CS 443
Database Management Systems

Slides adapted from Ramakrishnan & Gerhke pages.cs.wisc.edu/~dbbook/

Renée J. Miller

Course Information

- Text: Database Management Systems 3ed,
  - authors: R. Ramakrishnan and J. Gehrke
- Tentative Marking
  - 3 Term Tests – 20% each
  - 6 Practical Learning Tasks - 5% each
  - Project - 10%
  - Practical learning tasks and project can be done in groups of up to 3 people.

Course Information

- Prerequisite: CSC343H1/434H1, CSC369H1/468H1, 364H1/CSC373H1/CSC375H1; CGPA 3.0
  - Any requests for waiver must be made by email (containing copy of transcript) by Tuesday, Jan 15th
  - I can only consider waivers to CS students, other students must get waivers from their own UG office

Practical Learning

- We'll have 6 programming assignments (PL or practical learning tasks) but we'll be using class time (lecture and tutorial) to turn these into interactive learning opportunities
- Next week 1/14: we suggest bringing laptop to class
  - With TAs we'll walk through steps to mostly solve PL1
  - We'll give you time to work with your group on completing PL1 and starting PL2
  - TAs available to answer questions through-out session (lecture and tutorial time: 3-6pm)

Marking for Practical Learning Tasks

- TAs will do marking interactively with your group in tutorial
  - When your code is ready, ***all*** group members must submit a tar file of code and a file `members.txt` with name and CDF logins of all group members
    - use command `submit` on CDF (http://www.cdf.toronto.edu/)
    - deadline for submission will be posted but early submission welcome
- Your code need only run in your chosen environment (e.g., your laptop or CDF), for marking you'll need to be able to demo the code for TAs & answer questions about it
  - We require submission of code on CDF so we can verify you've done what you've said and so we can check for plagiarism

Database Management Systems

- Systems that manage large collections of structured data that typically provide most of the following
  - Storage management - managing (transparently) movement of data from disks to memory
  - Query processing
  - Query optimization
  - Concurrent access (multiple users) to data
  - Crash recovery (durability/persistence)

Main Features of DBMS

- Data independence
  - Applications are isolated from changes in data formats
  - Reduces application development time
- Isolation
  - Applications are isolated from concurrent use of data by other users
- Reduced application development by providing
  - Uniform data administration, security, etc.
  - Declarative, efficient data access

Why Study Databases??

- Shift from *computation* to *information*
  - at the "low end": scramble to webspace (a mess!)
  - at the "high end": scientific applications
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, EOS project, Linked Open Data
  - ... need for DBMS exploding
- DBMS encompasses most of CS
  - OS, languages, theory, AI, natural language, logic

Data Models

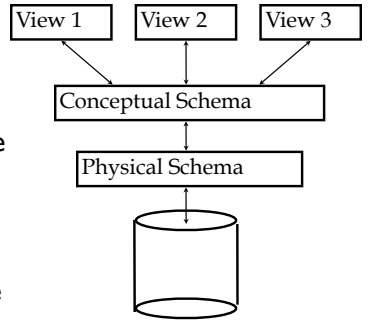
9

- A *data model* is a collection of concepts for describing data.
- A *schema* is a description of a particular collection of data, using the a given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.
  - Every relation has a *schema*, which describes the columns (attributes or fields).
  - Each row is a tuple.

Levels of Abstraction

10

- Many *views*, single *conceptual (logical) schema* and *physical schema*.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.



☛ *Schemas are defined using DDL; data is modified/queried using DML.*

Example: University Database

11

- Conceptual schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):
  - *Course_info(cid:string,enrollment:integer)*

Data Independence *

12

- Applications insulated from how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data.
- *Physical data independence*: Protection from changes in *physical* structure of data.

☛ *One of the most important benefits of using a DBMS!*

Concurrency Control

- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow, it is important to keep the cpu humming by working on several user programs concurrently.
- Interleaving actions of different user programs can lead to inconsistency: e.g., check is cleared while account balance is being computed.
- DBMS ensures such problems don't arise: users can pretend they are using a single-user system.

Transaction: An Execution of a DB Program

- Key concept is *transaction*, which is an *atomic* sequence of database actions (reads/writes).
- Each transaction, executed completely, must leave the DB in a *consistent state* if DB is consistent when the transaction begins.
  - Users can specify *integrity constraints* on the data, and the DBMS will enforce these constraints.
  - Beyond this, the DBMS does not really understand the semantics of the data. (e.g., it does not understand how the interest on a bank account is computed).
  - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user's responsibility!

Scheduling Concurrent Transactions

- DBMS ensures that execution of {T1, ... , Tn} is equivalent to some *serial* execution T1' ... Tn'.
  - Before reading/writing an object, a transaction requests a lock on the object, and waits till the DBMS gives it the lock. All locks are released at the end of the transaction. (Strict 2PL locking protocol.)
  - Idea: If an action of Ti (say, writing X) affects Tj (which perhaps reads X), one of them, say Ti, will obtain the lock on X first and Tj is forced to wait until Ti completes; this effectively orders the transactions.
  - What if Tj already has a lock on Y and Ti later requests a lock on Y? (Deadlock!) Ti or Tj is aborted and restarted or Ti forced to wait.

Ensuring Atomicity

- DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a Xact.
- Idea: Keep a *log* (history) of all actions carried out by the DBMS while executing a set of Xacts:
  - Before a change is made to the database, the corresponding log entry is forced to a safe location. (*WAL protocol*; OS support for this is often inadequate.)
  - After a crash, the effects of partially executed transactions are *undone* using the log. (Thanks to WAL, if log entry wasn't saved before the crash, corresponding change was not applied to database!)

The Log

17

- The following actions are recorded in the log:
  - *Ti writes an object*: The old value and the new value.
    - Log record must go to disk *before* the changed page!
  - *Ti commits/aborts*: A log record indicating this action.
- Log records chained together by Xact id, so it's easy to undo a specific Xact (e.g., to resolve a deadlock).
- Log is often *duplexed* and *archived* on "stable" storage.
- All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.

Summary

18

- DBMS used to maintain, query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- Levels of abstraction give data independence.
- Data is valuable - more so now than ever
- Data Management R&D is one of the broadest, most exciting areas in CS.

Why Study (Big) Data Management?

19

Big Data in popular press

Forget YOLO: Why 'Big Data' Should Be The Word Of The Year
(http://www.npr.org/2012/12/20/167702665/geoff-nunbergs-word-of-the-year-big-data)

"What's new is the way data is generated and processed. It's like dust... We kick up clouds of it wherever we go. Cellphones and cable boxes; Google and Amazon, Facebook and Twitter; cable boxes and the cameras at stoplights; the bar codes on milk cartons; and the RFID chip that whips you through the toll plaza — each of them captures a sliver of what we're doing, and nowadays they're all calling home."

"It's only when all those little chunks are **aggregated** that they turn into **Big Data**; then the software called analytics can scour it for patterns."

Project will let you explore a specific analytics/big data task