Today’s Plan

- Reading Homework 2

- SQL (Chapter 3)
  - Null values (3.6)
  - Aggregates (3.7)
  - Views (4.2)
  - Transactions (4.3)
  - Integrity Constraints (4.4)
  - Triggers (5.3)
The “dirty little secret” of SQL
(major headache for query optimization)

Can be a value of any attribute

\[ \text{e.g: branch} = \]

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>Boston</td>
<td>9M</td>
</tr>
<tr>
<td>Perry</td>
<td>Horseneck</td>
<td>1.7M</td>
</tr>
<tr>
<td>Mianus</td>
<td>Horseneck</td>
<td>.4M</td>
</tr>
<tr>
<td>Waltham</td>
<td>Boston</td>
<td>NULL</td>
</tr>
</tbody>
</table>

What does this mean?

(unknown) We don’t know Waltham’s assets?
(inapplicable) Waltham has a special kind of account without assets
(withheld) We are not allowed to know
SQL: Nulls

Arithmetic Operations with Null

\[ n + \text{NULL} = \text{NULL} \]

(similarly for all arithmetic ops: +, -, *, /, mod, …)

e.g: branch =

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>Boston</td>
<td>9M</td>
</tr>
<tr>
<td>Perry</td>
<td>Horseneck</td>
<td>1.7M</td>
</tr>
<tr>
<td>Mianus</td>
<td>Horseneck</td>
<td>.4M</td>
</tr>
<tr>
<td>Waltham</td>
<td>Boston</td>
<td>NULL</td>
</tr>
</tbody>
</table>

SELECT bname, assets * 2 as a2
FROM branch

<table>
<thead>
<tr>
<th>bname</th>
<th>a2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>18M</td>
</tr>
<tr>
<td>Perry</td>
<td>3.4M</td>
</tr>
<tr>
<td>Mianus</td>
<td>.8M</td>
</tr>
<tr>
<td>Waltham</td>
<td>NULL</td>
</tr>
</tbody>
</table>
Boolean Operations with **Null**

\[ n < \text{NULL} = \text{UNKNOWN} \quad \text{(similarly for all boolean ops: \(>, \leq, \geq, \neq, =, \ldots\))} \]

e.g: \( \text{branch} = \)

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>Boston</td>
<td>9M</td>
</tr>
<tr>
<td>Perry</td>
<td>Horseneck</td>
<td>1.7M</td>
</tr>
<tr>
<td>Mianus</td>
<td>Horseneck</td>
<td>.4M</td>
</tr>
<tr>
<td>Waltham</td>
<td>Boston</td>
<td>NULL</td>
</tr>
</tbody>
</table>

```
SELECT * =
FROM branch
WHERE assets = NULL
```

**Counter-intuitive:** \( \text{NULL} \times 0 = \text{NULL} \)

**Counter-intuitive:** select * from movies
where length \(\geq\) 120 or length \(\leq\) 120
SQL: Nulls

Boolean Operations with Null

\[ n < \text{NULL} = \text{UNKNOWN} \] (similarly for all boolean ops: \(>, \leq, \geq, \neq, =, \ldots\))

\[
e.g.: \text{branch} = \begin{array}{|c|c|c|}
\hline
\text{bname} & \text{bcity} & \text{assets} \\
\hline
\text{Downtown} & \text{Boston} & 9M \\
\text{Perry} & \text{Horseneck} & 1.7M \\
\text{Mianus} & \text{Horseneck} & .4M \\
\text{Waltham} & \text{Boston} & \text{NULL} \\
\hline
\end{array}
\]

\[
\text{SELECT} \ast \\
\text{FROM} \text{branch} \\
\text{WHERE} \text{assets IS NULL}
\]
Boolean Operations with Unknown

n < NULL = UNKNOWN  (similarly for all boolean ops: >, <=, >=, <>, =, ...)

FALSE OR UNKNOWN = UNKNOWN
TRUE AND UNKNOWN = UNKNOWN

Intuition: substitute each of TRUE, FALSE for unknown. If different answer results, results is unknown

UNKNOWN OR UNKNOWN = UNKNOWN
UNKNOWN AND UNKNOWN = UNKNOWN
NOT (UNKNOWN) = UNKNOWN

Can write:

SELECT ...
FROM ...
WHERE booleanexp IS UNKNOWN

UNKNOWN tuples are not included in final result
Today’s Plan

- Reading Homework 2

- SQL (Chapter 3)
  - Null values (3.6)
  - Aggregates (3.7)
  - Views (4.2)
  - Transactions (4.3)
  - Integrity Constraints (4.4)
  - Triggers (5.3)
Aggregates

Find the average salary of instructors in the Computer Science

```sql
select avg(salary)
from instructor
where dept_name = 'Comp. Sci';
```

Other common aggregates:
max, min, sum, count, stdev, ...

```sql
select count (distinct ID)
from teaches
where semester = 'Spring' and year = 2010
```

Can specify aggregates in any query.
Find max salary over instructors teaching in S’10

```sql
select max(salary)
from teaches natural join instructor
where semester = 'Spring' and year = 2010;
```

Aggregate result can be used as a scalar.
Find instructors with max salary:

```sql
select *
from instructor
where salary = (select max(salary) from instructor);
```
Aggregate result can be used as a scalar.
Find instructors with max salary:

```sql
select *
from instructor
where salary = (select max(salary) from instructor);
```

Following doesn’t work:

```sql
select *
from instructor
where salary = max(salary);
```

```sql
select name, max(salary)
From instructor;
```
Split the tuples into groups, and compute the aggregate for each group

```
select dept_name, avg(salary)
from instructor
group by dept_name;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>Elec. Eng.</td>
<td>80000</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
<td>Finance</td>
<td>80000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>60000</td>
</tr>
<tr>
<td>58583</td>
<td>Califieri</td>
<td>History</td>
<td>62000</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
</tr>
<tr>
<td>33456</td>
<td>Gold</td>
<td>Physics</td>
<td>87000</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dept_name</th>
<th>avg_salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biology</td>
<td>72000</td>
</tr>
<tr>
<td>Comp. Sci.</td>
<td>77333</td>
</tr>
<tr>
<td>Elec. Eng.</td>
<td>80000</td>
</tr>
<tr>
<td>Finance</td>
<td>85000</td>
</tr>
<tr>
<td>History</td>
<td>61000</td>
</tr>
<tr>
<td>Music</td>
<td>40000</td>
</tr>
<tr>
<td>Physics</td>
<td>91000</td>
</tr>
</tbody>
</table>
Figure 3.8 The natural join of the instructor relation with the teaches relation.

Output will have 3 tuples:
Summer, ....
Fall, ....
Spring, ...

Chapter 3
Introduction to SQL

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
<th>course_id</th>
<th>sec_id</th>
<th>semester</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
<td>CS-101</td>
<td>1</td>
<td>Fall</td>
<td>2009</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
<td>CS-315</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
<td>CS-347</td>
<td>1</td>
<td>Fall</td>
<td>2009</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>FIN-201</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>MU-199</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>PHY-101</td>
<td>1</td>
<td>Fall</td>
<td>2009</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>60000</td>
<td>HIS-351</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
<td>CS-101</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
<td>CS-319</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
<td>BIO-101</td>
<td>1</td>
<td>Summer</td>
<td>2009</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
<td>BIO-301</td>
<td>1</td>
<td>Summer</td>
<td>2010</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
<td>CS-190</td>
<td>1</td>
<td>Spring</td>
<td>2009</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
<td>CS-190</td>
<td>2</td>
<td>Spring</td>
<td>2009</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
<td>CS-319</td>
<td>2</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>Elec. Eng.</td>
<td>80000</td>
<td>EE-181</td>
<td>1</td>
<td>Spring</td>
<td>2009</td>
</tr>
</tbody>
</table>
Aggregates: Group By

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
<th>course_id</th>
<th>sec_id</th>
<th>semester</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
<td>CS-101</td>
<td>1</td>
<td>Fall</td>
<td>2009</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
<td>CS-315</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
<td>CS-347</td>
<td>1</td>
<td>Fall</td>
<td>2009</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>FIN-201</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>MU-199</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>PHY-101</td>
<td>1</td>
<td>Fall</td>
<td>2009</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>60000</td>
<td>HIS-351</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
<td>CS-101</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
<td>CS-319</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
<td>BIO-101</td>
<td>1</td>
<td>Summer</td>
<td>2009</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
<td>BIO-301</td>
<td>1</td>
<td>Summer</td>
<td>2010</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
<td>CS-190</td>
<td>1</td>
<td>Spring</td>
<td>2009</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
<td>CS-190</td>
<td>2</td>
<td>Spring</td>
<td>2009</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
<td>CS-319</td>
<td>2</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>Elec. Eng.</td>
<td>80000</td>
<td>EE-181</td>
<td>1</td>
<td>Spring</td>
<td>2009</td>
</tr>
</tbody>
</table>

Output will have 2 tuples:
2009,
2010,

Figure 3.8  The natural join of the instructor relation with the teaches relation.
Aggregates: Group By

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
<th>course_id</th>
<th>sec_id</th>
<th>semester</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
<td>CS-101</td>
<td>1</td>
<td>Fall</td>
<td>2009</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
<td>CS-315</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
<td>CS-347</td>
<td>1</td>
<td>Fall</td>
<td>2009</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
<td>FIN-201</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
<td>MU-199</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
<td>PHY-101</td>
<td>1</td>
<td>Fall</td>
<td>2009</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>60000</td>
<td>HIS-351</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
<td>CS-101</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
<td>CS-319</td>
<td>1</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
<td>BIO-101</td>
<td>1</td>
<td>Summer</td>
<td>2009</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
<td>BIO-301</td>
<td>1</td>
<td>Summer</td>
<td>2010</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
<td>CS-190</td>
<td>1</td>
<td>Spring</td>
<td>2009</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
<td>CS-190</td>
<td>2</td>
<td>Spring</td>
<td>2009</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
<td>CS-319</td>
<td>2</td>
<td>Spring</td>
<td>2010</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>Elec. Eng.</td>
<td>80000</td>
<td>EE-181</td>
<td>1</td>
<td>Spring</td>
<td>2009</td>
</tr>
</tbody>
</table>

Output will have 7 tuples:

Comp. Sci, Finance, Music, Physics, History, Biology, Elec. Eng.,

Figure 3.8  The natural join of the instructor relation with the teaches relation.
Aggregates: Group By

Attributes in the select clause must be aggregates, or must appear in the group by clause. Following wouldn’t work:

```
select dept_name, ID, avg(salary)
from instructor
group by dept_name;
```

“having” can be used to select only some of the groups:

```
select dept_name
from instructor
group by dept_name
having avg(salary) > 42000
```
Given

\[
\text{branch} =
\]

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>Boston</td>
<td>9M</td>
</tr>
<tr>
<td>Perry</td>
<td>Horseneck</td>
<td>1.7M</td>
</tr>
<tr>
<td>Mianus</td>
<td>Horseneck</td>
<td>.4M</td>
</tr>
<tr>
<td>Waltham</td>
<td>Boston</td>
<td>NULL</td>
</tr>
</tbody>
</table>

**Aggregate Operations**

\[
\text{SELECT} \ \text{SUM} \ (\text{assets}) = \ \text{SUM} \ 11.1 \ M
\]

**NULL is ignored for SUM**

*Same for AVG (3.7M), MIN (0.4M), MAX (9M)*

Also for COUNT(assets) -- returns 3

**But COUNT (*) returns**

<table>
<thead>
<tr>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
Aggregates and NULLs

Given

\[
\text{branch} = \begin{array}{ccc}
\text{bname} & \text{bcity} & \text{assets} \\
\end{array}
\]

\[
\text{SELECT SUM (assets) = } \begin{array}{c}
\text{SUM} \\
\text{NULL} \\
\end{array} \\
\text{FROM branch}
\]

- Same as AVG, MIN, MAX
- But COUNT (assets) returns 0
Today’s Plan

- Reading Homework 2

- SQL (Chapter 3)
  - Null values (3.6)
  - Aggregates (3.7)
  - Views (4.2)
  - Transactions (4.3)
  - Integrity Constraints (4.4)
  - Triggers (5.3)
Today’s Plan

- Reading Homework 2

- SQL (Chapter 3)
  - Null values (3.6)
  - Aggregates (3.7)
  - Views (4.2)
  - Transactions (4.3)
  - Integrity Constraints (4.4)
  - Triggers (5.3)
Provide a mechanism to hide certain data from the view of certain users. To create a view we use the command:

```
create view v as <query expression>
```

where:

- `<query expression>` is any legal expression
- The view name is represented by `v`

- Can be used in any place a normal table can be used
- For users, there is no distinction in terms of using it
Example Queries

- A view consisting of branches and their customers

create view all-customers as
(select branch-name, customer-name
from depositor, account
where depositor.account-number = account.account-number)
union
(select branch-name, customer-name
from borrower, loan
where borrower.loan-number = loan.loan-number)

Find all customers of the Perryridge branch

select customer-name
from all-customers
where branch-name = 'Perryridge'
Today’s Plan

- Reading Homework 2

- SQL (Chapter 3)
  - Null values (3.6)
  - Aggregates (3.7)
  - Views (4.2)
  - Transactions (4.3)
  - Integrity Constraints (4.4)
  - Triggers (5.3)
Next:

- Integrity constraints
- ??
- Prevent semantic inconsistencies
IC’s

- Predicates on the database
- Must always be true (checked whenever db gets updated)

- There are the following 4 types of IC’s:
  - **Key constraints** (1 table)
    - e.g., 2 accts can’t share the same acct_no
  - **Attribute constraints** (1 table)
    - e.g., accts must have nonnegative balance
  - **Referential Integrity constraints** (2 tables)
    - E.g. bnames associated w/ loans must be names of real branches
  - **Global Constraints** (n tables)
    - E.g., all loans must be carried by at least 1 customer with a savings acct
Key Constraints

Idea: specifies that a relation is a set, not a bag

SQL examples:

1. **Primary Key:**
   
   ```sql
   CREATE TABLE branch(
       bname  CHAR(15)  PRIMARY KEY,
       bcity      CHAR(20),
       assets    INT);
   
   or
   
   CREATE TABLE depositor(
       cname   CHAR(15),
       acct_no  CHAR(5),
       PRIMARY KEY(cname, acct_no));
   ```

2. **Candidate Keys:**
   
   ```sql
   CREATE TABLE customer (  
       ssn     CHAR(9)    PRIMARY KEY,
       cname  CHAR(15),
       address CHAR(30),
       city          CHAR(10),
       UNIQUE (cname, address, city));
   ```