

---

# Database Intro

INFO/CSE 100, Autumn 2004  
Fluency in Information Technology

<http://www.cs.washington.edu/100>

---

# Readings and References

- Reading
  - » *Fluency with Information Technology*
    - Chapter 13, Introduction to Database Concepts
- References
  - » *Access Database: Design and Programming*
    - by Steve Roman, published by O'Reilly

---

# 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



**The Internet Movie Database**


Visited by over 20 million movie lovers each month!  
Welcome to the Internet Movie Database, the biggest, best, most award-winning movie site on the planet.

---


# How to organize the data?

- Before relational databases (the kind we study) there were only “flat files”
  - » Structural information is difficult to express
  - » All processing of information is “special cased”
    - custom programs are needed
  - » Information repeated; difficult to combine
  - » Changes in format of one file means all programs that ever process that file must be changed
    - eg, adding ZIP codes

## tab-delimited file example



**UW-FHCRC**  
Variation Discovery Resource



FRED HUTCHINSON  
CANCER RESEARCH CENTER

**Download of Variation Data (Single File)**

[Global Switchboard Files](#)

This is a tab delimited text file in our "prettybase" format, which describes all SNP sites discovered by the SeattleSNPs PGS. The format of this file is:

Line format:  
<chromosome position><chromosome><HUGO\_NAME> <PGS sample ID> <allele1> <allele2>

Example: 74772592-10-PLAU D001 G T

The "chromosome position" is generated from mapping to the most recent genome assembly available from the [UCSC Genome Assembly](#).

1100322-IL3RA-X	D001	N	N
1100322-IL3RA-X	D002	G	G
1100322-IL3RA-X	D003	G	G
1100322-IL3RA-X	D004	G	G
1100322-IL3RA-X	D005	G	G
1100322-IL3RA-X	D006	G	G
1100322-IL3RA-X	D007	G	G
1100322-IL3RA-X	D008	G	G
1100322-IL3RA-X	D009	A	G
1100322-IL3RA-X	D010	N	N
1100322-IL3RA-X	D011	N	N
1100322-IL3RA-X	D012	N	N
1100322-IL3RA-X	D013	G	G
1100322-IL3RA-X	D014	A	G
1100322-IL3RA-X	D015	N	N
1100322-IL3RA-X	D016	N	N
1100322-IL3RA-X	D033	A	G
1100322-IL3RA-X	D034	A	G
1100322-IL3RA-X	D035	G	G
1100322-IL3RA-X	D036	A	G
1100322-IL3RA-X	D037	A	A
1100322-IL3RA-X	D038	G	G
1100322-IL3RA-X	D039	G	G
1100322-IL3RA-X	D040	G	G
...			

## Unix termcap example

```
# FILE FORMAT:
#
# The version you are looking at may be in any of three formats: master
# (terminfo with OT capabilities), stock terminfo, or termcap. You can tell
# which by the format given in the header above.
#
# The master format is accepted and generated by the terminfo tools in the
# ncurses suite; it differs from stock (System V-compatible) terminfo only
# in that it admits a group of capabilities (prefixed 'OT') equivalent to
# various obsolete termcap capabilities.
...
# ANSI capabilities are broken up into pieces, so that a terminal
# implementing some ANSI subset can use many of them.
ansi+local1:\
        :do=\E[B:le=\E[D:nd=\E[C:up=\E[A:
ansi+local:\
        :DO=\E[%dB:LE=\E[%dD:RI=\E[%dC:UP=\E[%dA:tc=ansi+local1:
ansi+tabs:\
        :bt=\E[Z:ct=\E[2g:st=\EH:ta=^I:
ansi+inittabs:\
        :it#8:tc=ansi+tabs:
```

## Library example

ISBN	Title	ASIN	Author	PubYear	PubID	PubName	PubPhone	Price
1-111-1111-1	Chris	4	Ernest	444-444-4444	1	Big Books	121-456-7890	\$23.95
0-09-999999-9	Ernie	1	Archer	111-111-1111	1	Big Books	121-456-7890	\$28.95
0-09-315678-7	Patric Queen	7	Spencer	777-777-7777	1	Big Books	121-456-7890	\$11.95
0-09-845678-3	Heard	5	Shakespeare	333-333-3333	2	Alpha Press	999-999-9999	\$28.95
0-101-45678-9	Lead	3	Ernest	333-333-3333	1	Big Books	121-456-7890	\$23.95
0-12-345678-4	Long Legs	1	Archer	111-111-1111	2	Small House	71-456-0080	\$48.95
0-09-777777-7	Long Legs	5	Shakespeare	333-333-3333	2	Alpha Press	999-999-9999	\$48.95
0-101-345678-9	Midweek	5	Shakespeare	333-333-3333	2	Alpha Press	999-999-9999	\$13.95
0-11-345678-9	Midway Link	2	Midweek	222-222-2222	1	Small House	71-456-0080	\$48.95
0-12-312345-1	On Liberty	1	Mil	333-333-3333	1	Big Books	121-456-7890	\$23.95
0-201-32123-1	Ballon	13	Starry	222-222-2222	2	Small House	71-456-0080	\$24.95
0-201-32123-1	Ballon	11	Starry	222-222-2222	2	Small House	71-456-0080	\$24.95
0-201-32123-1	Ballon	12	Starry	222-222-2222	2	Small House	71-456-0080	\$24.95
0-35-23456-9	Indian Street	18	Jones	123-123-1232	3	Small House	71-456-0080	\$23.95
0-35-33456-9	Indian Street	9	Smith	123-123-1232	3	Small House	71-456-0080	\$23.95
0-123-45678-9	Ulyness	6	Joyce	666-666-6666	2	Alpha Press	999-999-9999	\$24.95
1-22-33789-8	Visual Basic	4	Ernest	444-444-4444	1	Big Books	121-456-7890	\$23.95

notice the redundancy

from Access Database book, Steve Roman

## Relational Databases

- Information is stored in tables
  - » Tables store information about *entities*
  - » Entities have characteristics called *attributes*
  - » Each row in a table represents a single entity
    - Each row is a set of attribute values
    - 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 in a Database

Tables have names, attributes, rows

ID	Last	First	JobID	Hire	Street	City	State	Country
1	Davline	Nancy	0	5/11/90	507 20th Ave E	Seattle	WA	USA
2	Faller	Andrew	3	8/14/92	908 W. Capital Way	Seattle	WA	USA
3	Woozler	Bertie	1	4/11/90	722 Moss Bay Blvd	Seattle	WA	USA
4	Peacock	Margaret	2	5/31/93	4110 Old Redmond Rd	Kirkland	WA	USA
5	Buchanan	Steven	3	10/17/94	13 Garrett Hill	Seattle	WA	USA
6	Sullivan	Okian	2	12/12/94	Coventry House	Seattle	WA	USA

**Schema for Example table:**

ID	number	unique number(Key)
Last	text	person's last name
First	text	person's first name
JobCode	number	current position
Hire	date	first day on job
...		

instance

schema

# Two tables in a database

JobID	Title	Paycode
0	CEO	0
1	VP	3
2	Engineer	4
3	Administrative	6
4		0

# Redundancy in a database is Very Bad

- Not every assembly of tables is a good database
- Repeating data is a bad idea
  - » 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
    - if it is needed in multiple places, associate it with a key and store key rather than the data



# Relationships between tables

## “You can look it up”

- When looking for information, a single item might be the answer, but a table is more likely
  - » Which employees live in Kirkland?
    - Table of employees
  - » Who is taking INFO/CSE 100?
    - Table of students
  - » Whose mile run time  $\leq 4:00$ ?
    - Table of runners

First	Last	Hire	City
Margaret	Peacock	5/21/93	Kirkland

Query to a database (set of tables) produces a new table

## Relational Algebra: Tables From Tables

- There are five basic “algebraic” operations on 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

From this basis, many more complicated operations can be built up

## Select Operation

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

Select\_from Example On Hire < 1993

ID	Last	First	JobID	Hire	Street	City	State	Country
1	Davolio	Nancy	0	01-May-92	507 20th Ave E	Seattle	WA	USA
2	Fuller	Andrew	3	14-Aug-92	908 W. Capital Way	Seattle	WA	USA
3	Woodster	Bertan	1	01-Apr-93	722 Mass Bay Blvd	Seattle	WA	USA
4	Peacock	Margaret	2	03-May-93	4110 Old Redwood Rd	Kirkland	WA	USA
5	Buckhase	Steven	3	17-Oct-94	13 Garrett Hill	Seattle	WA	USA
6	Sullivan	Olan	2	12-Dec-94	Coventry House	Seattle	WA	USA
7			0					
8			0					
9			0					

ID	Last	First	JobID	Hire	Street	City	State	Country
1	Davolio	Nancy	0	01-May-92	507 20th Ave E	Seattle	WA	USA
2	Fuller	Andrew	3	14-Aug-92	908 W. Capital Way	Seattle	WA	USA

## Project

- Project creates a table from the columns of another table

Project Last, First From Example

Last	First
Davolio	Nancy
Fuller	Andrew
Woodster	Bertan
Peacock	Margaret
Buckhase	Steven
Sullivan	Olan

This is a projection from 9 dimensions to 2 dimensions

## Union

- Union combines two tables with same attributes

All employees = perms UNION temps

ID	Last	First	JobID	Hire	Street	City	State	Country
1	Davolio	Nancy	0	01-May-92	507 20th Ave E	Seattle	WA	USA
2	Fuller	Andrew	3	14-Aug-92	908 W. Capital Way	Seattle	WA	USA
3	Wooster	Boston	1	01-Apr-93	722 Moss Bay Blvd	Seattle	WA	USA
4	Peacock	Margaret	2	03-May-93	4110 Old Redmond Rd	Kirkland	WA	USA
5	Buchanan	Steven	3	17-Oct-94	13 Garrett Hill	Seattle	WA	USA
6	Sullivan	Oliver	2	12-Dec-94	Coventry House	Seattle	WA	USA
0			0					

22-Nov-2004

cse100-20-databases © 2004 University of Washington

17

## Difference

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

- Eastern = States - WestCoast

States : Table			WestCoast : Table		
Name	Capital	Sight	Name	Capital	Sight
Washington	Olympia	Mt. Rainier	Washington	Olympia	Mt. Rainier
Oregon	Salem	Crater Lake	Oregon	Salem	Crater Lake
California	Sacramento	Golden Gate	California	Sacramento	Golden Gate
Arizona	Phoenix	Grand Canyon			
Nevada	Carson City	Las Vegas			

Eastern : Table		
Name	Capitol	Sight
Arizona	Phoenix	Grand Canyon
Nevada	Carson City	Las Vegas

22-Nov-2004

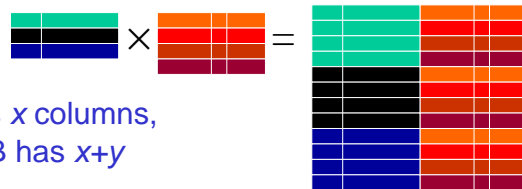
cse100-20-databases © 2004 University of Washington

18

## 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,  $A \times B$  has  $x \times y$  columns

**Row Rule:** If A has  $m$  rows, B has  $n$  rows  $A \times B$  has  $m \times n$  rows

22-Nov-2004

cse100-20-databases © 2004 University of Washington

19

## Join

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

Employee List = Perms  $\bowtie$  JobCodes

ID	Last	First	JobID	Title	Paycode
1	Davolio	Nancy	0	CEO	6
3	Wooster	Boston	1	VP	7
4	Peacock	Margaret	2	Engineer	4
6	Sullivan	Oliver	2	Engineer	4
2	Fuller	Andrew	3	Administrative	6
5	Buchanan	Steven	3	Administrative	6

22-Nov-2004

cse100-20-databases © 2004 University of Washington

19

## 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