
  - Query processing and optimization
- Theory:
  - Query equivalence
  - Yannakakis, FHTW, generalized distributivity law
- Parallel data analytics
  - MapReduce and Spark
  - Basic and not-so-basic parallel query processing
- Transactions

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

- Background readings from the following book
  - Database Management Systems. **Third Ed**. Ramakrishnan and Gehrke. McGraw-Hill. [recommended]

- Lecture notes (the 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, weather, stock, etc
  – **HW: Introduce yourself to everyone by posting a new message on discussion board**

Evaluation

- Assignments 30%
- Midterm 30%
- Project 30%
- Paper reviews + class participation 10%

Assignments – 30%

- **HW1**: Use a DBMS (posted!) -- due Monday 10/17
- **HW2**: Build a simple DBMS – due Friday 11/4
- **HW3**: Data analysis in the cloud– due Friday 11/18

- See course calendar for deadlines
- We will accept late assignments with ***very*** valid excuse

Midterm – 30%

- Tuesday, 11/8, in class

Project – 30%

- 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 / data management
  - Must involve either research or significant engineering
  - Open ended

Amazon AWS credits available!

- Final deliverables
  - Short conference-style paper (6 pages)
  - Conference-style presentation or posters depending on nb groups

Project – 30%

- Dates will be posted on course website
  - **M1**: form groups — Monday 10/10
  - **M2**: Project proposal — Monday 10/19
  - **M3**: Milestone report — Wednesday 11/16
  - **M4**: Poster presentation — Tuesday 12/13
  - **M5**: Project paper — Thursday 12/15

- More details will be on the website, including ideas & examples

- We will provide feedback throughout the quarter

Paper reviews – 10%

- Between 1/2 page and 1 page in length
  - Summary of the main points of the paper
  - Critical discussion of the paper
  - Guidelines on course website
  - First review due Sunday, 10/9 (to be posted on the website)
- 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
  - **MUST submit review 12 HOURS BEFORE lecture**
  - Individual assignments (but feel free to discuss paper with others)

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

Now onward to the world 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
  - Your ORCA card transactions, Facebook friends graph, past tweets, etc

Data Management

- Data is valuable but hard and costly to manage

- Example: database for a store
  - **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

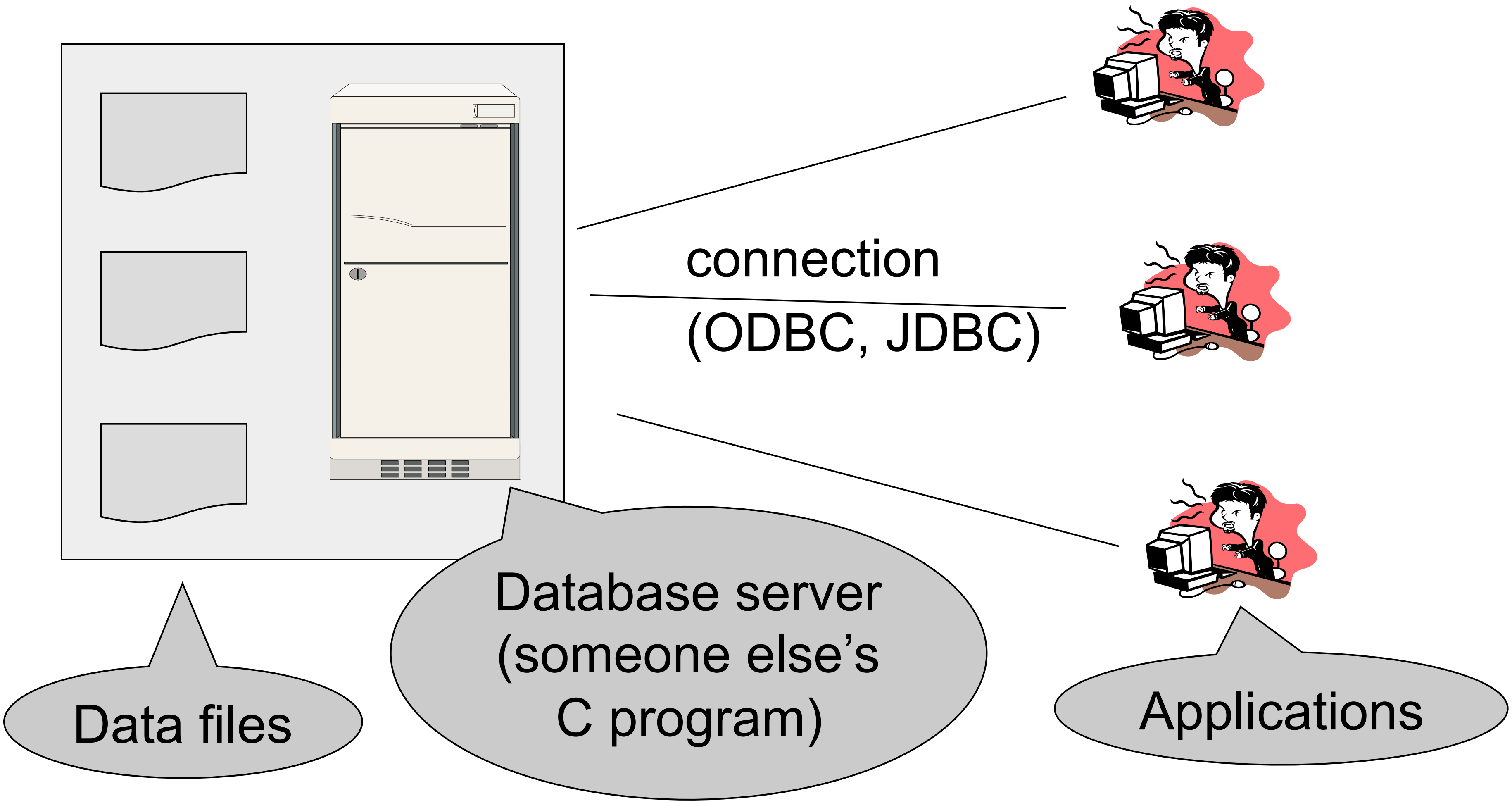
Database Management System

- A DBMS is a software system designed to provide data management services

- Examples of DBMS
  – Oracle, DB2 (IBM), SQL Server (Microsoft),
  – PostgreSQL, MySQL,…

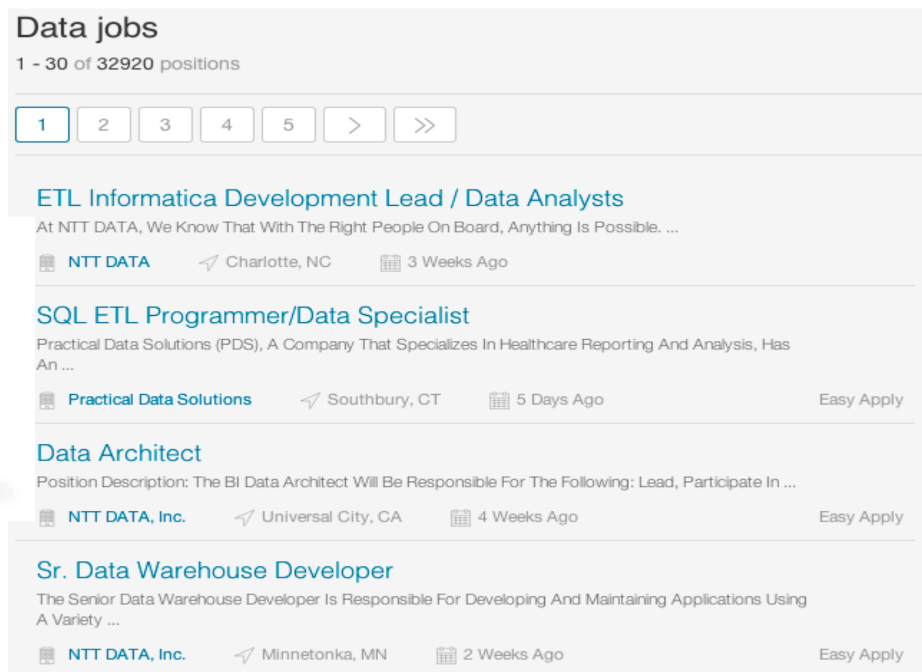
Typical System Architecture

"Two tier system" or "client-server"



Why should you care?

- 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
- Rise of big data



Why should *you* care?

- Most of CS today is data driven
- Your research will involve some data component – need to know how to use a DBMS
- Your research may involve some innovative data management solution – need to be up to date with what is known, beyond a DBMS

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
- Why database management systems (DBMS)?
- The relational model

Data Model

- An abstract mathematical concepts that defines the data
- Data models:
  - Relational (this course)
  - Semistructured (XML, JSon, Protobuf)
  - Graph data model
  - Object-Relational data model

Relation Definition

- **Database is collection of relations**

- Relation is a table with rows & columns
  - SQL uses the term “table” to refer to a relation

- **Relation R is subset of S_1 x S_2 x … x S_n**
  - Where $\mathbf{S_i}$ is the domain of attribute **i**
  - **n** is number of attributes of the relation

Example

- Relation schema

  Supplier(<u>sno: integer</u>, 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

sno is called a key
(what does it mean?)

Discussion of the Relational Model

- [This slide appeals to your background in DB; will may or may not discuss these concepts later in class]

- Relations are *flat* = called 1st Normal Form

- A relation may have a key, but no other FD's = either 3rd Normal form, or Boyce Codd Normal Form (BCNF) depending on some subtle details

[discuss on the white board]

Other Models: Semistructured

- E.g. you will encounter this in HW1:

```
<article mdate="2011-01-11" key="journals/acta/GoodmanS83">
   <author>Nathan Goodman</author>
   <author>Oded Shmueli</author>
   <title>NP-complete Problems Simplified on Tree Schemas.</title>
   <pages>171-178</pages>
   <year>1983</year>
   <volume>20</volume>
   <journal>Acta Inf.</journal>
   <url>db/journals/acta/acta20.html#GoodmanS83</url>
   <ee>http://dx.doi.org/10.1007/BF00289414</ee>
</article>
```

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: <, <=, =, ≠, >=, 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: $\sigma_{condition}(S)$
  - Condition is Boolean combination ($\wedge$,$\vee$) of terms
  - Term is: attr. op constant, attr. op attr.
  - Op is: $<$, $<=$, $=$, $\neq$, $>=$, or $>$
- Projection: $\pi_{list\text{-}of\text{-}attributes}(S)$
- 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

Join Galore

- **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**: $R \bowtie S = \pi_A (\sigma_\theta(R \times S))$
  - aka 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

$P \overset{o}{\bowtie} V$

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

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

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

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

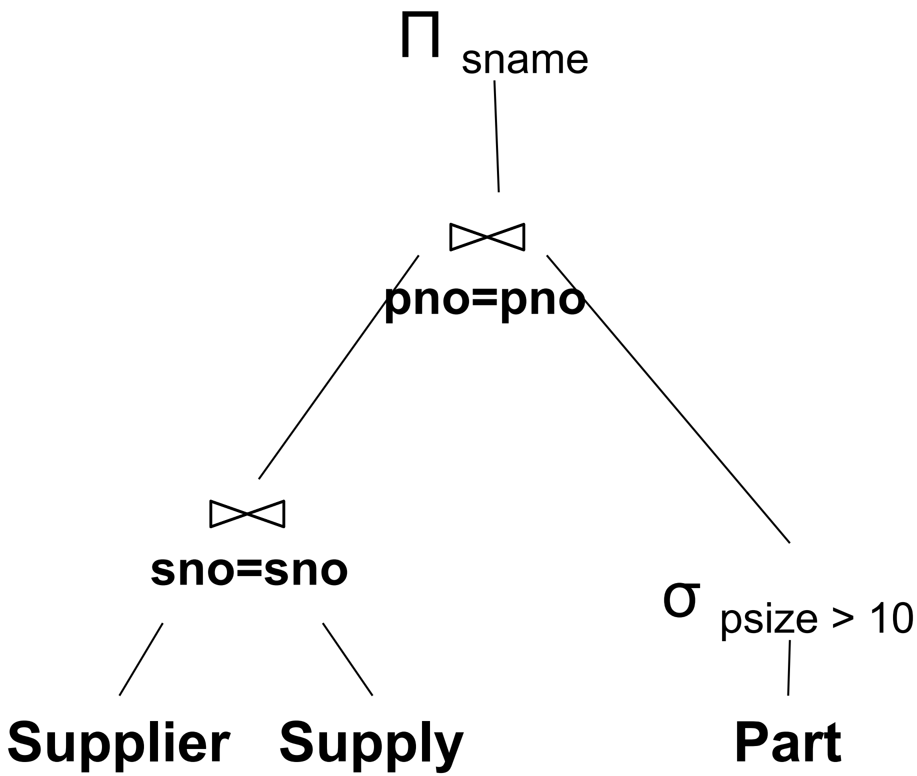
Q3: Name of supplier of red parts or parts with size greater than 10

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

(Many more examples in the R&G)

Logical Query Plans

An RA expression but represented as a tree



Next time:
How to evaluate queries