CompSci 516
Database Systems

Lecture 1
Introduction
and
SQL

Instructor: Sudeepa Roy

1

Course Website

- http://www.cs.duke.edu/courses/spring22/compsci516/
- Please check frequently for updates!

2

Instructor

- Sudeepa Roy
  - sudeepa@cs.duke.edu
  - https://users.cs.duke.edu/~sudeepa/
- About myself
  - Assistant Professor in CS
  - PhD: UPenn, Postdoc: Univ. of Washington
  - Joined Duke CS in Fall 2015
  - Research interests:
    - Data Analysis, causality, query optimization, data science, database theory, applications of data, uncertain data,…

3

Three TAs

- Yuxi Liu (grad TA)

- Shweta Patwa (grad TA)

- Joon Young Lee (UTA)

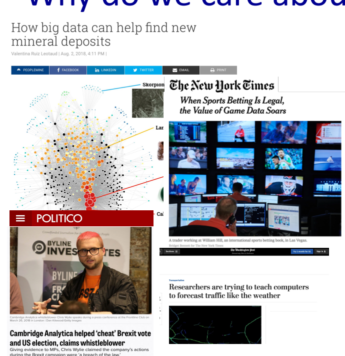
4

What are the goals of this course?

- Learn about "Database Systems" or Data Management in general

5

Why do we care about data? (easy)



… The three years of gathering and analyzing data culminated in what U.S. Sailing calls their "Rio Weather Playbook," a body of critical information about each of the seven courses only available to the U.S. team…

— FiveThirtyEight, **"Will Data Help U.S. Sailing Get Back On The Olympic Podium?"** Aug 15, 2016

Data =
Money
Information
Power
Fun
in
Science, Business, Politics, Security
Sports, Education, ….

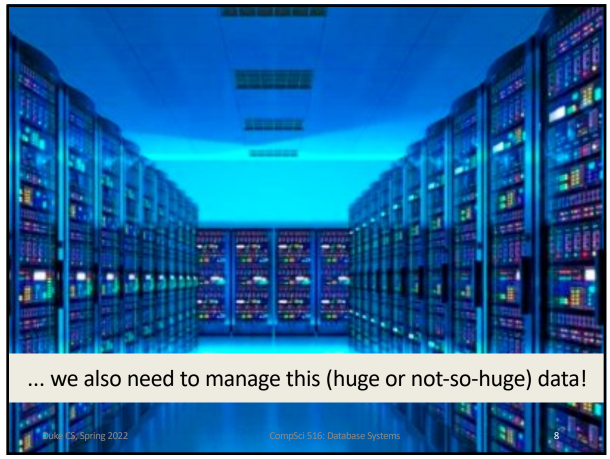
6

Wait.. don't we need to take a Machine Learning or Statistics course for those things?

Yes, but..

Pic: https://www.technobuffalo.com/sites/technobuffalo.com/files/styles/xlarge/public/wp/2012/05/confused-student.jpg

7



8

Also think about building any application based on data from scratch

- E.g., your own version of mini-Amazon or a Book Selling Platform
- Large data! (think about all books in the world or even in English)

- How do we start?
- A short background survey first…

9

Who are the key people? (book-selling website)

10

Who are the key people? (book-selling website)

11

What should the user be able to do?

- i.e. what the interface look like? (think about Amazon)

12

What should the user be able to do?

- i.e. what the interface look like? (think about Amazon)

13

What should the platform do?

14

What should the platform do?

15

What are the desired and necessary properties of the platform?

16

What are the desired and necessary properties of the platform?

17

That was the design phase
(a basic one though)

How about C++, Java, or Python?
On data stored in large files

https://i1.wp.com/dynamiclandscapes.vita-learn.org/wp-content/uploads/2019/05/lets-code.jpg?resize=768%2C432&ssl=1

18

Sounds simple!

James Morgan#Durham, NC

… …
A Tale of Two Cities#Charles Dickens#3.50#7
To Kill a Mockingbird#Harper Lee#7.20#1
Les Miserables#Victor Hugo#12.80#2
… …

- Text files – for books, customer, …
- Books listed with title, author, price, and no. of copies
- Fields separated by #'s

19

Query by programming

James Morgan#Durham, NC

… …
A Tale of Two Cities#Charles Dickens#3.50#7
To Kill a Mockingbird#Harper Lee#7.20#1
Les Miserables#Victor Hugo#12.80#2
… …

- James Morgan wants to buy "To Kill a Mockingbird"
- A simple script
  – Scan through the books file
  – Look for the line containing "To Kill a Mockingbird"
  – Check if the no. of copies is >= 1
  – Bill James $7.20 and reduce the no. of copies by 1

Better idea than scanning?

What if he changes the "query" and wants to buy a book by Victor Hugo?

20

Revisit: What are the desired and necessary properties of the platform?

- Should be able to handle a large amount of data
- Should be efficient and easy to use (e.g., search with authors as well as title)
- If there is a crash or loss of power, information should not be lost or inconsistent
  – Imagine a user was in the middle of a transaction when a crash happened, paid the money, but the book has not been purchased
- No surprises with multiple users logged in at the same time
  – Imagine one last copy of a book that two users are trying to purchase at the same time
- Easy to update and program
  – For the admin

Try to open a 10-100 GB file

Try to search both on a large flat file

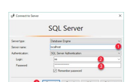
Imagine programmer's task

Imagine adding a new book or updating Copies (+ allow search) on a 10-100 GB text file

21

Solution?

- DBMS = Database Management System



PostgreSQL MySQL ORACLE DATABASE IBM DB2 mongoDB

22

A DBMS takes care of all of the following (and more):

In an easy-to-code, efficient, and robust way

- Should be able to handle a large amount of data ✓
- Should be efficient and easy to use (e.g., search with authors as well as title) ✓
- If there is a crash or loss of power, information should not be lost or inconsistent ✓
  – Imagine a user was in the middle of a transaction when a crash happened, paid the money, but the book has not been purchased
- No surprises with multiple users logged in at the same time ✓
  – Imagine one last copy of a book that two users are trying to purchase at the same time
- Easy to update and program ✓
  – For the admin

Optimization

Index

Recovery

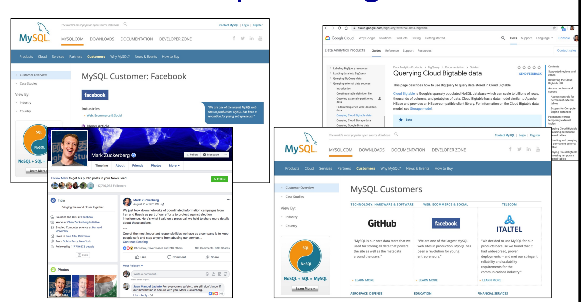
Concurrency Control

Declarative

* We will learn these in the course!

23

DBMS helps the big ones!



Note: Not always the "standard" DBMS (called Relational DBMS), but we need to know pros and cons of all alternatives

24

CompSci 516: how database systems work and can be used by users

- This is a graduate-level database course in CS
  - We will cover principles, internals, and applications of database systems
- How can a user use a DBMS (programmer's/designer's perspective)
  - Run queries, update data, views (SQL, Relational Algebra, Relational calculus)
  - Design a good database (normalization, constraints)
  - Use different types of data (Mostly relational, also XML/JSON)
- How does a DBMS work (system's or admin's perspective, also for programmers for writing better queries)
  - Storage, index, query processing, join algorithms, query optimizations
  - Transactions: recovery and concurrency control
- Glimpse of advanced topics and other DBMS
  - NOSQL, Spark (big data), data mining, Datalog/recursive queries, Parallel and distributed DBMS

25

What this course is NOT about

- Spark, AWS, cluster computing…
  - Partially covered in a HW and a lecture
- Machine learning based analytics
- Statistical methods for data analytics
- Python, R
- Data mining..

26

Warnings!

- Course-load: Three solid programming-heavy and problem-solving HW on three systems + project + quizzes/labs + midterm + final
- HWs will be mostly "disjoint" from lectures and largely self-taught
  - use online tutorials and help of TAs, save enough time
  - HW1: Traditional DBMS -- SQL and Postgres (and some XML too!)
  - HW2: Distributed data processing -- Spark and AWS
  - HW3: NOSQL, MongoDB
- Lecture materials are tested in exams
  - Make sure that you follow the lectures before the next class
  - Some practice problems in quizzes and in class
- We will assume backgrounds in some programming language, and in algorithms, data structures, sets, basic systems
  - If you are from a non-CS background, would need to learn these as needed quickly
- There will be some "theory", algorithms, maths, and proofs, e.g.,
  - Relational calculus ≣ First-order logic (∀, ∃, ∧, ∨, ⇒)
  - Database normal forms, fixpoints (recursive queries), directed acyclic graphs (transactions)

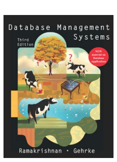
27

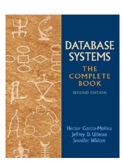
Logistics

- Discussion forum: Ed
  - All enrolled students are already there
- To reach course staff:
  - Private message on Ed to reach all staff, or send instructor email only for logistics
- Lecture slides will be uploaded before the class as incomplete notes
  - will be updated after the class
- Other tools
  - Gradescope (already enrolled)
  - Sakai
  - Gradiance (code will be sent)

28

Reading Material





- Will mostly follow the "cowbook" by Ramakrishnan-Gehrke
  - The chapter numbers will be posted
- You do not have to buy the books, but it might be good to consult them from time to time

29

Grading

- Three Homework: 30%
- Group Project: 13%
- Midterm: 20%
- Final: 25%
  - Exams are comprehensive, closed book/notes, no collaboration
- Class participation: 12%
  - Quizzes/labs in class or short deadline (lowest score dropped): 10%
  - Communication: 2%

Please bring laptops every day!

30

Grading Strategy

- Absolute and adjustable grading
  - 90% or above: Guaranteed A-range grades (A-, A, or A+)
  - 80-90%: Guaranteed B-range grades (B-, B, or B+)
  - Etc. (see website)
- Thresholds may slide down (your grade may go up) based on class performance
- +/- will depend on the class performance
  - Topper of the class gets A+, and all and only “above expectation” performances get A+.
- Everyone can get good grade by working hard!

31

Homework

- Due in about 2 weeks after they are posted/previous hw is due
  - ALWAYS start early!
  - Part of the homework may be due in 1 week
- Late policy:
  - 5% penalty per hour unless there are valid reasons or permissions
  - Check out website
  - Start early and do not count on late days!
- contact the instructor if you have a *valid* reason to be late
  - Another exam, project, hw is NOT a valid reason – we will always be fair to all
- To be done individually, but discussions allowed
  - More details in the next class

32

Projects

- 13% weight
- In groups of 4
  - Groups of smaller and larger sizes need instructor’s permission
- Flexible in terms of topic related to data management
  - Fixed option: Database-backed website (template provided)
  - Open options (needs to be approved by the instructor): analyzing data, any db-related research problem
  - Running a standard ML classification/regression on a dataset is not enough
- Work done should be at least equivalent to
  - 1.3 hw * no. of group members
- Weekly updates + proposal + midterm report + final report + demo/presentation
- More information and ideas for projects will be posted later

33

Some Important Dates

- January 20:
  - Team members’ names + tentative project topic due
- February 1:
  - Project proposal due
- February 15:
  - Midterm
- HW1 will be released next week

34

Please ask questions in class!

- In general, actively participate in the class!
  - Ask questions in class and on Ed
  - Stop me as many times as you need to understand the lectures
  - Answer each other’s questions on Ed – this is a big class and getting answers from TAs may take time, although we will monitor all threads
- Anonymous feedback form link on Ed
  - To be checked at least once weekly
  - All feedback, suggestions, concerns welcome!

35

Let’s get started!

Relational Data Model

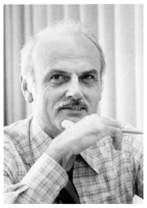
What is a good model to store data?
Tree? Nested data? Graph?

(just) Tables!

https://studentreasures.com/blog/student-publishing/young-authors-in-the-spotlight-celebrating-the-winners-of-the-2018-national-book-challenge/

36

Edgar F. Codd (1923-2003)



- Pilot in the Royal Air Force in WW2
- Inventor of the relational model and algebra while at IBM (1970)
- Turing Award, 1981

RDBMS = Relational DBMS

Motivation of relational model
- Simplicity
- Easy query optimizations
- Separation of abstraction and operations

http://en.wikipedia.org/wiki/File:Edgar_F_Codd.jpg

37

Relational Data Model

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith1@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- The data description construct is a Relation
  - Represented as a "table"
  - Basically a "set" of records (set semantic)
  - order does not matter
  - and all records are distinct
- however, it is true for the relational model, not for standard DBM
  - allow duplicate rows (bag semantic)
  - unless restricted by key constraints. Why?

Bag: {1, 1, 2, 2, 3, 2, 1, 5, 6, 1}
Set: {1, 2, 3, 5, 6}

38

Bag vs. Set

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith1@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- Why "bag semantic" and not "set semantic" in standard DBMSs?

39

Relational Data Model

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith1@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Attribute/ Column/ Field

Tuple/ Row/ Record

Value

What is a poorly chosen attribute in this relation?

- Relational database = a set of relations
- A Relation : made up of two parts
  1. Schema
  2. Instance

40

Schema and Instance

- One schema can have multiple instances
- Schema:
  - A template for describing an entity/relationship (e.g. students)
  - specifies name of relation + name and type of each column

  e.g. Students(sid: string, name: string, login: string, age: integer, gpa: real).
- Instance:
  - When we fill in actual data values in a schema
  - a table, has rows and columns
  - each row/tuple follows the schema and domain constraints
  - #Rows = cardinality, #fields = degree or arity
  - example below

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith1@math	19	3.8

Cardinality = 3, degree = 5

41

SQL (Structured Query Language)

See the separate instructions to install postgres to practice!

42

Relational Query Languages

- A major strength of the relational model: supports simple, powerful querying of data.
- Queries can be written intuitively, and the DBMS is responsible for an efficient evaluation
  - The key: precise semantics for relational queries
  - Based on a sound theory!
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

43

The SQL Query Language

- Developed by IBM (systemR) in the 1970s based on Ted Codd's relational model
  - First called "SEQUEL" (Structured English Query Language)
- First commercialized by Oracle (then Relational Software)in 1979
- Standards by ANSI and ISO since it is used by many vendors
  - SQL-86, -89 (minor revision), -92 (major revision), -96, -99 (major extensions), -03, -06, -08, -11, -16

44

Purposes of SQL

- Data Manipulation Language (DML)
  - Querying: SELECT-FROM-WHERE
  - Modifying: INSERT/DELETE/UPDATE (next week)
- Data Definition Language (DDL)
  - CREATE/ALTER/DROP (next week)

45

The SQL Query Language

- To find all 18 year old students, we can write:

all attributes

```
SELECT *
FROM Students S
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- To find just names and logins, replace the first line:

```
SELECT S.name, S.login
```


46

Querying Multiple Relations

- What does the following query compute?

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade="A"
```

Given the following instances of Enrolled and Students:

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

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

we get: ??

47

Querying Multiple Relations

- What does the following query compute?

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade="A"
```

Given the following instances of Enrolled and Students:

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

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

we get:

48

Read yourself, after reading the next few slides first

Basic SQL Query

SELECT [DISTINCT] *<target-list>*
FROM *<relation-list>*
WHERE *<qualification>*

- relation-list A list of relation names
  - possibly with a "range variable" after each name
- target-list A list of attributes of relations in relation-list
- qualification Comparisons
  - (Attr op const) or (Attr1 op Attr2)
  - where op is one of = , <, >, <=, >= combined using AND, OR and NOT
- DISTINCT is an optional keyword indicating that the answer should not contain duplicates
  - Default is that duplicates are not eliminated!

49

Read yourself, after reading the next few slides first

Conceptual Evaluation Strategy

SELECT [DISTINCT] *<target-list>*
FROM *<relation-list>*
WHERE *<qualification>*

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of *<relation-list>*
  - Discard resulting tuples if they fail *<qualifications>*
  - Delete attributes that are not in *<target-list>*
  - If DISTINCT is specified, eliminate duplicate rows
- This strategy is probably the least efficient way to compute a query!
  - An optimizer will find more efficient strategies to compute the same answers

50

Example of Conceptual Evaluation

SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

What does this query return?

51

Example of Conceptual Evaluation

SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Step 1: Form "**cross product**" of Sailor and Reserves

sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	10/10/96
22	dustin	7	45	58	103	11/12/96
31	lubber	8	55	22	101	10/10/96
31	lubber	8	55	58	103	11/12/96
58	rusty	10	35	22	101	10/10/96
58	rusty	10	35	58	103	11/12/96

52

Example of Conceptual Evaluation

SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Step 2: Discard tuples that do not satisfy <qualification>

sid	sname	rating	age	sid	bid	day
~~22~~	~~dustin~~	~~7~~	~~45~~	~~22~~	~~101~~	~~10/10/96~~
~~22~~	~~dustin~~	~~7~~	~~45~~	~~58~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~22~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~58~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~22~~	~~101~~	~~10/10/96~~
58	rusty	10	35	58	103	11/12/96

53

Example of Conceptual Evaluation

SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Step 3: Select the specified attribute(s)

sid	sname	rating	age	sid	bid	day
~~22~~	~~dustin~~	~~7~~	~~45~~	~~22~~	~~101~~	~~10/10/96~~
~~22~~	~~dustin~~	~~7~~	~~45~~	~~58~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~22~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~58~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~22~~	~~101~~	~~10/10/96~~
58	rusty	10	35	58	103	11/12/96

54

Recap

3 **SELECT S.sname**
1 FROM Sailors S, Reserves R
2 WHERE S.sid=R.sid AND R.bid=103

Always start from "FROM" -- form cross product
Apply "WHERE" -- filter out some tuples (rows)
Apply "SELECT" -- filter out some attributes (columns)

Ques. Does this get evaluated this way in practice in a Database Management System (DBMS)?

No! This is conceptual evaluation for finding what is correct!
We will learn about join and other operator algorithms later

55

A Note on "Range Variables"

- Sometimes used as a short-name
- The previous query can also be written as:

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND bid=103
```

OR

```
SELECT sname
FROM   Sailors, Reserves
WHERE  Sailors.sid=Reserves.sid
       AND bid=103
```

It is good style, however, to use range variables always!

56

A Note on "Range Variables"

- Really needed only if the same relation appears twice in the FROM clause (called self-joins)
- Find pairs of Sailors of same age

```
SELECT S1.sname, S2. name
FROM   Sailors S1, Sailors S2
WHERE  S1.age = S2.age AND S1.sid < S2.sid
```

Why do we need the 2nd condition?

57

Find sailor ids who've reserved at least one boat

```
SELECT ????
FROM  Sailors S, Reserves R
WHERE  S.sid=R.sid
```

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

58

Find sailor ids who've reserved at least one boat

```
SELECT ???
FROM  Sailors S, Reserves R
WHERE  S.sid=R.sid
```

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

- Would adding DISTINCT to this query make a difference?

59

Find sailors who've reserved at least one boat

```
SELECT ???
FROM  Sailors S, Reserves R
WHERE  S.sid=R.sid
```

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

- Would adding DISTINCT to this query make a difference?
- What is the effect of replacing *S.sid* by *S.sname* in the SELECT clause?

60

Simple Aggregate Operators

Check yourself:
What do these queries compute?

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)
single column

```
SELECT COUNT (*)
FROM Sailors S
```

```
SELECT AVG (S.age)
FROM Sailors S
WHERE S.rating=10
```

```
SELECT S.sname
FROM Sailors S
WHERE S.rating= (SELECT MAX(S2.rating)
                 FROM Sailors S2)
```

```
SELECT COUNT (DISTINCT S.rating)
FROM Sailors S
WHERE S.sname='Bob'
```

```
SELECT AVG ( DISTINCT S.age)
FROM Sailors S
WHERE S.rating=10
```


61

Next: different types of joins

- Theta-join
- Equi-join
- Natural join
- Outer Join

62

Condition/Theta Join

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid=R.sid and age >= 40
```

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Form cross product, discard rows that do not satisfy the condition

sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	10/10/96
~~22~~	~~dustin~~	~~7~~	~~45~~	~~58~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~22~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~58~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~22~~	~~101~~	~~10/10/96~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~58~~	~~103~~	~~11/12/96~~

sid	bid	day
22	101	10/10/96
58	103	11/12/96

63

Equi Join

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid=R.sid and age = 45
```

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

A special case of theta join
Join condition only has equality predicate =

sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	10/10/96
~~22~~	~~dustin~~	~~7~~	~~45~~	~~58~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~22~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~58~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~22~~	~~101~~	~~10/10/96~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~58~~	~~103~~	~~11/12/96~~

sid	bid	day
22	101	10/10/96
58	103	11/12/96

64

Natural Join

```
SELECT *
FROM Sailors S NATURAL JOIN Reserves R
```

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

A special case of equi join
Equality condition on ALL common predicates (sid)
Duplicate columns are eliminated

sid	sname	rating	age	bid	day
22	dustin	7	45	101	10/10/96
~~22~~	~~dustin~~	~~7~~	~~45~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~101~~	~~10/10/96~~
58	rusty	10	35	103	11/12/96

sid	bid	day
22	101	10/10/96
58	103	11/12/96

65

Outer Join

```
SELECT S.sid, R. bid
FROM Sailors S LEFT OUTER JOIN Reserves R
ON S.sid=R.sid
```

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Preserves all tuples from the left table whether or not there is a match
if no match, fill attributes from right with null
Similarly RIGHT/FULL outer join

sid	bid	day
22	101	10/10/96
58	103	11/12/96

sid	bid
22	101
31	null
58	103

66

Expressions and Strings

```
SELECT S.age,    age1=S.age-5,    2*S.age AS age2
FROM  Sailors S
WHERE  S.sname LIKE 'B_%B'
```

- Illustrates use of arithmetic expressions and string pattern matching
- *Find triples (of ages of sailors and two fields defined by expressions) for sailors*
  - *whose names begin and end with B and contain at least three characters*
- LIKE is used for string matching. `_' stands for any one character and `%' stands for 0 or more arbitrary characters
  - You will need these often

67

Find sid's of sailors who've reserved a red or a green boat

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

- UNION: Can be used to compute the union of any two *union-compatible* sets of tuples
  - can themselves be the result of SQL queries
- If we replace OR by AND in the first version, what do we get?
- Also available: EXCEPT (What do we get if we replace UNION by EXCEPT?)

68

Find sid's of sailors who've reserved a red and a green boat

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

69

Find sid's of sailors who've reserved a red and a green boat

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

- INTERSECT: Can be used to compute the intersection of any two union-compatible sets of tuples.
  - Included in the SQL/92 standard, but some systems don't support it

70

Nested Queries

Find names of sailors who've reserved boat #103:

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.sid IN  (SELECT  R.sid
                  FROM  Reserves R
                  WHERE  R.bid=103)
```

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

- A very powerful feature of SQL:
  - a WHERE/FROM/HAVING clause can itself contain an SQL query
- To find sailors who've not reserved #103, use NOT IN.
- To understand semantics of nested queries, think of a nested loops evaluation
  - For each Sailors tuple, check the qualification by computing the subquery

71

Nested Queries with Correlation

Find names of sailors who've reserved boat #103:

```
SELECT  S.sname
FROM  Sailors S
WHERE  EXISTS (SELECT  *
               FROM  Reserves R
               WHERE  R.bid=103 AND S.sid=R.sid)
```

- EXISTS is another set comparison operator, like IN
- Illustrates why, in general, subquery must be re-computed for each Sailors tuple

72

Nested Queries with Correlation

Find names of sailors who've reserved boat #103 at most once:

```
SELECT S.sname
FROM Sailors S
WHERE UNIQUE (SELECT R.bid
              FROM Reserves R
              WHERE R.bid=103 AND S.sid=R.sid)
```

- If UNIQUE is used, and * is replaced by *R.bid*, finds sailors with at most one reservation for boat #103
  - UNIQUE checks for duplicate tuples

73

More on Set-Comparison Operators

- We've already seen IN, EXISTS and UNIQUE
- Can also use NOT IN, NOT EXISTS and NOT UNIQUE.
- Also available: *op* ANY, *op* ALL, *op* IN
  - where op : >, <, =, <=, >=
- Find sailors whose rating is greater than that of some sailor called Horatio
  - similarly ALL

```
SELECT *
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
                      FROM Sailors S2
                      WHERE S2.sname='Horatio')
```


74

Summary

- Relational Data
- SQL
  - Semantic
  - Join
  - Simple Aggregates
  - Nested Queries

75