Announcements

• WQ1 due on Sunday (night)
  – Any issues?

• HW1 due on next Tuesday (June 27)

• Office Hours
  – Moving to 2nd floor breakout
Announcements

• WQ1 due on Sunday night
  Any issues?
  Trevor : Monday 10:00 – 12:00
  Ryan : Tuesday 11:30 – 12:30
  Rob : Friday 1:00 – 2:00

• Trevor : Wednesday  11:00 – 1:00 (CSE 220)
  – Moving to 2nd floor breakout
Review

• Relational data model
  – Instance and schema
• SQL for manipulating relational data
  – Create tables
  – Retrieve records from tables
  – Declare keys and foreign keys
Review

• SQL is declarative
  – Say what you want not how to do it
• Tables are FLAT
  – No nested attributes
• Tables DO NOT prescribe how they are implemented / stored on disk
  – This is called physical data independence
Relation Schema

• Names and types form part of the table “schema”:

  Company(cname, country, no_employees, for_profit)

  Company(cname: varchar(30), country: char(20),
           no_employees: int, for_profit: char(1))

• Instance

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>Y</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>
Adding Attributes

Let’s add a list of products that each company produces

- How? Recall that tables are flat!

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>Y</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>
Foreign Keys

• A column (or columns) whose value is a key of another table (Must be unique!)
  – i.e., a reference to another row in another table
Foreign Keys

- A column (or columns) whose value is a key of another table (Must be unique!)
  - i.e., a reference to another row in another table

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>Y</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>

Product(pname, price, category, mname, mcountry,
FOREIGN KEY (mname, mcountry)
REFERENCES company(cname, country)
)
Foreign Keys

Company(cname, country, no_employees, for_profit)

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>Y</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>

Product( pname, price, category, cname, country, 
FOREIGN KEY (cname,country) 
REFERENCES Company(cname,country) )

<table>
<thead>
<tr>
<th>pname</th>
<th>price</th>
<th>category</th>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleTouch</td>
<td>149.99</td>
<td>photography</td>
<td>Canon</td>
<td>Japan</td>
</tr>
<tr>
<td>AC</td>
<td>300</td>
<td>Appliance</td>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>
Best Practice: Use Integer Primary Key

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>Y</td>
</tr>
<tr>
<td>2</td>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>

Product(pid INTEGER PRIMARY KEY, pname, price, category, cid REFERENCES Company.cid)

<table>
<thead>
<tr>
<th>pid</th>
<th>pname</th>
<th>price</th>
<th>category</th>
<th>cid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SingleTouch</td>
<td>149.99</td>
<td>photography</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>AC</td>
<td>300</td>
<td>Appliance</td>
<td>2</td>
</tr>
</tbody>
</table>

“All problems in computer science can be solved by another level of indirection”
A Note On Multiple Keys

Company(cname: varchar(30) PRIMARY KEY NOT NULL,
country: char(20),
no_employees: int,
for_profit: char(1) NOT NULL
);

A Note On Multiple Keys

Company(cname: varchar(30) PRIMARY KEY NOT NULL,
country: char(20),
no_employees: int,
for_profit: char(1)
);

Company(cname: varchar(30) NOT NULL,
country: char(20) NOT NULL,
no_employees: int,
for_profit: char(1),
PRIMARY KEY (cname, country)
);

Same for UNIQUE and FOREIGN KEY
Today

• SQL Basics
  – Selection
  – Projection
  – Ordering
  – Joins
Demo

Setup Database
Selections in SQL

• Condition on the WHERE clause to filter returned tuples.

```sql
SELECT * 
FROM Product 
WHERE price > 100.0
```

• Condition operators:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not equal to</td>
</tr>
<tr>
<td>IS</td>
<td>Is</td>
</tr>
<tr>
<td>IS NOT</td>
<td>Is not</td>
</tr>
<tr>
<td>IN</td>
<td>In</td>
</tr>
<tr>
<td>LIKE</td>
<td>Like</td>
</tr>
<tr>
<td>GLOB</td>
<td>GLOB</td>
</tr>
<tr>
<td>MATCH</td>
<td>MATCH</td>
</tr>
<tr>
<td>REGEXP</td>
<td>REGEXP</td>
</tr>
</tbody>
</table>

AND

OR
Demo

Selection
Projections in SQL

• What does the * mean in SELECT *
  – Shortcut for ALL attributes
  – What if we only want a few?

```sql
SELECT category
FROM Product
```

• Can combine with selection (build complex queries)

```sql
SELECT category
FROM Product
WHERE price > 100.0
```
DISTINCT and ORDER BY

• Query results do not have to be relations
  – i.e., they can have duplicate rows
  – remove them using DISTINCT

• Result order is normally unspecified
  – choose an order using ORDER BY
  – e.g., ORDER BY country, cname
  – e.g., ORDER BY price ASC, pname DESC
Demo

Projection, Order, Distinct
## Complex Queries

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
<td>20000</td>
<td>Y</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pname</th>
<th>price</th>
<th>category</th>
<th>cname</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperGizmo</td>
<td>250.00</td>
<td>Gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>AC</td>
<td>300</td>
<td>Appliance</td>
<td>Hitachi</td>
</tr>
</tbody>
</table>

How do we get all products made in Japan?

Need information from BOTH tables
Joins in SQL

Product(pname, price, category, manufacturer)
Company(cname, country)

SELECT pname, price
FROM   Product, Company
WHERE  Product.pname = Company.cname AND
        country='Japan' AND price < 150

• What does this query do?

Retrieve all Japanese products that cost < $150
Joins in SQL

SELECT `pname`, `price`
FROM `Product`, `Company`
WHERE `Product.cname` = `Company.cname`

What is the **cardinality** of this query?

A) 3    B) 6    C) 9    D) 27
### Joins in SQL

Retrieve all Japanese companies that manufacture products less than $100

<table>
<thead>
<tr>
<th>pname</th>
<th>price</th>
<th>cname</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiTouch</td>
<td>199.99</td>
<td>Canon</td>
</tr>
<tr>
<td>SingleTouch</td>
<td>49.99</td>
<td>Canon</td>
</tr>
<tr>
<td>SuperGizmo</td>
<td>250.00</td>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>

```sql
SELECT pname, price
FROM   Product, Company
WHERE   Company.country = 'Japan' AND Product.price < 100
```
Joins in SQL

Product\((\text{pname}, \text{price}, \text{category}, \text{cname})\)
Company\((\text{cname}, \text{country})\)

<table>
<thead>
<tr>
<th>pname</th>
<th>price</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiTouch</td>
<td>199.99</td>
<td>Canon</td>
</tr>
<tr>
<td>SingleTouch</td>
<td>49.99</td>
<td>Canon</td>
</tr>
<tr>
<td>SuperGizmo</td>
<td>250.00</td>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
</tr>
</tbody>
</table>

SELECT DISTINCT \text{cname}
FROM Product, Company
WHERE \text{country}=\text{\textquotesingle}Japan\text{\textquotesingle} AND \text{price} < 100.0
AND Product.\text{cname} = Company.\text{cname}
**Aliases**

```sql
SELECT country, pname, price 
FROM Company LEFT JOIN Product 
ON Company.cname = Product.cname
```

```
SELECT country, pname, price 
FROM Company as c 
LEFT JOIN Product as p 
ON c.cname = p.cname
```
Joins in SQL

• This query is called an **inner join**
  
  – Each row in the result **must come from both tables in the join**

```sql
SELECT DISTINCT cname
FROM   Product
INNER JOIN Company on Company.cname = Product.cname
WHERE  country='USA' AND category = 'gadget'
```

• What happens if a company makes no products?
  
  – Not returned in the results
Today

- SQL Basics
  - Selection
    - WHERE clause with condition
  - Projection
    - Field List or *
  - Ordering
  - Joins
    - Inner
SQLite SELECT
https://sqlite.org/lang_select.html