CSE 344

JANUARY 10TH –JOINS

ADMINISTRATIVE MINUTIAE

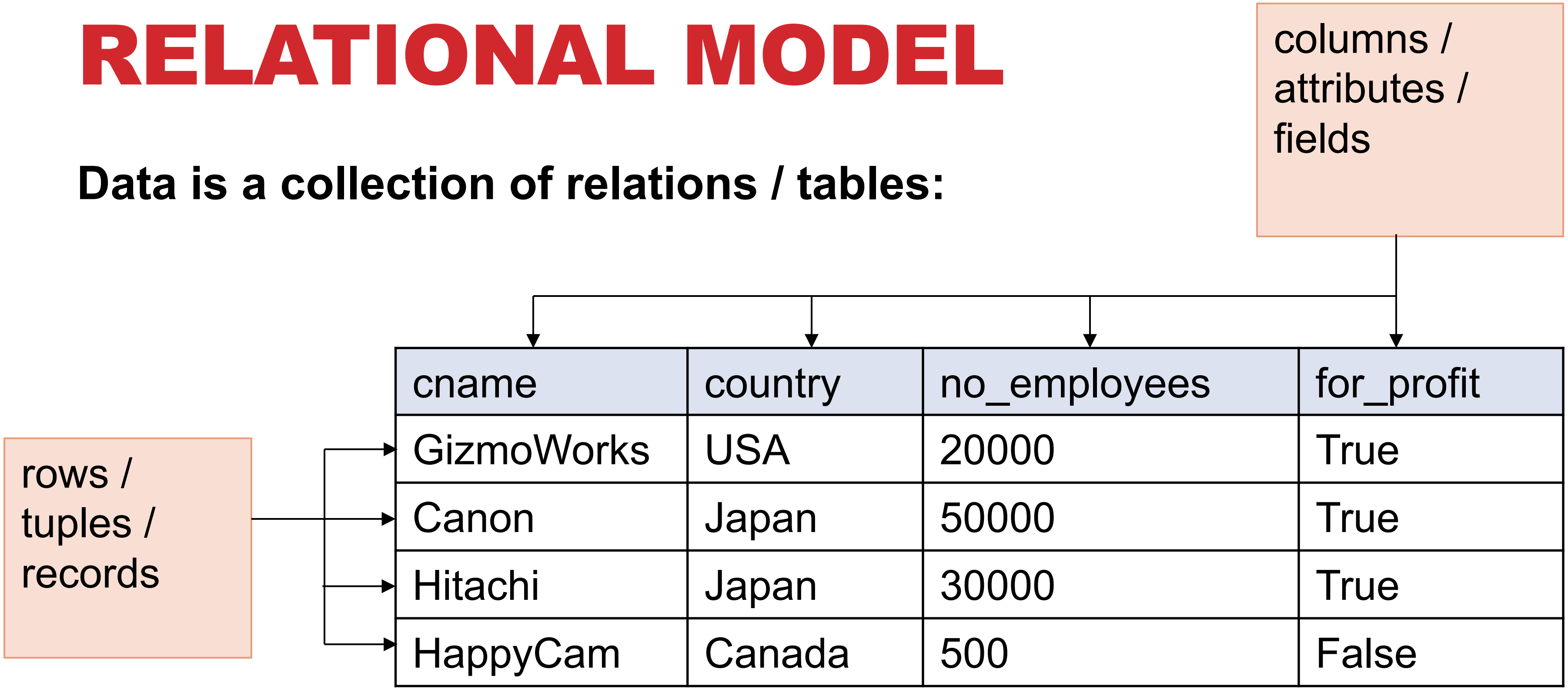
- **HW1 out**
  - Piazza post for getting the correct upstream assignments
- **Online Quiz posted**
  - 6 questions (SQL)
- **Both due WED Jan 17**
- **OH locations posted**
- **Posting lectures before**

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:



columns / attributes / fields

rows / tuples / records

cname	country	no_employees	for_profit
GizmoWorks	USA	20000	True
Canon	Japan	50000	True
Hitachi	Japan	30000	True
HappyCam	Canada	500	False

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

Key

Not a key

Is this a key?

No: future updates to the database may create duplicate no_employees

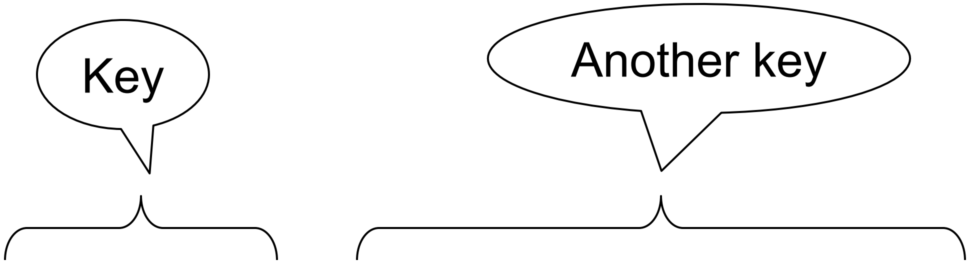
cname	country	no_employees	for_profit
GizmoWorks	USA	20000	True
Canon	Japan	50000	True
Hitachi	Japan	30000	True
HappyCam	Canada	500	False

MULTI-ATTRIBUTE KEY

Key = fName,lName
(what does this mean?)

fName	lName	Income	Department
Alice	Smith	20000	Testing
Alice	Thompson	50000	Testing
Bob	Thompson	30000	SW
Carol	Smith	50000	Testing

MULTIPLE KEYS



SSN	fName	lName	Income	Department
111-22-3333	Alice	Smith	20000	Testing
222-33-4444	Alice	Thompson	50000	Testing
333-44-5555	Bob	Thompson	30000	SW
444-55-6666	Carol	Smith	50000	Testing

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)
```

Foreign key to Country.name

Company

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

Country

name	population
USA	320M
Japan	127M

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

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

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?*

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*

THEORY BREAK

- 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

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

JOINS IN SQL

pname	price	category	manufacturer
MultiTouch	199.99	gadget	Canon
SingleTouch	49.99	photography	Canon
Gizom	50	gadget	GizmoWorks
SuperGizmo	250.00	gadget	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

Retrieve all Japanese products that cost < $150

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

JOINS IN SQL

pname	price	category	manufacturer
MultiTouch	199.99	gadget	Canon
SingleTouch	49.99	photography	Canon
Gizom	50	gadget	GizmoWorks
SuperGizmo	250.00	gadget	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

Retrieve all Japanese products that cost < $150

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

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

JOINS IN SQL

pname	price	category	manufacturer
MultiTouch	199.99	gadget	Canon
SingleTouch	49.99	photography	Canon
Gizom	50	gadget	GizmoWorks
SuperGizmo	250.00	gadget	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

Retrieve all Japanese products that cost < $150

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

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

JOINS IN SQL

pname	price	category	manufacturer
MultiTouch	199.99	gadget	Canon
SingleTouch	49.99	photography	Canon
Gizom	50	gadget	GizmoWorks
SuperGizmo	250.00	gadget	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

Retrieve all USA companies
that manufacture “gadget” products

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

JOINS IN SQL

pname	price	category	manufacturer
MultiTouch	199.99	gadget	Canon
SingleTouch	49.99	photography	Canon
Gizom	50	gadget	GizmoWorks
SuperGizmo	250.00	gadget	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

Retrieve all USA companies
that manufacture "gadget" products

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

Why
DISTINCT?

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

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

INNER JOIN

Employee

id	name
1	Joe
2	Jack
3	Jill

Sales

employeeID	productID
1	344
1	355
2	544

Retrieve employees and their sales

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

INNER JOIN

Employee

id	name
1	Joe
2	Jack
3	Jill

Sales

employeeID	productID
1	344
1	355
2	544

Retrieve employees and their sales

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

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

INNER JOIN

Employee

id	name
1	Joe
2	Jack
3	Jill

Sales

employeeID	productID
1	344
1	355
2	544

Retrieve employees and their sales

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

id	name	empolyeeID	productID
1	Joe	1	344
1	Joe	1	355
2	Jack	2	544

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

INNER JOIN

Employee

id	name
1	Joe
2	Jack
3	Jill

Sales

employeeID	productID
1	344
1	355
2	544

Retrieve employees and their sales

Jill is missing

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

id	name	empolyeeID	productID
1	Joe	1	344
1	Joe	1	355
2	Jack	2	544

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

INNER JOIN

Employee

id	name
1	Joe
2	Jack
3	Jill

Sales

employeeID	productID
1	344
1	355
2	544

Retrieve employees and their sales

Jill is missing

Alternative syntax

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

id	name	empolyeeID	productID
1	Joe	1	344
1	Joe	1	355
2	Jack	2	544

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

OUTER JOIN

Employee

id	name
1	Joe
2	Jack
3	Jill

Sales

employeeID	productID
1	344
1	355
2	544

Retrieve employees and their sales

```
SELECT *
FROM   Employee E
       LEFT OUTER JOIN
       Sales S
    ON E.id = S.employeeID
```

Jill is present

id	name	empolyeeID	productID
1	Joe	1	344
1	Joe	1	355
2	Jack	2	544
3	Jill	NULL	NULL

(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
```

Product

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
Camera	Photo	Hitachi
OneClick	Photo	Hitachi

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

(INNER) JOINS

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

Product

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
Camera	Photo	Hitachi
OneClick	Photo	Hitachi

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

(INNER) JOINS

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

Product

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
Camera	Photo	Hitachi
OneClick	Photo	Hitachi

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

(INNER) JOINS

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

Product

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
Camera	Photo	Hitachi
OneClick	Photo	Hitachi

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

pname	category	manufacturer	cname	country
Gizmo	gadget	GizmoWorks	GizmoWorks	USA

(INNER) JOINS

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

Product

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
Camera	Photo	Hitachi
OneClick	Photo	Hitachi

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

(INNER) JOINS

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

Product

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
Camera	Photo	Hitachi
OneClick	Photo	Hitachi

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

(INNER) JOINS

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

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