**DATABASE & SQL**

Why Not Replication:

Expense, Impracticality, Uniqueness, Ethics

A database is a collection of related data with some inherent meaning

DBMS defining, constructing, manipulating

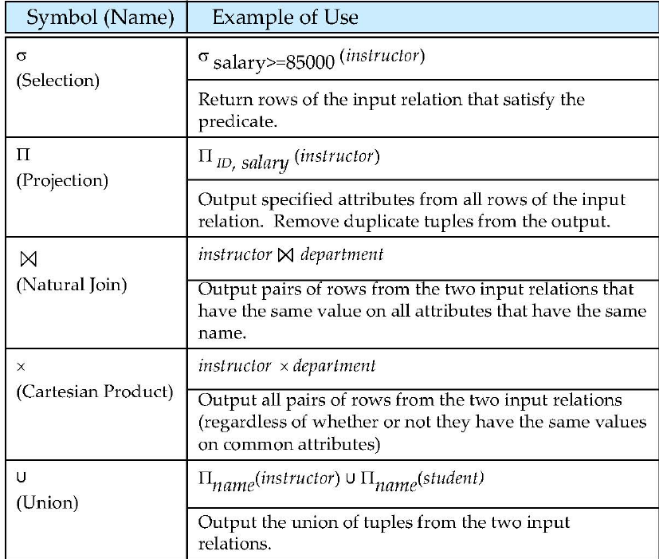
and sharing databases

Advantage: remove data redundancy, Enforce Integrity constraints, restrict unauthorized access, provide multiple user interfaces.

Operational systems, execution, transaction system, OLTP

Analytic systems, evaluation, data warehouse, OLAP

Relational Data: A relation is a table; an attribute is a column; a tuple is a row.



SQL stands for “Structured Query Language”, commands (usually called queries)

Each table has columns (also known as fields) that describe the data, and rows (also known as records) which contain the data.

eliminate the redundant:

*SELECT DISTINCT quant FROM Survey;*

*SELECT taken, person, quant FROM Survey ORDER BY taken ASC, person DESC;*

we can use IN to see if a value is in a specific set: *SELECT \* FROM Survey WHERE person IN ('lake', 'roe');*

same as *SELECT \* FROM Survey WHERE person = 'lake' OR person = 'roe';*

LEFTJOIN a ON b, ab共有加上a独有

String: CHAR(n), VARCHAR2(n), CLOB(n)

Numeric: Number(p,s)

Date: Date, Timestamp

Binary: Raw, BLOB, BFILE

as重命名，括号里先算:

*SELECT taken, round(5 \* (reading - 32) / 9, 2) as Celsius FROM Survey WHERE quant ='sal' AND (person = 'lake' OR person = 'roe');*

Use UNION to combine the results of two or more queries.*SELECT \* FROM Person WHERE id = 'dyer' UNION SELECT \* FROM Person WHERE id = 'roe';*

*SELECT Site.lat, Site.long, Visited.dated, Survey.quant, Survey.reading*

Foreign Key linked to primary key

*FROM Site JOIN Visited JOIN Survey*

*ON Site.name = Visited.site*

*AND Visited.id = Survey.taken*

*AND Visited.dated IS NOT NULL;*

Database Normalization to organize data and reduce redundancy and improve integrity.

* Every value in a database should be atomic: indivisible (1NF)
* Every record should have a unique primary key.
* A database should not contain redundant information. (2NF)
* Units and similar metadata should be stored with the data.
* Ideal for OLTP (3NF for Normal Forms)

% for multiple, \_ for single

*SELECT dept FROM department*

*WHERE building LIKE ‘%Watson\_ast%’;*

SUBQUERY Example: List all departments along with the number of instructors each department

*SELECT dept\_name,*

*(SELECT COUNT(\*) FROM instructor*

*WHERE instructor.dept\_name = department.dept\_name)*

*AS num\_instructors FROM department;*

Example: Find the average instructors’ salaries of those departments where the average salary is greater than $42,000

*SELECT dept\_name, avg\_salary*

*FROM (SELECT dept\_name, AVG(salary) as avg\_salary FROM instructor*

*GROUP BY dept\_name)*

*WHERE avg\_salary > 42000;*

Example: Find courses offered in Fall 2009 and in Spring 2010

*SELECT DISTINCT course\_id FROM section*

*WHERE semester = ’Fall’ AND year= 2009 AND cource\_id IN (SELECT course\_id FROM section WHERE semester = ’Spring’ AND year = 2010);*

Example: Find names of instructors with salary greater than average salary for all instructors in the Biology department.

*SELECT name from instructor*

*where salary > some (select avg(salary)*

*from instructor*

*where dept name = ‘Biology’);*

*%%sql*

*DROP TABLE IF EXISTS rides;*

*CREATE TABLE rides(*

*Var*

*FOREIGN KEY (Bike\_ID) references bikes(Bike\_ID) on delete cascade on update cascade）*

**UNIX SHELL**

graphical user interfaces (GUI): windows, icons and pointers.

shell provides - a simple language and a command-line interface (CLI)

The heart of a command-line interface is a read-evaluate-print loop (REPL).

structure: a **command**, some **flags** (also called **options** or **switches**) and an **argument**.

Arguments tell the command what to operate on (e.g. files and directories).

The shell’s main advantages are its **high action-to-keystroke ratio**, its support for **automating repetitive tasks**, and its capacity to **access networked machines**.

The shell’s main disadvantages are its **primarily textual nature** and how **cryptic** **its commands and operation** can be.

A **relative path** specifies a location starting from the current location.

An **absolute path** specifies a location from the root of the file system.

Two meanings for the / character (**Slash**).

When it appears at the front of a file or directory name, it’s the root directory.

When it appears inside a name, it’s just a separator.

About “..”, For example, if we are in /Users/nelle/data, the command ls .. will give us a listing of /Users/nelle

the character ~ (tilde) at the start of a path to mean “the current user’s home directory”. For example, if Nelle’s home directory is /Users/nelle, then ~/data is equivalent to /Users/nelle/data.

find finds files with specific properties that match patterns.

cd path changes the current working directory.

rm -i : the -i option will prompt before every removal. **To remove safely**

rm -r direc 可以整个删除direc 或者 先rm files 然后 rmdir direc

\* is a **wildcard**. It matches zero or more characters

? only matches a single character

?\* match up for at least on char.

>> appends to a file or creates the file if it doesn't exist.

> overwrites the file if it exists or creates it if it doesn't exist.

unzip -l shows you what will be unpacked before you actually unpack it.

Pwd print working directory

. meaning "the currenty working directory"

.. meaning “the directory containing this one”

ls -F .. listing the files and direcs under the parent directory with / indicators

mkdir make a new directory names XXX

touch create a new empty file

cp old new create a new file and copy old stuff to new file

head/tail -n 9 take a quick peek at the end of the file

grep -n add matching line numbers

grep -w whole word matches

grep -v invert the search, i.e. do not contain the word

grep -E put the pattern in quotes

[grep '^a.\*e$' file] look for those lines starting (^) with a, then 0 or more characters and finally and e at the end of the line ($)

[grep -r "0\.49"] to search word with dot.

wc -l/w/c count lines/words/characters

-i upper and lower case insensitive

ls -l long listing format, showing all names , size and time

ls -h makes the file size "human readable", i.e. 5.3k instead of 5369

ls -R lists the contents of directories recursively, i.e. sub-direc, sub-sub-direc

ls -t list things by time of last change, most changed first.

ls -a show all, show us files and directory names that begin w/ . and ..

ls -r display things in reverse order

tr '[:upper:]' '[:lower:]' trnsfer all letters to lower cases

uniq -c combine all the same characters

sort | uniq remove all duplicated lines

sort -n numeric sort

sort -r sort reversely

cut -d separate each line by comma

cut -f 2 to print the second field (2012-11-05,deer) in each line

echo print the file's name as text for testing

csvcut –n display column name

csvcut –c choose columns