SQL: The Query Language

Today's course

- Database Management Systems (DBMS) store and manage large quantities of data
- We want an **intuitive** way to ask **questions** to it!
  - For this course: questions → **queries**
- You have been taught *procedural languages* (C, java)
  - which specify **how** to solve a problem (or answer a question)
- Today we will talk about **SQL**
- SQL is a **declarative query** language
  - We ask **what we want** and the DBMS is going to deliver!

Introduction to SQL

- SQL is a relational **query language**
- Supports **simple** yet **powerful** *querying* of data
- It has two parts:
  - DDL: Data Definition Language (define and modify schema)
    - More about that in the next lecture
  - DML: Data Manipulation Language (**intuitively** query data)

Let's agree on some terminology

- Relation (or table)

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

- Row (or tuple)

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

- Column (or attribute)

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

Let's agree on some terminology

- Primary Key (PK)

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

- The PK of a relation is the column (or the group of columns) that can uniquely define a row.

- In other words:

Two rows **cannot** have the same PK

The simplest SQL query

- "Find all contents of a table"
- In this example: "Find all info for all students"

```
SELECT *
FROM Students S
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53777	White	white@cs	19	4.0

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

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

Show specific columns

- "Find name and login for all students"

```
SELECT S.name, S.login
FROM Students S
```

name	login
Jones	jones@cs
Smith	smith@ee
White	white@cs

This is called: "**Project** name and login from table Students"

Show specific rows

- "Find all 18 year old students"

```
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

This is called: "**Select** students with age 18."

Clauses of a SQL query

- Conceptually, a SQL query can be computed:
  1. **FROM** : compute *cross-product* of tables (e.g., Students and Enrolled).
  2. **WHERE** : Check conditions, discard tuples that fail. (called "selection").
  3. **SELECT** : Delete unwanted fields. (called "projection").
  4. If **DISTINCT** specified, eliminate duplicate rows.

SQL Vocabulary

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

- *relation-list* : A list of relation names
  - possibly with a *range-variable* after each name
- *qualification* : Comparisons combined using AND, OR, and NOT.
  - Comparisons are of the form: Attr *op* Const or Attr1 *op* Attr2, where *op* is one of AND, OR, and NOT
- *target-list* : A list of attributes of tables in *relation-list*
- *DISTINCT*: optional keyword indicating that the answer should not contain duplicates.
  - In SQL SELECT, the default is that duplicates are *not* eliminated! (Result is called a "multiset")

Querying Multiple Relations

- Can specify a join over two tables as follows:
  - Find all students with grade “B”

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

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

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

result =

S.name	E.cid
Jones	History105

(Naïve) query evaluation in steps

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

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

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

Step 1 – Cross Product

Combine with cross-product all tables of the **FROM** clause.

S.sid	S.name	S.login	S.age	S.gpa	E.sid	E.cid	E.grade
53666	Jones	jones@cs	18	3.4	53831	Carnatic101	C
53666	Jones	jones@cs	18	3.4	53832	Reggae203	B
53666	Jones	jones@cs	18	3.4	53650	Topology112	A
53666	Jones	jones@cs	18	3.4	53666	History105	B
53688	Smith	smith@ee	18	3.2	53831	Carnatic101	C
53688	Smith	smith@ee	18	3.2	53831	Reggae203	B
53688	Smith	smith@ee	18	3.2	53650	Topology112	A
53688	Smith	smith@ee	18	3.2	53666	History105	B

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

Step 2 - Discard tuples that fail predicate

Make sure the **WHERE** clause is true!

S.sid	S.name	S.login	S.age	S.gpa	E.sid	E.cid	E.grade
53666	Jones	jones@cs	18	3.4	53831	Carnatic101	C
53666	Jones	jones@cs	18	3.4	53832	Reggae203	B
53666	Jones	jones@cs	18	3.4	53650	Topology112	A
53666	Jones	jones@cs	18	3.4	53666	History105	B
53688	Smith	smith@ee	18	3.2	53831	Carnatic101	C
53688	Smith	smith@ee	18	3.2	53831	Reggae203	B
53688	Smith	smith@ee	18	3.2	53650	Topology112	A
53688	Smith	smith@ee	18	3.2	53666	History105	B

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

Step 3 - Discard Unwanted Columns

Show only what is on the **SELECT** clause.

S.sid	S.name	S.login	S.age	S.gpa	E.sid	E.cid	E.grade
53666	Jones	jones@cs	18	3.4	53831	Carnatic101	C
53666	Jones	jones@cs	18	3.4	53832	Reggae203	B
53666	Jones	jones@cs	18	3.4	53650	Topology112	A
53666	Jones	jones@cs	18	3.4	53666	History105	B
53688	Smith	smith@ee	18	3.2	53831	Carnatic101	C
53688	Smith	smith@ee	18	3.2	53831	Reggae203	B
53688	Smith	smith@ee	18	3.2	53650	Topology112	A
53688	Smith	smith@ee	18	3.2	53666	History105	B

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

Now the Details…

We will use these instances of relations in our examples.

Reserves

sid	bid	day
22	101	10/10/96
95	103	11/12/96

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
95	Bob	3	63.5

Insert Into Table

- Insert a row into a table

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

```
INSERT
INTO Students
VALUES (53777, White, white@cs, 19, 4.0)

INSERT
INTO Students
(sid, name, login, age, gpa)
VALUES (53777, White, white@cs, 19, 4.0)
```

Insert Into Table

- Insert a row into a table

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53777	White	white@cs	19	4.0

```
INSERT
INTO Students
VALUES (53777, White, white@cs, 19, 4.0)

INSERT
INTO Students
(sid, name, login, age, gpa)
VALUES (53777, White, white@cs, 19, 4.0)
```

Delete From Table

- Delete a row from a table

```
DELETE
FROM Students S
WHERE sid=53777
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53777	White	white@cs	19	4.0

Delete From Table

- Delete a row from a table

```
DELETE
FROM Students S
WHERE sid=53777
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53777	White	white@cs	19	4.0

Delete From Table

- Delete a row from a table

```
DELETE
FROM Students S
WHERE sid=53777
```

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

Aggregate Operators

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

single column

- Significant extension of relational algebra.
- *Find the number of sailors*

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

- *Find average age of sailors whose rating is 10*

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

- *Count unique ratings of sailors whose name is Bob*

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

Find name and age of the oldest sailor(s)

- The first query is incorrect!
  - Max returns a ***single value*** over the specified column.
  - How to combine it with appropriate *sname*?
- Third query equivalent to second query

```
SELECT S.sname, MAX (S.age)
FROM Sailors S
```

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

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

GROUP BY

- So far, we've applied aggregate operators to all (qualifying) tuples.
  - Sometimes, we want to apply them to each of several *groups* of tuples.
- Consider: *Find the age of the youngest sailor for each rating level.*
  - In general, we don't know how many rating levels exist, and what the rating values for these levels are!
  - Suppose we know that rating values go from 1 to 10; we can write 10 queries that look like this (!):

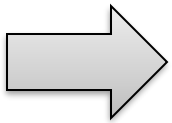
For i = 1, 2, ... , 10:

```
SELECT  MIN (S.age)
FROM  Sailors S
WHERE  S.rating = i
```

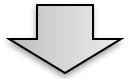
GROUP BY

- Consider: *Find the age of the youngest sailor for each rating level*

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
95	Bob	3	63.5
52	Smith	3	74.5
68	John	7	56.5
81	Ana	7	71.5



sid	sname	rating	age
22	Dustin	7	45.0
68	John	7	56.5
81	Ana	7	71.5
31	Lubber	8	55.5
95	Bob	3	63.5
52	Smith	3	74.5



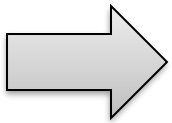
```
SELECT  MIN (S.age), S.rating
FROM  Sailors S
GROUP BY S.rating
```

sid	sname	rating	age
Not aggregated		7	45.0
		8	55.5
		3	63.5

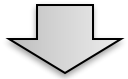
GROUP BY – HAVING

- Consider: *Find the age of the youngest sailor for each rating level with at least two sailors*

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
95	Bob	3	63.5
52	Smith	3	74.5
68	John	7	56.5
81	Ana	7	71.5



sid	sname	rating	age
22	Dustin	7	45.0
68	John	7	56.5
81	Ana	7	71.5
31	Lubber	8	55.5
95	Bob	3	63.5
52	Smith	3	74.5



```
SELECT  MIN (S.age), S.rating
FROM  Sailors S
GROUP BY S.rating
HAVING COUNT(*) > 1
```

sid	sname	rating	age
Not aggregated		7	45.0
		3	55.5

Summary

- An advantage of the relational model is its well-defined query semantics.
- SQL provides functionality close to that of the basic relational model.
  - some differences in duplicate handling, null values, set operators, etc.
- Typically, many ways to write a query
  - the system is responsible for figuring a fast way to actually execute a query regardless of how it is written.
- Lots more functionality beyond these basic features.

Architecture of a (typical) DBMS