CSE 344

JANUARY 5TH – INTRO TO THE RELATIONAL DATABASE
ADMINISTRATIVE MINUTIAE

- Midterm Exam: February 9th: 3:30-4:20
- Final Exam: March 15th: 2:30 – 4:20
ADMINISTRATIVE MINUTIAE

• Midterm Exam: February 9th: 3:30-4:20
• Final Exam: March 15th: 2:30 – 4:20
• HW#1 “Out” on Monday
• Online Quiz #1 out on Monday
• Syllabus and course website
• Expect email w/link to Piazza over the weekend
ADMINISTRATIVE MINUTIAE

- Midterm Exam: February 9th: 3:30-4:20
- Final Exam: March 15th: 2:30 – 4:20
- HW#1 “Out” on Monday
- Online Quiz #1 out on Monday
- Syllabus and course website
- Expect email w/link to Piazza over the weekend
- Next week: section will be very helpful – setting up git and SQLite. Don’t hesitate to come to OH if you’re having trouble – tutorial w/ lecture slides
CLASS OVERVIEW

Unit 1: Intro

Unit 2: Relational Data Models and Query Languages
  • Data models, SQL RA, Datalog

Unit 3: Non-relational data

Unit 4: RDMBS internals and query optimization

Unit 5: Parallel query processing

Unit 6: DBMS usability, conceptual design

Unit 7: Transactions

Unit 8: Advanced topics (time permitting)
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
DATA MODELS

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

- author, title, publisher, pub date, price, etc
- How should we describe this data?

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

Structure:
• 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”
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:
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
Data is a collection of relations / tables:

<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>

mathematically, relation is a set of tuples

- each tuple (or entry) must have a value for each attribute
- order of the rows is unspecified
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?
Data is a collection of relations / tables:

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

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
KEYS

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

Key

Not a key
KEYS

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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cname</td>
<td>country</td>
<td>no_employees</td>
<td>for_profit</td>
<td></td>
</tr>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
<td>20000</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>HappyCam</td>
<td>Canada</td>
<td>500</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>
**KEYS**

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

---

**Is this a key?**

No: future updates to the database may create duplicate **no_employees**

<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>
**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>
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>Company</th>
<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>
<td></td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>population</td>
</tr>
<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
QUERY LANGUAGE

SQL

- **Structured Query Language**
- Developed by IBM in the 70s
- Most widely used language to query relational data

Other relational query languages

- Datalog, relational algebra
OUR FIRST DBMS

SQL Lite

Will switch to SQL Server later in the quarter
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
- select
- delete from
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?
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:
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
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:
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
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:
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
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?
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 $\sim=$ new DS(), insert into $\sim=$ insert(k,v), select ! $\sim=$ find(k), delete from $\sim=$ remove(k)
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 * FROM [tablename]
DEMO 1

• Common Syntax
  • CREATE TABLE [tablename] 
    ([att1] [type1],
    [att2] [type2]…);
  • INSERT INTO [tablename] VALUES ([val1],[val2]…);
  • SELECT [att1],[att2],… FROM [tablename]
Common Syntax

- CREATE TABLE [tablename]
  ([att1] [type1],
   [att2] [type2]...);
- INSERT INTO [tablename] VALUES ([val1],[val2]...);
- SELECT [att1],[att2],… FROM [tablename]
  WHERE [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]
• 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
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
How would you implement this?

<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>
How would you implement this?

<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>

Row major: as an array of objects

<table>
<thead>
<tr>
<th>GizmoWorks</th>
<th>Canon</th>
<th>Hitachi</th>
<th>HappyCam</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>Japan</td>
<td>Japan</td>
<td>Canada</td>
</tr>
<tr>
<td>20000</td>
<td>50000</td>
<td>30000</td>
<td>500</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
How would you implement this?

<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>

Column major: as one array per attribute

<table>
<thead>
<tr>
<th>GizmoWorks</th>
<th>Canon</th>
<th>Hitachi</th>
<th>HappyCam</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>Japan</td>
<td>Japan</td>
<td>Canada</td>
</tr>
<tr>
<td>20000</td>
<td>50000</td>
<td>30000</td>
<td>500</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
How would you implement this?

<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>

Physical data independence
The logical definition of the data remains unchanged, even when we make changes to the actual implementation.
### FIRST NORMAL FORM

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

<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>
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>
<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>
**FIRST NORMAL FORM**

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>
<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>

| products | |
|----------|-----|----------------|
| pname    | price | category       |
|.SingleTouch | 149.99 | Photography    |
| Gadget   | 200   | Toy            |

| products | |
|----------|-----|----------------|
| pname    | price | category       |
| AC       | 300   | Appliance      |
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          |         
Hitachi| Japan    | 30000        | Y          |         
```

```
pname | price  | category
-------|--------|----------
SingleTouch | 149.99 | Photography
Gadget       | 200    | Toy
AC           | 300    | Appliance
```
### Company

<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>

### Products

<table>
<thead>
<tr>
<th>pname</th>
<th>price</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleTouch</td>
<td>149.99</td>
<td>Photography</td>
<td>Canon</td>
</tr>
<tr>
<td>AC</td>
<td>300</td>
<td>Appliance</td>
<td>Hitachi</td>
</tr>
<tr>
<td>Gadget</td>
<td>200</td>
<td>Toy</td>
<td>Canon</td>
</tr>
</tbody>
</table>
DEMO 3
DATA MODELS: SUMMARY

Schema + Instance + Query language

Relational model:

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