cis20.2
design and implementation of software applications II
spring 2008
session # II.2
database systems

topics:

- relational databases
- SQL
- database servers
- distributed databases

relational databases

- a *relational* database consists of multiple *tables*
- each table is defined as having a number of *fields*
- data is *stored* in a table so that a single entry in a table, called a *record*, provides one data element for each field
- a table can be thought of as a spreadsheet, where the *fields* are *columns* in the spreadsheet, and the *records* are *rows*
- records can have "unique" fields, which are called *keys*
- if a record does not have a value for a particular field, then a *NULL* value is entered
- "relational" databases consist of multiple tables that *relate* to each other by having one column (field) in common

- for example:

people_table

ssn	name
012 34 5678	suz
123 45 6789	jen
234 56 7890	lex

the ssn (social security number) uniquely identifies a single person

phone_table

ssn	phone
012 34 5678	212 555 1234
123 45 6789	212 555 5678
234 56 7890	212 555 9000

one can *join* the phone_table to the people_table in order to look up a person's phone number

SQL

- SQL = Structured Query Language
- basic commands:
  - `INSERT` — used to put data into a table
  - `SELECT` — used to see what is in a table
  - `DELETE` — used to remove data from a table
  - `UPDATE` — used to edit data that is already in a table
  - `COMMIT` — like saving a file...
  - `ROLLBACK` — like revert to previous version, "undo"
  - `GRANT` — used to give users a variety of privileges (read, write, delete...)
  - `REVOKE` — like taking away privileges...
- implementations:
  - Oracle
  - mysql
  - Access

database servers

- key functionalities
  - DBMS = database management system
  - SQL "queries" are sent by clients to the server; clients can be integrated GUIs (graphical user interfaces)
  - queries should be *optimized* for fast access
  - the dbms should use a *locking* mechanism to synchronize access and maintain data integrity
  - *deadlock* = when two transactions are waiting for each other to complete, each locking the other out of a needed resource
  - *security* makes sure that users only get access to what they have privileges to access
  - *backup* and *recovery*
- stored procedures
  - useful for speeding up access
  - makes network traffic more efficient

  - maintains database modularity — keeping code separate from data but easily maintainable
- referential integrity
  - makes sure that when tables refer to data in other tables, the other data is actually there...
  - e-commerce applications typically use a 3-tiered architecture: presentation layer = visual objects server layer = business objects database layer = data(base) objects
  - integrity has to be maintained in two ways: (1) in the database table definitions and (2) by coordinating updates to the tables
- relational middleware
  - SQL API (application programmer interface)
  - database driver (converts API SQL and sends messages to database server)
  - protocol stack (facilitates two-way communication between the client and the database server)
  - server software (access the database directly)
  - server administration software (facilitates adding/editing/deleting user accounts and privileges, backups and restores)

distributed databases

- tables are distributed amongst multiple networked computers
- reasons: (1) separate tables by functionality and frequency of access; (2) legacy systems
- problems:
  - replicated data must be kept replicated (both for reading and writing)
  - security must be maintained
  - updates must be synchronized; common states maintained
  - clocks must be synchronized!
- methods:
  - downloading — client-server data distribution; updates periodically from server to client(s); clients can be out of date, so this scheme is only useful in situations where this isn't a problem; this is easiest to implement and maintain
  - data replication — data is copied to places on the network close to where it is needed
  - horizontal fragmentation — tables are split by rows
  - vertical fragmentation — tables are split by columns