CSE 344

JANUARY 10TH – JOINS
ADMINISTRATIVE MINUTIAE

• HW1 out
  • Piazza post for getting the correct upstream assignments
• Online Quiz posted
  • 6 questions (SQL)
• Both due WED Jan 17
• OH locations posted
• Posting lectures before
ADMINISTRATIVE MINUTIAE

• Office hours
  • Jayanth: Mon 11-12, CSE 220
  • Colin: Wed 2-3, 5th floor breakout
  • Allison: Mon 1-2, CSE 025
  • Cindy: Tue 2-3, CSE 023
  • James: Tue 10-11, CSE 220
  • Jonathan: Tue 4-5, CSE 023
  • Joshua: Tue 1-2, CSE 023
RELATIONAL MODEL

Data is a collection of relations / tables:

mathematically, relation is a set of tuples

- each tuple (or entry) must have a value for each attribute
- order of the rows is unspecified

What is the schema for this table?

Company(cname, country, no_employees, for_profit)
**KEYS**

Key = one (or multiple) attributes that uniquely identify a record

<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>True</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>True</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>True</td>
</tr>
<tr>
<td>HappyCam</td>
<td>Canada</td>
<td>500</td>
<td>False</td>
</tr>
</tbody>
</table>

Is this a key?

No: future updates to the database may create duplicate no_employees
## MULTI-ATTRIBUTE KEY

Key = fName,lName

(what does this mean?)

<table>
<thead>
<tr>
<th>fName</th>
<th>lName</th>
<th>Income</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Smith</td>
<td>20000</td>
<td>Testing</td>
</tr>
<tr>
<td>Alice</td>
<td>Thompson</td>
<td>50000</td>
<td>Testing</td>
</tr>
<tr>
<td>Bob</td>
<td>Thompson</td>
<td>30000</td>
<td>SW</td>
</tr>
<tr>
<td>Carol</td>
<td>Smith</td>
<td>50000</td>
<td>Testing</td>
</tr>
</tbody>
</table>
**MULTIPLE KEYS**

We can choose one key and designate it as *primary key*.  
E.g.: primary key = SSN
FOREIGN KEY

Company(cname, country, no_employees, for_profit)
Country(name, population)

<table>
<thead>
<tr>
<th></th>
<th></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>

Country

<table>
<thead>
<tr>
<th>name</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>320M</td>
</tr>
<tr>
<td>Japan</td>
<td>127M</td>
</tr>
</tbody>
</table>
KEYS: SUMMARY

Key = columns that uniquely identify tuple

- Usually we underline
- A relation can have many keys, but only one can be chosen as primary key

Foreign key:

- Attribute(s) whose value is a key of a record in some other relation
- Foreign keys are sometimes called semantic pointer
DEMO 1

- **Common Syntax**
  - CREATE TABLE [tablename]
    ([att1] [type1],
     [att2] [type2]…);
  - INSERT INTO [tablename] VALUES ([val1],[val2]…);
  - SELECT [att1],[att2],… FROM [tablename]
    WHERE [condition]
  - DELETE FROM [tablename]
    WHERE [condition]
DEMO 2

- Two other operations we want to support
  - ALTER TABLE: Adds a new attribute to the table
  - UPDATE: Change the attribute for a particular tuple in the table.
- Common Syntax
  - ALTER TABLE [tablename] ADD [attname] [atttype]
  - UPDATE [tablename] SET [attname]=[value]
    WHERE [condition]
DISCUSSION

Tables are NOT ordered
  • they are sets or multisets (bags)

Tables are FLAT
  • No nested attributes

Tables DO NOT prescribe how they are implemented / stored on disk
  • This is called physical data independence
DISCUSSION

- Tables may not be ordered, but data can be returned in an order with the ORDER BY modifier
  - ORDER BY [attname] [DESC/ASC]
  - Supports sorting by multiple variables
DISCUSSION

• Tables may not be ordered, but data can be returned in an order with the ORDER BY modifier

• Whew, today’s been a lot of coding... I know what you’re thinking...
THEORY BREAK

- We can think of accessing information through queries as some combination of functions
THEORY BREAK

• We can think of accessing information through queries as some combination of functions
  • Consider a table of UW students (with all relevant info):
THEORY BREAK

• We can think of accessing information through queries as some combination of functions
  • Consider a table of UW students (with all relevant info):
    • How would we need to get the birth year of all UWBW students from California?
THEORY BREAK

• We can think of accessing information through queries as some combination of functions
  • Consider a table of UW students (with all relevant info):
    • How would we need to get the birth year of all UWBW students from California?
    • *Think of the file as a set of tuples*
THEORY BREAK

• We can think of accessing information through queries as some combination of functions
  • Consider a table of UW students (with all relevant info):
    • How would we need to get the birth year of all UWBW students from California?
    • *Think of the file as a set of tuples*
    • Find the set of UWBW students and the set of students from California; Find the intersection of these sets, return just the year from the birthday values of this set
• We can think of accessing information through queries as some combination of functions

• Consider a table of UW students (with all relevant info):
  • How would we need to get the birth year of all UWBW students from California?
  • *Think of the file as a set of tuples*
  • Find the set of UWBW students and the set of students from California; Find the intersection of these sets, return just the year from the birthday values of this set
  • *What does this return?*
**THEORY BREAK**

- **We can think of accessing information through queries as some combination of functions**
  - Consider a table of UW students (with all relevant info):
    - How would we need to get the birth year of all UWBW students from California?
    - *Think of the file as a set of tuples*
    - Find the set of UWBW students and the set of students from California; Find the intersection of these sets, return just the year from the birthday values of this set
    - *What does this return?*
    - Years, but with many duplicates. Even though sets don’t allow duplicates, the objects are unique.
THEORY BREAK

• If we only want to return unique elements, we can use the DISTINCT modifier
  • Even if we hide some attributes from the output, the data is all still there.
  • When we select a subset of the attributes, this function is called a projection
• This was all for a single table.
• Data models specify how our data are stored and how the data are related
• Need to utilize these relations, or the database was pointless
• This involves a JOIN
JOIN: INTRO

• The JOIN is the way we indicate in a query how multiple tables are related.
  • Example, if we want all of the products and their relevant company information, we need to join those two tables.
  • The result of the join is all of the relevant information from both tables
  • Join occurs based on the join condition.
  • This allows us to access information that comes from multiple tables
Retrieve all Japanese products that cost < $150

<table>
<thead>
<tr>
<th>company</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>

**Join in SQL**

```sql
SELECT P.pname, P.price, P.category, C.manufacturer
FROM Product P
JOIN Company C ON P.manufacturer = C.cname
WHERE C.country = 'Japan' AND P.price < 150
```
Retrieve all Japanese products that cost < $150

```
SELECT pname, price
FROM   Product, Company
WHERE  ...
```
**JOINS IN SQL**

**To retrieve all Japanese products that cost < $150:**

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

<table>
<thead>
<tr>
<th>pname</th>
<th>price</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiTouch</td>
<td>199.99</td>
<td>gadget</td>
<td>Canon</td>
</tr>
<tr>
<td>SingleTouch</td>
<td>49.99</td>
<td>photography</td>
<td>Canon</td>
</tr>
<tr>
<td>Gizom</td>
<td>50</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>SuperGizmo</td>
<td>250.00</td>
<td>gadget</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>
JOINS IN SQL

Retrieve all USA companies that manufacture "gadget" products

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Category</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiTouch</td>
<td>199.99</td>
<td>gadget</td>
<td>Canon</td>
</tr>
<tr>
<td>SingleTouch</td>
<td>49.99</td>
<td>photography</td>
<td>Canon</td>
</tr>
<tr>
<td>Gizom</td>
<td>50</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>SuperGizmo</td>
<td>250.00</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</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>
JOINS IN SQL

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

<table>
<thead>
<tr>
<th>pname</th>
<th>price</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiTouch</td>
<td>199.99</td>
<td>gadget</td>
<td>Canon</td>
</tr>
<tr>
<td>SingleTouch</td>
<td>49.99</td>
<td>photography</td>
<td>Canon</td>
</tr>
<tr>
<td>Gizom</td>
<td>50</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>SuperGizmo</td>
<td>250.00</td>
<td>gadget</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>

Retrieve all USA companies that manufacture "gadget" products

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
AND manufacturer = cname
```
JOINS IN SQL

The standard join in SQL is sometimes called an inner join

- Each row in the result must come from both tables in the join

Sometimes we want to include rows from only one of the two table: outer join
INNER JOIN

<table>
<thead>
<tr>
<th>Employee</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>1</td>
<td>Joe</td>
</tr>
<tr>
<td>2</td>
<td>Jack</td>
</tr>
<tr>
<td>3</td>
<td>Jill</td>
</tr>
</tbody>
</table>

Retrieve employees and their sales
INNER JOIN

Employee(id, name)
Sales(employeeID, productID)

Retrieve employees and their sales

Employee

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe</td>
</tr>
<tr>
<td>2</td>
<td>Jack</td>
</tr>
<tr>
<td>3</td>
<td>Jill</td>
</tr>
</tbody>
</table>

Sales

<table>
<thead>
<tr>
<th>employeeID</th>
<th>productID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>344</td>
</tr>
<tr>
<td>1</td>
<td>355</td>
</tr>
<tr>
<td>2</td>
<td>544</td>
</tr>
</tbody>
</table>

SELECT *
FROM Employee E, Sales S
WHERE E.id = S.employeeID
Employee(id, name)
Sales(employeeID, productID)

**INNER JOIN**

<table>
<thead>
<tr>
<th>Employee</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>id</strong></td>
<td><strong>name</strong></td>
</tr>
<tr>
<td>1</td>
<td>Joe</td>
</tr>
<tr>
<td>2</td>
<td>Jack</td>
</tr>
<tr>
<td>3</td>
<td>Jill</td>
</tr>
</tbody>
</table>

Retrieve employees and their sales

```sql
SELECT * 
FROM Employee E, Sales S 
WHERE E.id = S.employeeID
```
INNER JOIN

Employee

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe</td>
</tr>
<tr>
<td>2</td>
<td>Jack</td>
</tr>
<tr>
<td>3</td>
<td>Jill</td>
</tr>
</tbody>
</table>

Sales

<table>
<thead>
<tr>
<th>employeeID</th>
<th>productID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>344</td>
</tr>
<tr>
<td>1</td>
<td>355</td>
</tr>
<tr>
<td>2</td>
<td>544</td>
</tr>
</tbody>
</table>

Retrieve employees and their sales

SELECT * FROM Employee E, Sales S WHERE E.id = S.employeeID

Jill is missing
INNER JOIN

Employee(id, name)
Sales(employeID, productID)

Retrieve employees and their sales

```sql
SELECT *
FROM Employee E
INNER JOIN Sales S
ON E.id = S.employeID
```

Jill is missing

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>employeID</th>
<th>productID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe</td>
<td>1</td>
<td>344</td>
</tr>
<tr>
<td>1</td>
<td>Joe</td>
<td>1</td>
<td>355</td>
</tr>
<tr>
<td>2</td>
<td>Jack</td>
<td>2</td>
<td>544</td>
</tr>
</tbody>
</table>
Screenshot of a slide explaining an OUTER JOIN operation in SQL. The slide contains a table of employees, a table of sales, and a SQL query to retrieve employees and their sales. The slide also includes a note that Jill is present.
(INNER) JOINS

Product(pname, price, category, manufacturer)
Company(cname, country)
-- manufacturer is foreign key to Company

SELECT DISTINCT cname
FROM Product, Company
WHERE country = 'USA' AND category = 'gadget'
AND manufacturer = cname
(INNER) JOINS

SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
AND manufacturer = cname
### (INNER) JOINS

```sql
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
    AND manufacturer = cname
```

#### Product

<table>
<thead>
<tr>
<th>pname</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
</tbody>
</table>

#### Company

<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>
(INNER) JOINS

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
AND manufacturer = cname
```
(INNER) JOINS

```sql
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
AND manufacturer = cname
```
### (INNER) JOINS

```sql
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
AND manufacturer = cname
```
(INNER) JOINS

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
AND manufacturer = cname
```
**(INNER) JOINS**

```sql
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
  AND manufacturer = cname
```

```sql
SELECT DISTINCT cname
FROM Product
JOIN Company
ON country = 'USA'
  AND category = 'gadget'
  AND manufacturer = cname
```