CSE 344

JANUARY 8TH – SQLITE AND JOINS

ADMINISTRATIVE MINUTIAE

- **Next Monday, MLK day**
  - HW1, and QZ1 due next Wednesday
- **Online Quizzes**
  - Newgradiance.com
  - Course token: **B5B103B6**
- **Code assignment**
  - Through gitlab
- **Piazza**
  - Make sure you’re enrolled, announcements coming soon

ADMINISTRATIVE MINUTIAE

- **Office hours**
    - Jayanth: Mon 11-12
    - Colin: Wed 2-3
    - Allison: Mon 1-2
    - Cindy: Tue 2-3
    - James: Tue 10-11
    - Jonathan: Tue 4-5
    - Joshua : Tue 1-2

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:

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)

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

Key

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

Key

Not a key

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

Key

Not a key

Is this a key?

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

Key

Not a key

Is this a key?

No: future updates to the database may create duplicate no_employees

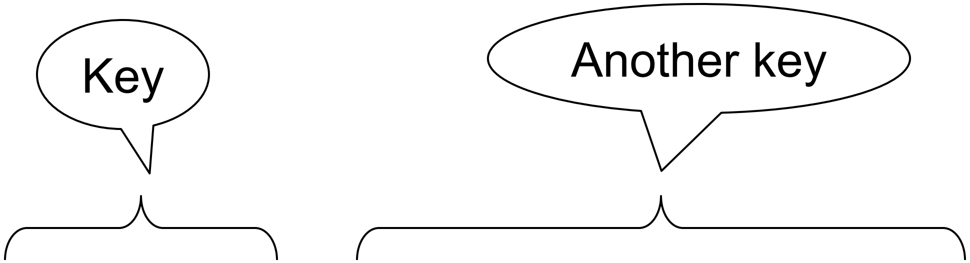
cname	country	no_employees	for_profit
GizmoWorks	USA	20000	True
Canon	Japan	50000	True
Hitachi	Japan	30000	True
HappyCam	Canada	500	False

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



Key

Another key

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

Foreign key to Country.name

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

Country

name	population
USA	320M
Japan	127M

KEYS: SUMMARY

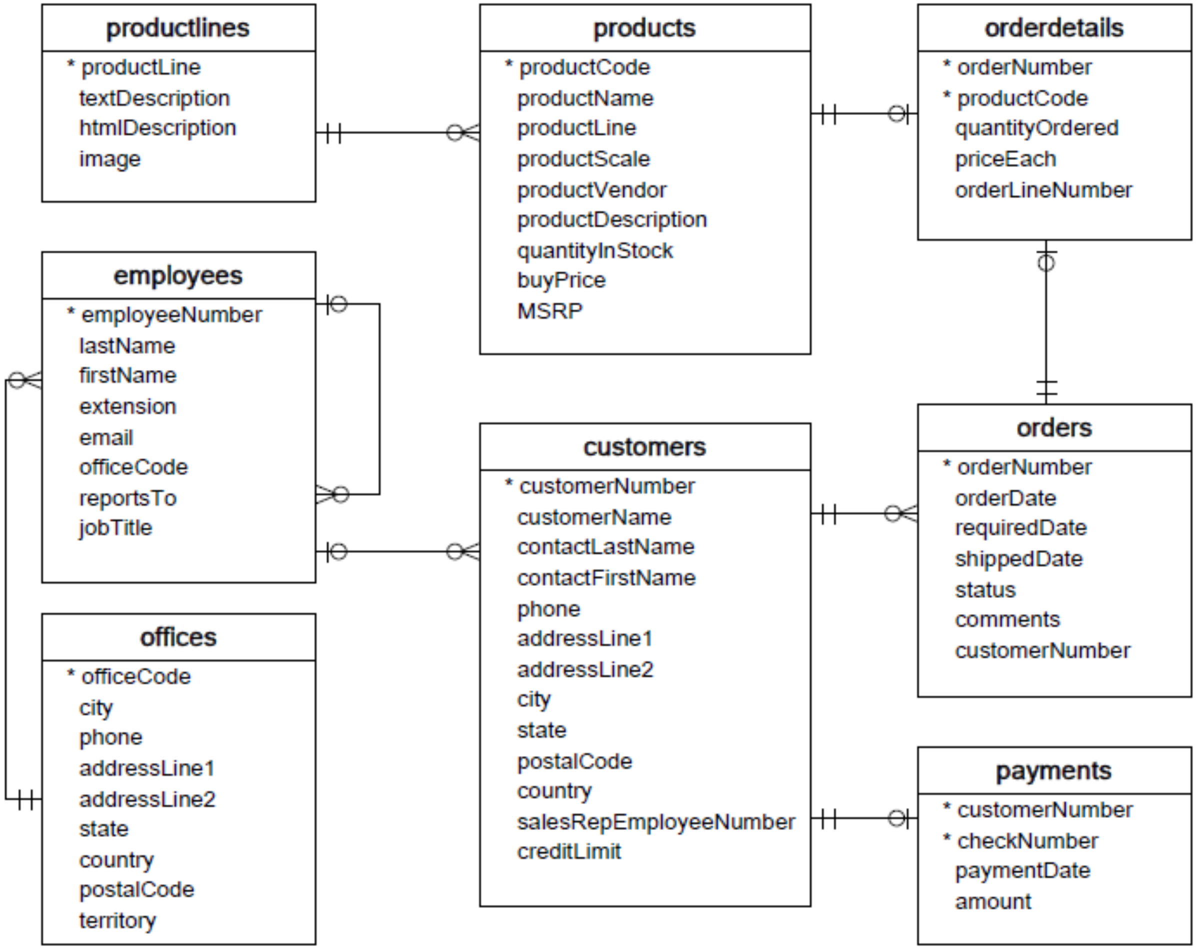
Key = columns that uniquely identify tuple

- 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*

KEYS: EXAMPLE



productlines
* productLine
textDescription
htmlDescription
image

products
* productCode
productName
productLine
productScale
productVendor
productDescription
quantityInStock
buyPrice
MSRP

orderdetails
* orderNumber
* productCode
quantityOrdered
priceEach
orderLineNumber

employees
* employeeNumber
lastName
firstName
extension
email
officeCode
reportsTo
jobTitle

customers
* customerNumber
customerName
contactLastName
contactFirstName
phone
addressLine1
addressLine2
city
state
postalCode
country
salesRepEmployeeNumber
creditLimit

orders
* orderNumber
orderDate
requiredDate
shippedDate
status
comments
customerNumber

offices
* officeCode
city
phone
addressLine1
addressLine2
state
country
postalCode
territory

payments
* customerNumber
* checkNumber
paymentDate
amount

RELATIONAL DATABASES

- Why?

RELATIONAL DATABASES

- **Why?**
  - 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:

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

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:

Non-1NF!

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

FIRST NORMAL FORM

Now it’s in 1NF

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?**
  - 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

- **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 other behavior do we expect from these functions?**
  - Much of the behavior is similar to a dictionary from 332.
  - Create table ~= new DS(), insert into ~= insert(k,v), select !~= find(k), delete from ~= remove(k)
  - *Also have the key constraints!*

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

DISCUSSION

- **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]

DISCUSSION

- **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]

DEMO 2