CSE 344

JUNE 20TH

RELATIONAL DATABASES AND SQLITE

ADMINISTRATIVE MINUTIAE

- **Online Quizzes**
  - newgradiance.com
  - Course token: **6F084FB3**
- **Discussion board (Piazza)**
  - Link on web site
- **HW1**
  - We will create your gitlab repo today
  - Will have HW1 in it (with instructions)
- **Section**
  - Largely help with setup, but some practice with basic SQLite

REVIEW

What is a database?

- A collection of files storing related data

What is a DBMS?

- An application program that allows us to manage efficiently the collection of data files

EXAMPLE: YOUR NEW APP

What app should we build?

- disease finder app

What data do we need to store?

- diseases: name, category
  - list of symptoms, list of treatments
- users: age, phone
  - list of date, pulse, blood pressure, etc.
- searches: date, list of words
- sessions: list of actions and times
  - (not patient specific, can anonymize)

EXAMPLE: YOUR NEW APP

What operations do we need?

- search for disease matching symptoms
  - search through disease database
  - add search to search history
- list recent searches
- add new user
- user update:
  - change patient phone
  - add new pulse reading
- ...

What constraints can we put on the data?

- phone number must have 10 digits
- pulse must be >= 0
- ...

MORALS

Almost any application has lots of important data

Getting the data right is often half the battle

- what operations do you want to support?
- what data do you need for that?
- what constraints does the data have?

DBMSs

- make app development easier
- make apps more reliable
- make apps more efficient
- make apps more easily changeable

DATA MODELS

Recall our example: want to design a database of diseases:

- name, symptoms, tests, treatments, etc.

How should we describe this data precisely?

Data model = mathematical formalism (or conceptual way) for describing the data

DATA MODELS

Relational

- Data represented as relations

Semi-structured (Json/XML)

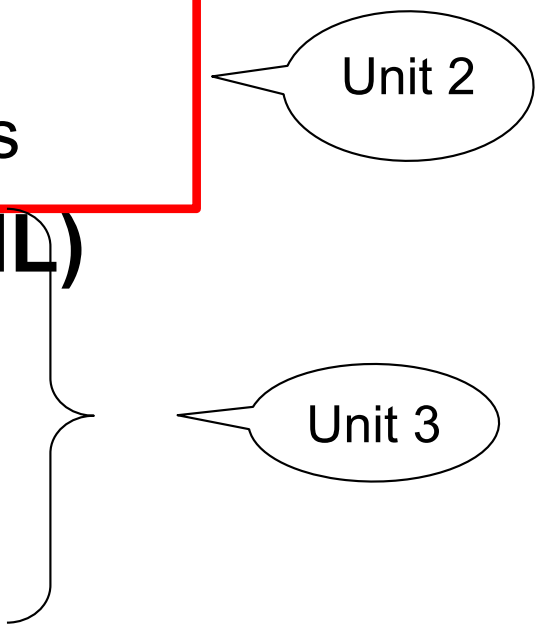
- Data represented as trees

Key-value pairs

- Used by NoSQL systems

Graph

Object-oriented



DATABASES VS. DATA STRUCTURES

- **What are some important distinctions between database systems, and data structure systems?**

DATABASES VS. DATA STRUCTURES

- **What are some important distinctions between database systems, and data structure systems?**
  - *Structure*: Java – concerned with “physical structure”. DBMS – concerned with “conceptual structure”

DATABASES VS. DATA STRUCTURES

- **What are some important distinctions between database systems, and data structure systems?**
  - *Structure*: Java – concerned with “physical structure”. DBMS – concerned with “conceptual structure”
  - *Operations:* Java – low level, DBMS – restricts allowable operations. *Efficiency and data control*

DATABASES VS. DATA STRUCTURES

- **What are some important distinctions between database systems, and data structure systems?**
  - *Structure*: Java – concerned with "physical structure". DBMS – concerned with "conceptual structure"
  - *Operations:* Java – low level, DBMS – restricts allowable operations. *Efficiency and data control*
  - *Data constraints:* Enforced typing allows us to maximize our memory usage and to be confident our operations are successful

3 ELEMENTS OF DATA MODELS

Instance

- The actual data

Schema

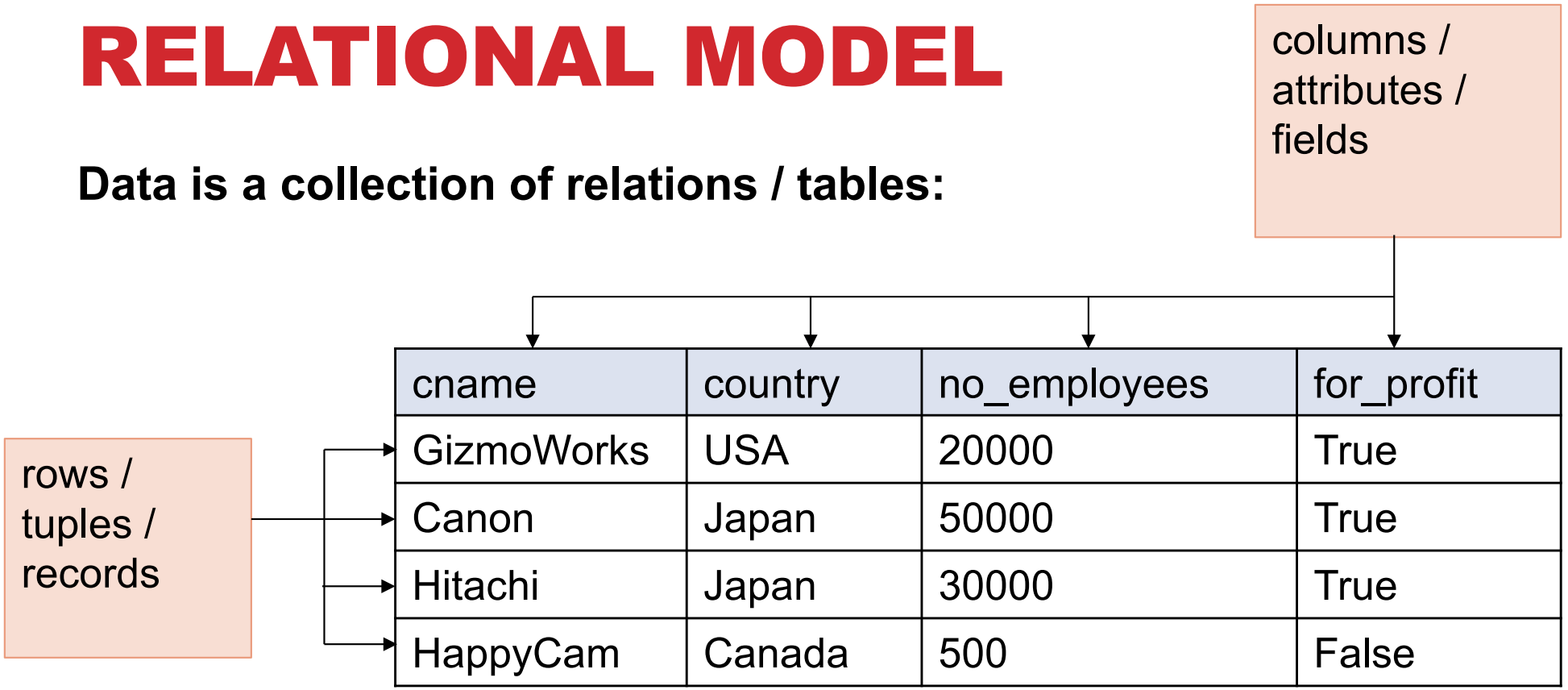
- Describe what data is being stored

Query language

- How to retrieve and manipulate data

RELATIONAL MODEL

Data is a collection of relations / tables:



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)

THE RELATIONAL DATA MODEL

- **Degree (arity) of a relation = #attributes**
- **Each attribute has a type.**
  - Examples types:
    - Strings: CHAR(20), VARCHAR(50), TEXT
    - Numbers: INT, SMALLINT, FLOAT
    - MONEY, DATETIME, …
    - Few more that are vendor specific
  - Statically and strictly enforced
- **Independent of the implementation of the tables**

TABLE IMPLEMENTATION

How would you implement this?

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

TABLE IMPLEMENTATION

How would you implement this?

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

Row major: as an array of objects

GizmoWorks USA 20000 True	Canon Japan 50000 True	Hitachi Japan 30000 True	HappyCam Canada 500 False

TABLE IMPLEMENTATION

How would you implement this?

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

Column major: as one array per attribute

GizmoWorks	Canon	Hitachi	HappyCam
USA	Japan	Japan	Canada
20000	50000	30000	500
True	True	True	False

TABLE IMPLEMENTATION

How would you implement this?

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

Physical data independence
The logical definition of the data remains unchanged, even when we make changes to the actual implementation

KEYS

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

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

KEYS

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

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

KEYS

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

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

KEYS

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

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

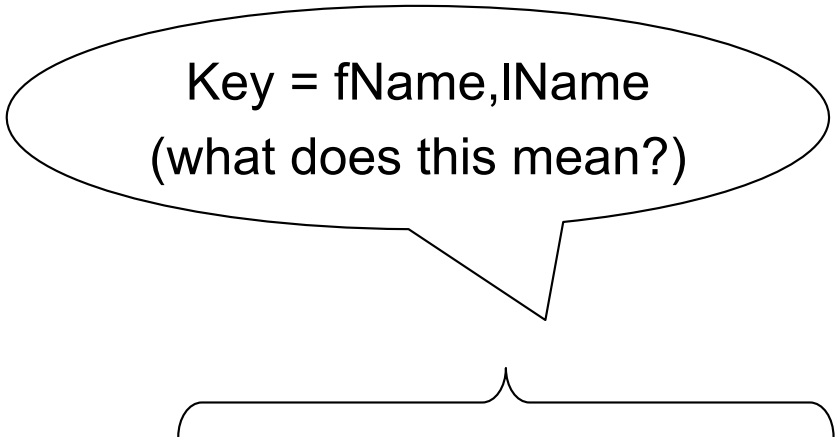
KEYS

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

No: future updates to the database may create duplicate no_employees

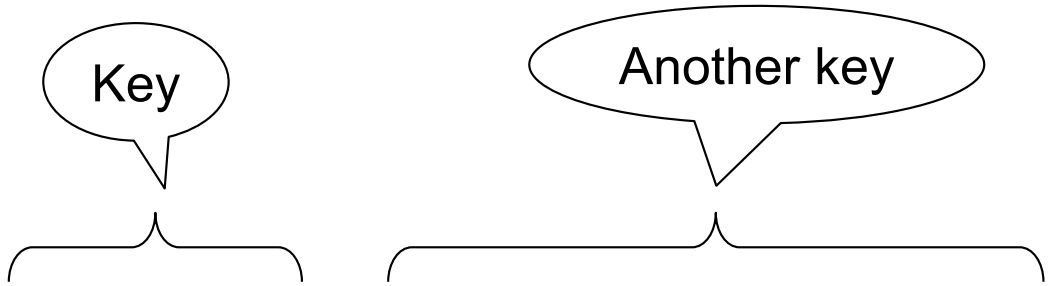
cname	country	no_employees	for_profit
GizmoWorks	USA	20000	True
Canon	Japan	50000	True
Hitachi	Japan	30000	True
HappyCam	Canada	500	False

MULTI-ATTRIBUTE KEY



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)
```

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

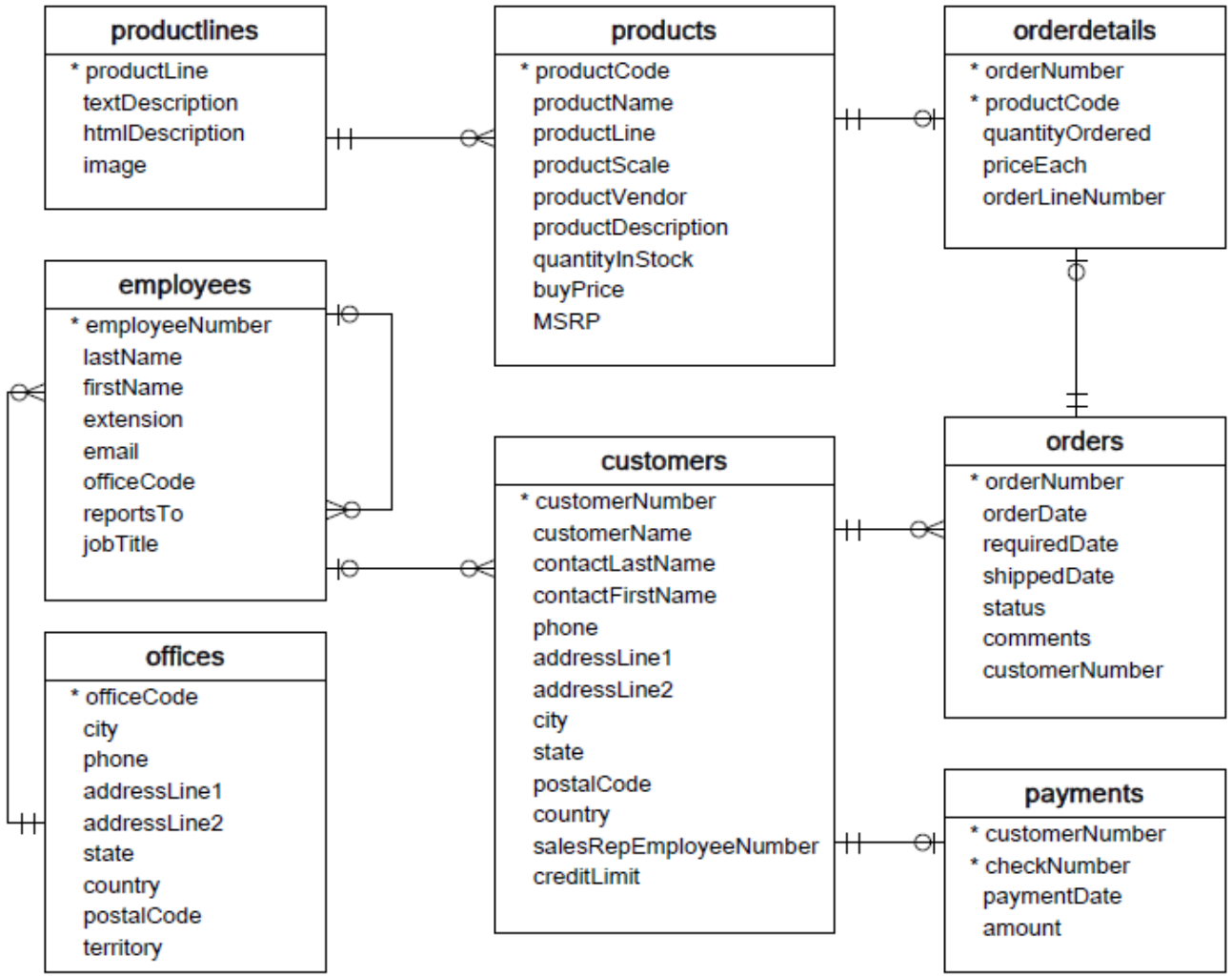
- Usually we underline
- A relation can have many keys, but only one can be chosen as *primary key*

Foreign key:

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

(These are our first examples of *constraints*)

KEYS: EXAMPLE



RELATIONAL DATABASES

- **Why relations?**

RELATIONAL DATABASES

- **Why relations?**
  - Preserves data – if two objects refer to the same common object, that objects data are consistent
  - Saves space – no need to repeat relevant data if it can be relinked later

FIRST NORMAL FORM

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

All relations must be flat: we say that the relation is in *first normal form*

FIRST NORMAL FORM

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

All relations must be flat: we say that the relation is in *first normal form*

E.g. we want to add products manufactured by each company:

FIRST NORMAL FORM

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

All relations must be flat: we say that the relation is in *first normal form*

E.g. we want to add products manufactured by each company:

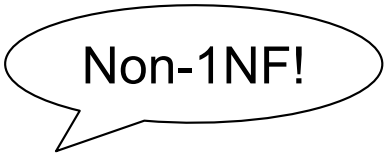
cname	country	no_employees	for_profit	products
Canon	Japan	50000	Y	pname / price / category SingleTouch / 149.99 / Photography Gadget / 200 / Toy
Hitachi	Japan	30000	Y	pname / price / category AC / 300 / Appliance

FIRST NORMAL FORM

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

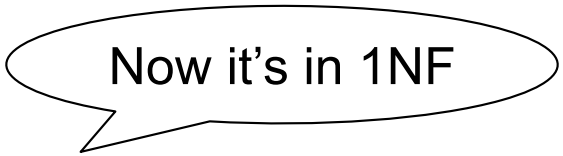
All relations must be flat: we say that the relation is in *first normal form*

E.g. we want to add products manufactured by each company:



cname	country	no_employees	for_profit	products
Canon	Japan	50000	Y	pname / price / category SingleTouch / 149.99 / Photography Gadget / 200 / Toy
Hitachi	Japan	30000	Y	pname / price / category AC / 300 / Appliance

FIRST NORMAL FORM



Company

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

Products

pname	price	category	manufacturer
SingleTouch	149.99	Photography	Canon
AC	300	Appliance	Hitachi
Gadget	200	Toy	Canon

DATA MODELS: SUMMARY

Schema + Instance + Query language

Relational model:

- Database = collection of tables
- Each table is flat: “first normal form”
- Key: may consists of multiple attributes
- Foreign key: “semantic pointer”
- Physical data independence

DEMO 1

- **What operations should we expect SQLite (or any DBMS) to support just on what we know right now?**

DEMO 1

- **What operations should we expect SQLite (or any DBMS) to support just on what we know right now?**
  - create table
  - insert into
  - show rows (“select”)
  - delete from
- **What sorts of inputs do these functions need to have?**

DEMO 1

- **What operations should we expect SQLite (or any DBMS) to support just on what we know right now?**
  - create table
  - insert into
  - select
  - delete from
- **What sorts of inputs do these functions need to have?**
  - create table: table name, schema
  - insert into: table name, tuple
  - select: table name, attributes
  - delete from: table name, condition

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 1