Database Management Systems
CS430 & CS630

Umass Boston
Summer 2023
Cristina Maier

Part of slides are based on “Database Management System, 3rd ed., by Ramakrishnan and Gehrke

CS430 & CS630

- Instructor: Cristina Maier
  - Contact: cristina.maier AT umb DOT edu
- **Class website**: https://www.cs.umb.edu/~cmaier/cs430cs630
  - *Please check this website on a regular basis for updates!!!*
- If you send me an email, subject line has to begin with **[CS430]** or **[CS630]**

Class Attendance

- Class attendance is required
- All subjects presented during the class sessions could appear in the exams and homeworks/assignments, even if they are not in the slides, textbook or other material.

Class info

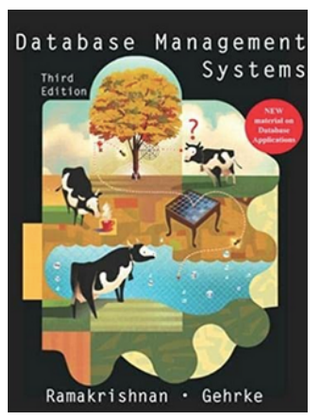
- One or two exams remote.
- Homeworks
- Possible quizzes
- Lectures attendance is required

Passing Threshold

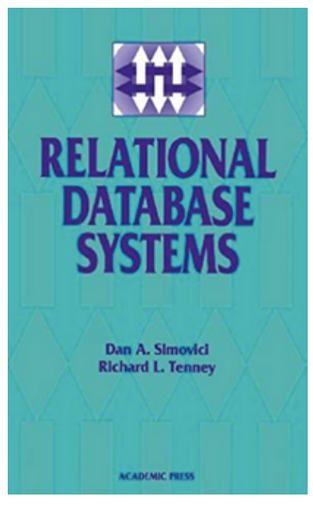
- Undergraduate: min 60% to pass
- Graduate: min 75% to pass
- Class attendance is required

Textbook and Recommended Readings

- Textbook:
  - Database Management Systems, 3rd Edition by Ramakrishnan and Gehrke



- Other Recommended Readings:
  - Relational Database Systems, by D. Simovici and R. Tenney



  - Other resources will be posted on the class website

Class pre-requisites

- CS 310
- CS 240
- Discrete math
- Programming: Java; some basic Python familiarity
- Familiarity with Unix OS: We will use an Oracle 12G server running on a Unix machine in the CS department(DBS3 Machine)

University Policies

- **Student Conduct**: Students are required to adhere to the University Policy on Academic Standards and Cheating, to the University Statement on Plagiarism and the Documentation of Written Work, and to the Code of Student Conduct as delineated in the University Catalog and Student Handbook. The Code is available online at:

  http://www.umb.edu/life_on_campus/policies/code/

- **Accommodations:** Section 504 of the Americans with Disabilities Act of 1990 offers guidelines for curriculum modifications and adaptations for students with documented disabilities. If applicable, students may obtain adaptation recommendations from the Ross Center for Disability Services, CC-UL Room 211, (617-287-7430). The student must present these recommendations and discuss them with each professor within a reasonable period, preferably by the end of Drop/Add period.

Things to do by Wednesday (July 19th)

- Requesting a UNIX account(if you don't have one already):
  - https://portal.cs.umb.edu/register/
- Register to CS630 class on the Unix account:
  - https://portal.cs.umb.edu/accounts/login/
  - Both cs430 and cs630 students MUST register to the cs630 class from the cs portal. Once you login to the cs portal(using your unix account), you will have to go to Courses, then select cs630 and register to this course! This is required to be able to submit homework. It is also required in order to get an Oracle account!

Topics

- Introduction to DBMS
- Relational Data Model
- Relational Algebra
- Conceptual Design - the Entity-Relationship Model
- Structured Query Language (SQL)
- Database Security and Authorization
- Schema Refinement and Normal Forms
- Application Development (Java, Python)
- Some NoSQL topics (If time permitted)

Introduction to DBMS

- What is a DBMS
- Why DBMS are important
- History of DBMS
- Some newer paradigms

The need of DBMS

- The need to store data existed since ancient times. E.g.: taking notes on a piece of paper
- With the invention of computers, data could be stored on disk in files of different formats
- Limitations: main memory might not be enough for large files, data inconsistency challenges, apps must address concurrent access, the search within a file is most often linear, access/security limitations, etc.

Database Management Systems (DBMS)

- Collection of specialized software that provides:
  - Uniform and transparent access to data
  - Efficient data access
    - Fast search, indexing
  - Data consistency
    - e.g.: you cannot delete student record if grade record still in DBMS
  - Concurrent access to data
  - Persistent storage and recovery from failure
  - Security

Database Management Systems (DBMS)

- *Database*: A collection of structured data, organized for fast retrieval and search
- Example:
  - Entities: Students, Faculty, Courses
  - Relationship between entities: Faculty teaches Courses, Students enrolled into Courses

Why studying DBMS is important

- Databases are ubiquitous
- In industry, most software jobs deal (at different degrees) with DMBS
  - Database Administrators (DBAs)
  - Data Engineers
  - Data Analysts and Data Scientists
  - Backend Software Engineers
  - Front-end Software Engineers

History of DMBS

- Early 1960s: first general-purpose DBMS, called Integrated Data Store, using a network data model. Designed by Charles Bachman, at General Electric
- Late 1960s: IBM developed the Information Management System (IMS) DBMS, using a hierarchical data model
- 1970: Edgar Codd, at IBM proposed a new data model called the ***relational data model***. Followed by a rapid development of several commercial DBMS
  - *Relations* (*tables*) with *tuples* (*records*) and *fields* (*columns*)

History of DBMS (cont.)

- 1973: INGRES was developed UC Berkeley (used QUEL language)
- 1974: IBM developed SQL (Structured Query language), initially called SEQUEL (Structured English Query Language), as part of the IBM System R database system
- 1978: Oracle, first commercially available RDBMS
- 1983: DB2 (i.e. IBM Database 2)
- 1986: Sybase DB (i.e. Sybase SQL Server) developed for Unix
- 1989: Microsoft SQL Server started as a project to port Sybase DB to OS/2

History of DBMS (cont.)

- 1980s: relational data model consolidated its position as the dominant DBMS paradigm.
- Late 1980s: SQL was standardized. In 1986 adopted by the American National Standards Institute (ANSI), and in 1987 adopted by the International Organization for Standardization (ISO)
- 1996: PostgreSQL was released. A successor of Ingres, developed at UC Berkley

Nowadays

- Some popular RDMBS (Relational Database Management Systems) in industry include:
  - Oracle
  - PostgreSQL
  - MySQL
  - Amazon Aurora
  - AWS RDS: managed relational database service for MySQL, Oracle BYOL, PostgreSQL, etc

Some column-based DBMS

- Cassandra
- Vertica
- AWS Redshift: data warehouse, based on PostgreSQL

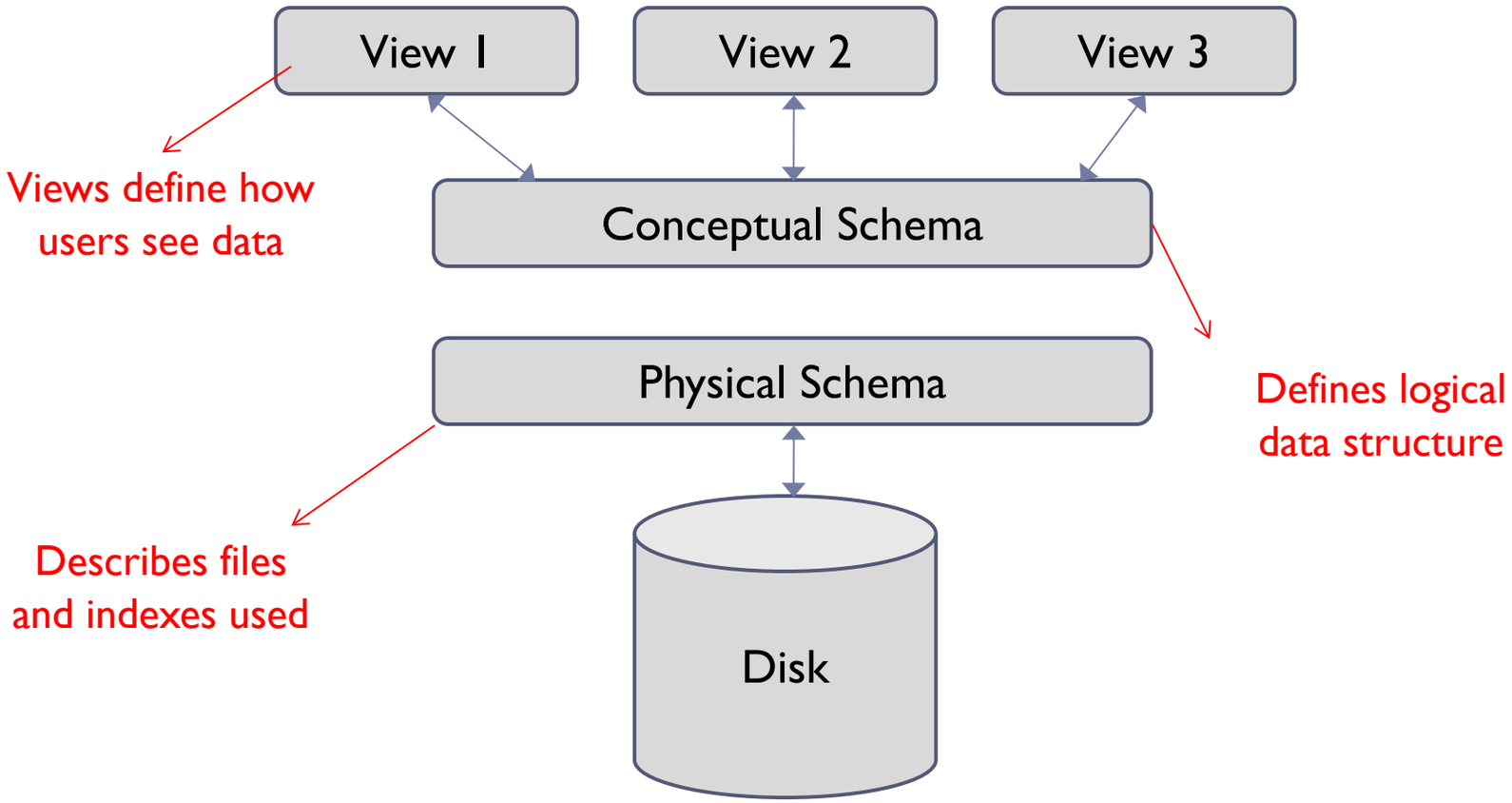
Example of other NoSQL paradigms

- Key-value stores (E.g.: Amazon DynamoDB)

- Graph databases (E.g.: Neo4j, Amazon Neptune)

- Big Data Processing and Querying Technologies (Spark SQL, Apache Hive ,etc)

Levels of Abstraction in a DBMS



Database Management Systems 3rd ed, Ramakrishnan and Gehrke

Levels of Abstraction in a DBMS (cont.)

- *Conceptual schema*: describes the stored data in terms of the data model

- *Physical schema*: specifies additional storage details. It summarizes how relations described by the conceptual schema are stored on disk. Stores all additionally structures such as indexes.

- *External Schemas/Views*: a collection of one or more views and relations from conceptual schemas.

Topics

- Introduction to DBMS
- *Relational Data Model*
- Relational Algebra
- Conceptual Design - the Entity-Relationship Model
- Structured Query Language (SQL)
- Schema Refinement and Normal Forms
- Database Security and Authorization
- Application Development (Java, Python)
- Some NoSQL topics (If time permitted)

Data Model

- Structure of data
  - Relational model uses tables
  - Programming languages deal with arrays, maps..etc
- Operations on data
  - queries: operations that retrieve information
  - modifications: operations that change/update data
- Constraints
  - Domain constraints. E.g.: age must be numeric
  - Other constraints. E.g.: each student id must be unique

Relational Data Model

- Relational database: a set of relations
- Relation:
  - 2D table (rows and columns)
  - #rows = cardinality;
  - # columns = degree (or arity)
  - Each row represents an entity
    - E.g. :A student, a course, a book
    - Rows are also called tuples or records
  - Each column represents a property of the entity
    - E.g. a student entity can contain: student name, student id, gpa
    - Column values are atomic within a given domain
    - Columns are also called fields or attributes

Students Relation (i.e. Table)

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Cardinality = 3 ; Degree = 5

Book Relation

Bid	Name	Author	Year
123	Book_1	Xy	2020
456	Book_k	Zw	2022

Cardinality = ? ; Degree = ?

Relational Schema

- *Schema*: specifies the name of the relation and the name and domain for each column/entity

Student (sid: integer,
name: string,
login: string,
age: integer,
gpa: real)

- Each relation must have a schema

- *Relational Database Schema* = collection of relations' schemas

Relations (cont.)

- Relations are sets of tuples (i.e. records, rows)
  - Sets are not ordered
- *Relation Instance*
  - Represents a set of tuples at a given time
  - Content might change over time. E.g. add/ remove/ modify a tuple/record
- Schema change might occur as well
  - Could be expensive

Instance of Students relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Cardinality = 3 ; Degree = 5

Another Instance of Students relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.6
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8
53699	Joe	joe@umb	20	4

Cardinality = 4 ; Degree = 5

Keys

- A *key* of a relation is a set of fields K such that

1. No two tuples in the relation have the same key value (same value for all fields forming the key). This means that it uniquely identifies a tuple (i.e.record)
2. No subset of fields in K is a key (otherwise K is a *superkey*)

- Key might not be unique
  - Multiple *candidate keys* might exist
  - One key is chosen to be the *primary key*

- In the relational data model duplicate tuples do not exits
  - In practice, most DBMS allow duplicates
  - Key constraints must be set by DBA to avoid duplicates

Example of Keys

- Student (sid: integer, name: string, login:string, age:integer, gpa:real)
- sid = primary key; (sid, name) = super key

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- In practice, it is difficult to know when a unique combination of fields/columns exist. Therefore, synthetic/artificial keys are created (e.g. sid)

Topics

- Introduction to DBMS
- Relational Data Model
- *Relational Algebra*
- Conceptual Design - the Entity-Relationship Model
- Structured Query Language (SQL)
- Schema Refinement and Normal Forms
- Database Security and Authorization
- Application Development (Java, Python)
- Some NoSQL topics (If time permitted)

Formal query languages

- Database queries (SQL) are used to communicate with a RDBMS
- Input and output of queries are relations
- Two formal query languages are at the foundation of SQL:
  - *Relational Algebra*
    - Operational
    - Useful to represent the *execution plans*
    - Very relevant, as it is used by the *query optimizers*
  - *Relational Calculus*
    - Declarative. It allows the user to describe the results, not how to compute them
    - We will not focus on this in this class

Relational Algebra

- Queries are composed using a *collection of operations*
- Every operator in the algebra accepts one or two relation instances as arguments and returns a relation instance as the result
- Above property makes it easy to *compose* multiple operators to create complex queries
- A relational algebra expression is defined recursively, a unary algebra operator applied to a single expression, or a binary algebra operator applied to two expressions

Basic Relational Algebra Operators

- *Projection* π selects only a given subset of columns from the relation
- *Renaming* ρ helper operator. It does not return a new relation.
  - It only renames the relation and/or the fields
- *Selection* σ selects a subset of tuples/rows from the relation
- *Cross-product* $\times$ allows us to combine multiple relations, by performing the cartesian product
- *Join* $\bowtie$ combines multiple relations using conditions
  - condition join, equijoin, natural-join
- *Set-difference* $-$, *Union* $\cup$, *Intersection* $\cap$
- *Division* / used for some special types of queries. It is more complex

Projection

$$\pi_{attr1,attr2,..,attrk} relation$$

- Unary operator
- Input: one relation
- Extracts only the attributes that are in the attributes list (i.e. attr1, attr1, …, attrK). It projects out the attributes that are not in the list.
- Output: another relation that contains as columns only the attributes from the attributes list (in the same order as in the list)
- Projection operator has to eliminate duplicates
  - Real systems do not typically eliminate duplicates by default
    - Eliminating duplicates is very expensive (sorting)
    - Users must explicitly request duplicate elimination (in SQL, using DISTINCT)

Projection Example 1

Sailors table S

S

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
44	guppy	5	35.0
58	rusty	10	35.0

$\pi_{sname,rating}S$

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

Projection Example 2

S

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
44	guppy	5	35.0
58	rusty	10	35.0
40	rusty	10	40.0

$\pi_{sname,rating}S$

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

Projection eliminates duplicates !

Renaming Operator

$$\rho(R(\bar{F}), E)$$

- E = relational algebra expression
- R = the output relation. It has same schema and the same number of records as E, but some fields are renamed
- The fields of R are the same as E, except that some fields are renamed in the renaming list $\bar{F}$
- $\bar{F}$ is a list containing renaming statements $oldName \rightarrow newName$ or $position \rightarrow newName$. If more renaming statements, they are separated by comma
- **Very important:** for ρ to be well defined, oldName and position used must exist, and no two fields in the result can have the same name

Renaming Example 1

S

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
44	guppy	5	35.0
58	rusty	10	35.0
40	rusty	10	40.0

$\pi_{sname,rating}S$

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$$\rho(C(sname -> sailor), \pi_{sname,rating}S)$$

sailor	rating
yuppy	9
lubber	8
guppy	5
rusty	10

Same could have been written as

$$\rho(C(1 -> sailor), \pi_{sname,rating}S)$$

Basic Relational Algebra Operators

- *Projection* π selects only a given subset of columns from the relation
- *Renaming* ρ helper operator. It does not return a new relation.
  - It only renames the relation and/or the fields
- *Selection* σ selects a subset of tuples/rows from the relation
- *Cross-product* $\times$ allows us to combine multiple relations, by performing the cartesian product
- *Join* $\bowtie$ combines multiple relations using conditions
  - condition join, equijoin, natural-join
- *Set-difference* $-$, *Union* $\cup$, *Intersection* $\cap$
- *Division* / used for some special types of queries. It is more complex

Selection

$$\sigma_{condition} relation$$

- Unary operator
- Input: one relation
- Condition contains constants and attributes (i.e. fields, columns) from the relation
  - Condition: attribute <op> constant
  - Condition: attribute <op> attribute
  - <op> could be: <, <=, = , >=, >, <> (or !=)
  - Conditions are evaluated for each individual row (i.e. tuple)
  - Only rows that satisfy the conditions are returned
  - If more than one condition, they must be connected by logical connectors (AND $\wedge$, OR $\vee$, NOT $\neg$)
- Output: a new relation with same schema as original relation, containing only the rows that satisfy the condition(s)
- The result contains no duplicates! Why?

Selection Example 1

$\sigma_{rating>=8}S$?

S

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
44	guppy	5	35.0
58	rusty	10	35.0
40	rusty	10	40.0

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
~~44~~	~~guppy~~	~~5~~	~~35.0~~
58	rusty	10	35.0
40	rusty	10	40.0

$\sigma_{rating>=8}S$

Result is

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
58	rusty	10	35.0
40	rusty	10	40.0

Selection Example 2

S

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
44	guppy	5	35.0
58	rusty	10	35.0
40	rusty	10	40.0

$\sigma_{(rating>=8)\wedge(age=35.0)}S$

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Selection Example 3 (projection and selection combined)

S $\pi_{sname,rating}(\sigma_{(rating>=8)\wedge(age=35.0)}S)$

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
44	guppy	5	35.0
58	rusty	10	35.0
40	rusty	10	40.0

?

Selection Example 3 (projection and selection combined) (cont.)

S $\pi_{sname,rating}(\sigma_{(rating>=8)\wedge(age=35.0)}S)$

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
44	guppy	5	35.0
58	rusty	10	35.0
40	rusty	10	40.0

First step: we calculate the selection

sid	sname	rating	age
28	yuppy	9	35.0
~~31~~	~~lubber~~	~~8~~	~~55.0~~
~~44~~	~~guppy~~	~~5~~	~~35.0~~
58	rusty	10	35.0
~~40~~	~~rusty~~	~~10~~	~~40.0~~

Intermediate result is

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Selection Example 3 (projection and selection combined (cont.)

S $\pi_{sname,rating}(\sigma_{(rating>=8)\wedge(age=35.0)}S)$

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.0
44	guppy	5	35.0
58	rusty	10	35.0
40	rusty	10	40.0

Second (and final) step: we calculate the projection

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Result is

sname	rating
yuppy	9
rusty	10

Selection Example 4
Students relationT

T

sid	name	city	state
12661	Mary	Burlington	MA
12890	Joe	New York	NY
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

$$\sigma_{(city='Boston')\vee(city='Burlington')}T$$

sid	name	city	state
12661	Mary	Burlington	MA
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

Given the relation T from above, how do we write the relational algebra expression to extract information about students who live in Massachusetts, cities Boston or Burlington?

Selection Example 4 (cont.)

T

sid	name	city	state
12661	Mary	Burlington	MA
12890	Joe	New York	NY
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

$$\sigma_{(city='Boston')\vee(city='Burlington')}T$$

sid	name	city	state
12661	Mary	Burlington	MA
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

$$\sigma_{((city='Boston')\vee(city='Burlington'))\wedge(state='MA')}T$$

sid	name	city	state
12661	Mary	Burlington	MA
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

Selection Example 5 (projection and selection combined)

T

sid	name	city	state
12661	Mary	Burlington	MA
12890	Joe	New York	NY
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

$$\sigma_{(city='Boston')\vee(city='Burlington')}T$$

sid	name	city	state
12661	Mary	Burlington	MA
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

Given relation T, the relational algebra expression that extracts the name and state for all students who live in Massachusetts in cities Boston or Burlington

?

Selection Example 5 (projection and selection combined) (cont.)

T

sid	name	city	state
12661	Mary	Burlington	MA
12890	Joe	New York	NY
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

$\sigma_{(city='Boston')\vee(city='Burlington')}T$

sid	name	city	state
12661	Mary	Burlington	MA
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

Given relation T, the relational algebra expression that extracts the name and state for all students that live in Massachusetts in cities Boston or Burlington

$$\pi_{name,state}(\sigma_{((city='Boston')\vee(city='Burlington'))\wedge(state='MA')}T)$$

Result?

Selection Example 5 (projection and selection combined) (cont.)

T

sid	name	city	state
12661	Mary	Burlington	MA
12890	Joe	New York	NY
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

$\sigma_{(city='Boston')\vee(city='Burlington')}T$

sid	name	city	state
12661	Mary	Burlington	MA
13566	Anne	Burlington	VT
14022	Jerry	Boston	MA
15544	Mary	Boston	MA

Given relation T, the relational algebra expression that extracts the name and the state for all students that live in Massachusetts in cities Boston or Burlington

$$\pi_{name,state}(\sigma_{((city='Boston')\vee(city='Burlington'))\wedge(state='MA')}T)$$

name	state
Mary	MA
Jerry	MA

Observe that duplicates are removed from the result!

Basic Relational Algebra Operators

- *Projection* π selects only a given subset of columns from the relation
- *Renaming* ρ helper operator. It does not return a new relation.
  - It only renames the relation and/or the fields
- *Selection* σ selects a subset of tuples/rows from the relation
- *Cross-product* $\times$ allows us to combine multiple relations, by performing the cartesian product
- *Join* $\bowtie$ combines multiple relations using conditions
  - condition join, equijoin, natural-join
- *Set-difference* $-$, *Union* $\cup$, *Intersection* $\cap$
- *Division* / used for some special types of queries. It is more complex

Cross-product

$$R \times S$$

- Binary operator
- R, S are two relations
- Performs the cartesian product between the two relations from the input
  - Each row in R is paired with each row in S
  - Output: a new relation that contains the cartesian product between the two input relations.
    - Resulted relation will contain all attributes from both relations
    - First attributes of R , in the same order as in R, followed by attributes from S (in the same order)
    - If a field from R has the same name as a field from S, a naming conflict occurs. In such cases, the corresponding fields from R X S are unnamed and are referred to solely by position. Note that fields position starts from 1.

Cross-product Example 1
Sailors S, Reserves R, Boats B

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

$S \times R$?

Cross-product Example 1 (cont.)

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

$S \times R$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/22
22	dustin	7	45.0	58	103	10/11/22
31	lubber	8	55.0	22	101	10/10/22
31	lubber	8	55.0	58	103	10/11/22
58	rusty	10	35.0	22	101	10/10/22
58	rusty	10	35.0	58	103	10/11/22

Note that fields at position 1 and 5 got unnamed

Cross product and Renaming, Example 2

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

$$\rho(C(1 \rightarrow sid1, 5 \rightarrow sid2), S \times R)$$

C

sid1	sname	rating	age	sid2	bid	day
22	dustin	7	45.0	22	101	10/10/22
22	dustin	7	45.0	58	103	10/11/22
31	lubber	8	55.0	22	101	10/10/22
31	lubber	8	55.0	58	103	10/11/22
58	rusty	10	35.0	22	101	10/10/22
58	rusty	10	35.0	58	103	10/11/22

Cross product Example 3

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

$S \times B$

sid	sname	rating	age	bid	name	color
22	dustin	7	45.0	101	interlake	red
22	dustin	7	45.0	103	clipper	green
22	dustin	7	45.0	102	vacation	red
31	lubber	8	55.0	101	interlake	red
31	lubber	8	55.0	103	clipper	green
31	lubber	8	55.0	102	vacation	red
58	rusty	10	35.0	101	interlake	red
58	rusty	10	35.0	103	clipper	green
58	rusty	10	35.0	102	vacation	red

Result contains all possible combinations of sailors and boats

Basic Relational Algebra Operators

- *Projection* π selects only a given subset of columns from the relation
- *Renaming* ρ helper operator. It does not return a new relation.
  - It only renames the relation and/or the fields
- *Selection* σ selects a subset of tuples/rows from the relation
- *Cross-product* $\times$ allows us to combine multiple relations, by performing the cartesian product
- *Join* $\bowtie$ combines multiple relations using conditions
  - condition join, equijoin, natural-join
- *Set-difference* $-$, *Union* $\cup$, *Intersection* $\cap$
- *Division* / used for some special types of queries. It is more complex

Condition Join

$$R \bowtie_C S$$

- Also called the Theta Join, also noted $R \bowtie_\theta S$
- Result equal to the Cross product of relations R and S followed by a selection with condition C
- $R \bowtie_C S = \sigma_C(R \times S)$
- Condition has same format as a Selection condition with attributes typically from both relations
- Output: one new relation. Its schema is the same as the schema of the cross product
- The reference of the attributes from a relation R are referenced either by position R.i (where i is the position on the attribute; first attribute in a relation is at position 1), or referenced by name, as R.attributename

Condition Join Example 1

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

We want to calculate

$$S \bowtie_{S.sid<R.sid} R$$

Condition Join Example 1 (cont.)

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

$$S \bowtie_{S.sid<R.sid} R$$

Step 1:cross product

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/22
22	dustin	7	45.0	58	103	10/11/22
31	lubber	8	55.0	22	101	10/10/22
31	lubber	8	55.0	58	103	10/11/22
58	rusty	10	35.0	22	101	10/10/22
58	rusty	10	35.0	58	103	10/11/22

Condition Join Example 1 (cont.)

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

$$S \bowtie_{S.sid<R.sid} R$$

Step 2: do the filtering

(sid)	sname	rating	age	(sid)	bid	day
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~22~~	~~101~~	~~10/10/22~~
22	dustin	7	45.0	58	103	10/11/22
~~31~~	~~lubber~~	~~8~~	~~55.0~~	~~22~~	~~101~~	~~10/10/22~~
31	lubber	8	55.0	58	103	10/11/22
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~22~~	~~101~~	~~10/10/22~~
58	rusty	10	35.0	58	103	10/11/22

Condition Join Example 1 (cont.)

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

Result is:

$$S \bowtie_{S.sid<R.sid} R$$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	10/11/22
31	lubber	8	55.0	58	103	10/11/22

Condition Join Example 1 (cont.)

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

We can also use the renaming operator

$$\rho(C(1 \rightarrow sids, 5 \rightarrow sidr), S \bowtie_{S.sid<R.sid} R)$$

C

sids	sname	rating	age	sidr	bid	day
22	dustin	7	45.0	58	103	10/11/22
31	lubber	8	55.0	58	103	10/11/22

Equijoin

$$R \bowtie_{R.attX=R.attrY} S$$

- Subset of condition join
- Here condition can only contain equalities of form S.name1=R.name2. In the resulted relation, R.name2 field is dropped. Two or more equality conditions connected by $\wedge$ are possible. Example: $R \bowtie_{(R.attrX=S.attrY)\wedge(R.attrZ=S.attrX)} S$
- Output: a new relation with a schema that will contain all fields of R, followed by all fields of S that do not appear in the join condition

Equijoin Example 1

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

$$S \bowtie_{S.sid=R.sid} R$$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/22
58	rusty	10	35.0	103	10/11/22

Equijoin Example 2 (same as before, but R's sid column is named as sailorid)

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sailorid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

$$S \bowtie_{S.sid=R.sailorid} R$$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/22
58	rusty	10	35.0	103	10/11/22

Natural Join

$$R \bowtie S$$

- A special case of equijoin
- Condition is equality on all fields from R and S that share the same name. Because of this, no condition needs to be specified, as it is implied
- The resulted relation will contain the results from a cross-product between R and S, but retain only the records that satisfy the equality conditions
- Schema of the resulted relation will contain all fields from R, followed by all fields from S that are not present in R

Natural Join Example 1

S

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.0
58	rusty	10	35.0

R

sid	bid	day
22	101	10/10/22
58	103	10/11/22

B

bid	name	color
101	interlake	red
103	clipper	green
102	vacation	red

$S \bowtie R$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/22
58	rusty	10	35.0	103	10/11/22

Natural Join Example 2

A

actorid	name	city	age
210	sandy	Los Angeles	30
205	robert	Boston	50
211	linda	New York	25

P

prodid	name	city	agep	company
60	andy	Hollywood	45	WB
50	sandy	Los Angeles	30	Universal

$A \bowtie P$

actorid	name	city	age	prodid	agep	company
210	sandy	Los Angeles	30	50	30	Universal

Questions?