
## Announcements

Project 2d must be time stamped Friday (2:00AM) for Friday turn-in

- Please keep up with the reading


## Databases

*Databases are collections of information; our study repeats a theme: Tell the computer the structure, and it can help you!*


## Why Study Databases?

Some of us want to compute, but all of us want information …

- Much of the archived information is in tables
- Databases enhance applications, e.g. Web
- Once you know how to create databases, you can use them to personal advantage
- Databases introduce interesting ideas

How much of your information can be in a table?


## Stone Age Databases

Before relational databases (the kind we study) there were only "flat files"

- Structural information was difficult to express
- All processing of information was "special cased" -- custom programs were needed
- Information repeated; difficult to combine
- Changes in format of one file means all programs that ever process that file must be changed … adding ZIP codes

E.F. Codd of IBM invented relational databases


## Relational Databases

Information is stored in tables

- Tables store information about *entities* -- things or stuff … keep entities of one kind
- Entities have characteristics called *attributes*
- Tables are *tuples* (rows or records) of attributes (columns or fields )
- Every row must be unique, identified by a key
- Relationships -- associations among the data values are stored

Table structure = schema
Table contents = instance


## A Table

Tables have names, attributes, tuples

Example : Table

| ID | Last | First | Hire | Addr |
|---|---|---|---|---|
| 1 | Davolino | Nancy | 01 May 1992 | 507 20th Ave E |
| 2 | Fuller | Andrew | 14 Aug 1992 | 908 W. Capital Way |
| 3 | Wooster | Berton | 01 Apr 1993 | 722 Moss Bay Blvd |
| 4 | Peacock | Margaret | 03 May 1993 | 4110 Old Redmond Rd |
| 5 | Buchanan | Steven | 17 Oct 1994 | 13 Garrett Hill |
| 6 | Sullimani | Okan | | |

Example:

| | | |
|---|---|---|
| ID | number | unique number(Key) |
| Last | text | person's last name |
| First | text | person's first name |
| Hire | date | first day on joy |
| Addr | text | street address |

Instance
Schema


## Redundancy Is Very Bad

Not every assembly of tables is a good database -- repeating data is bad

- Replicated data can differ in its different locations, e.g. multiple addresses can differ
- *Inconsistent data* is worse than no data
- Keep a single copy of any data, and if it is needed in multiple places, associate it with a key, and store key rather than the data


## "You can look it up"

When looking for information, a single item might be the answer, but a table is more likely

- "Who is taking FIT100"? Table of students
- "Whose mile run time ≤ 4:00?" Runner table
- "Who won 2003 Grammy for 'Best New Artist?" A table containing only a single row
- "Who is president of UW?" Empty Table

Queries to a DB (set of tables) produces tables


## Tables From Tables

There are five fundamental operations on tables to create tables:

- Select -- pick rows from a table
- Project -- pick columns from a table
- Union -- combine two tables w/like columns
- Difference -- remove one table from another
- Product -- create "all pairs" from two tables

Though not primitive "Join" is usually included


## Select Operation

Select creates a table from the rows of another table meeting a criterion

**Select_from** Example **On** Hire < 1993

Example : Table

| ID | Last | First | Hire | Addr |
|---|---|---|---|---|
| 1 | Davolino | Nancy | 01 May 1992 | 507 20th Ave E |
| 2 | Fuller | Andrew | 14 Aug 1992 | 908 W. Capital Way |
| 3 | Wooster | Berton | 01 Apr 1993 | 722 Moss Bay Blvd |
| 4 | Peacock | Margaret | 03 May 1993 | 4110 Old Redmond Rd |
| 5 | Buchanan | Steven | 17 Oct 1994 | 13 Garrett Hill |
| 6 | Sullimani | Okan | 12 Dec 1994 | Coventry House |

Example : Table

| ID | Last | First | Hire | Addr |
|---|---|---|---|---|
| 1 | Davolino | Nancy | 01 May 1992 | 507 20th Ave E |
| 2 | Fuller | Andrew | 14 Aug 1992 | 908 W. Capital Way |


## Project

Project creates a table from the columns of another table

**Project** Last, First **From** Example

Example : Table

| ID | Last | First | Hire | Addr |
|---|---|---|---|---|
| 1 | Davolino | Nancy | 01 May 1992 | 507 20th Ave E |
| 2 | Fuller | Andrew | 14 Aug 1992 | 908 W. Capital Way |
| 3 | Wooster | Berton | 01 | |
| 4 | Peacock | Margaret | 03 | |
| 5 | Buchanan | Steven | 17 | |
| 6 | Sullimani | Okan | 12 | |

Example : Table

| Last | First |
|---|---|
| Davolino | Nancy |
| Fuller | Andrew |
| Wooster | Berton |
| Peacock | Margaret |
| Buchanan | Steven |
| Sullimani | Okan |


## Union

Union (written like addition) combines two tables with same attributes

- PoliticalUnits = States **+** Provinces

States : Table

| Name | Capitol | Sight |
|---|---|---|
| Washington | Olympia | Mt. Rainier |
| Oregon | Salem | Crater Lake |
| California | Sacramento | Golden G |

Provinces : Table

| Name | Capitol | S |
|---|---|---|
| British Columbia | Victoria | Stanle |
| Alberta | Edmonton | Banff |

PoliticalUnits : Table

| Name | Capitol | Sight |
|---|---|---|
| British Columbia | Victoria | Stanley Park |
| Alberta | Edmonton | Banff |
| Washington | Olympia | Mt. Rainier |
| Oregon | Salem | Crater Lake |
| California | Sacramento | Golden Gate |

## Difference

Difference (written like subtraction) removes 1 table's rows from another

- Eastern = States - WestCoast

States : Table

| Name | Capitol | Sight |
|---|---|---|
| Washington | Olympia | Mt. Rainier |
| Oregon | Salem | Crater Lake |
| California | Sacramento | Golden Gate |
| Arizona | Phoenix | Grand Canyon |
| Nevada | Carson City | Las Vegas |

WestCoast : Table

| Name | Capitol | Sight |
|---|---|---|
| Washington | Olympia | Mt. Rainier |
| Oregon | Salem | Crater Lake |
| California | Sacramento | Golden Gate |

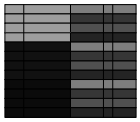
Eastern : Table

| Name | Capitol | Sight |
|---|---|---|
| Arizona | Phoenix | Grand Canyon |
| Nevada | Carson City | Las Vegas |

## Product

Product (written like multiplication) combines columns and pairs all rows

Colors = Blues **x** Reds



**Column Rule:** If A has *x* columns, B has *y* columns, A **x** B has *x+y* columns

**Row Rule:** If A has *m* rows, B has *n* rows A **x** B has *mn* rows

There's divide, too, but forget it

## Join

Join (written like a bow tie) combines rows (like **x**) if common field matches

Homes = States ⋈ Students

States : Table

| State | Capitol | Sight |
|---|---|---|
| Washington | Olympia | Mt. Rainier |
| Oregon | Salem | Crater Lake |
| California | Sacramento | Golden Gate |
| Arizona | Phoenix | Grand Canyon |
| Nevada | Carson City | Las Vegas |

Students : Table

| First | Last | State |
|---|---|---|
| John | Jones | Washington |
| Jennifer | Smith | California |
| Brian | Tims | Manitoba |

Homes : Table

| State | Capitol | Sight | First | Last |
|---|---|---|---|---|
| Washington | Olympia | Mt. Rainier | John | Jones |
| California | Sacramento | Golden Gate | Jennifer | Smith |

## DB Operations

The five DB Operations can create any table from a given set of tables

- All modern database systems are built on these relational operations
- Join is not primitive, but can be built from 5
- Join, select and project are used most often
- The operations are not usually used directly, but are used indirectly from other languages

SQL, the DB language we learn, is built on basic 5