COMP3311 Week 09 (Final Lecture)

Databases: Past, Present, Future

Databases: Past/Present/Future

Core "database" goals:

- deal with very large amounts of data (terabytes, petabyes, ...)
- very-high-level languages (deal with big data in uniform ways)
- query optimisation (evaluation too slow ⇒ useless)

Three phases in DBMS history:

- 1960's: pre-history, first attempts at generalised data access
- 1970's-2000's: relational databases, structure, transactions
- 2000's: Big Data, less structure, relaxed transactions

... Databases: Past/Present/Future

1960's: Database Pre-history

- simple structured records
  - atomic-valued records (like relational tuples)
- *hierachical data model*
  - all data organised via tree structure
  - access by following paths in hierarchy
  - could only represent 1:n relationships
- *network data model*
  - relationships between records via links
  - query via traversal of record graphs
  - developed in late 60's, persisted until early 80's

... Databases: Past/Present/Future

1970's: The Rise of Relational DBMS's

- Codd developed relational model
  - sets of tuples, relational algebra, ...
- IBM developed a DBMS based on model (System R)
  - high-level (declarative) query language
  - implementation of efficient relational operations
  - implementation of transactions/recovery
  - System R eventually developed into DB2
- Larry Ellison commercialised the idea (Oracle)

... Databases: Past/Present/Future

1980's: Development of RDBMS's

- distribution/replication of data
- improving implementation of relational operators

1990's: Standardization and Extension

- SQL standard developed
- SQL extended with e.g. PL/SQL, recursion, ...
- improving implementation of relational operators

2000's: New Data

- temporal aspects of data
- stream data (continuous queries)
- high-dimensional data (e.g. image feature spaces)

RDBMS's have been dominant DB technology for 40 years.

... Databases: Past/Present/Future

Assumptions underlying relational DBMSs ...

- data resides on disk device (latency, block-transfer)
- data located on a single machine (?)
- data fits on (large array of) disks on a LAN
- data can be represented via atomic-valued tuples
- queries involve precise matching (e.g. equality, ...)

How things have changed lately ...

- growth in use of SSDs for data storages
- need to store *very* large amounts of data

... Databases: Past/Present/Future

Limitations/pitfalls of RDBMSs:

- NULL is ambiguous: unknown, not applicable, not supplied
- "limited" support for constraints/integrity and rules
- no support for uncertainty (data represents *the* state-of-the-world)
- data model too simple (e.g. no support for complex objects)
- query model too rigid (e.g. no approximate match)
- cotinually changing data sources not well-handled
- data must be "molded" to fit a single rigid schema
- database systems must be manually "tuned"
- do not scale well to some data sets (e.g. Google, Telco's)

... Databases: Past/Present/Future

How to overcome (some of) these limitations?

Extend the relational model ...

- add new data types and query ops for new applications
- deal with uncertainty/inaccuracy/approximation in data

Replace the relational model ...

- object-oriented DBMS ... OO programming with persistent objects
- XML DBMS ... all data stored as XML documents, new query model
- application-effective data model (e.g. *(key,value)* pairs)

Performance: DBMSs that "tune" themselves ...

Big Data

Some modern applications have massive data sets (e.g. Google)

- far too large to store on a single machine/RDBMS
- query demands far too high even if could store in DBMS

Approach to dealing with such data

- distribute data over large collection of nodes (redundancy)
- provide computational mechanisms for distributing computation

Often this data does not need full relational selection

- represent data via *(key,value)* pairs
- unique *key* values can be used for addressing data
- *values* can be large objects (e.g. web pages, images, ...)

... Big Data

Popular computational approach to Big Data: *map/reduce*

- suitable for widely-distributed, very-large data
- allows parallel computation on such data to be easily specified
- distribute (map) parts of computation across network
- compute in parallel (possibly with further map'ing)
- merge (reduce) multiple results for delivery to requestor

Some Big Data advocates see no future need for SQL/relational ...

- depends on application (e.g. hard integrity vs eventual consistency)

Humour: Parody of noSQL fans (strong language warning)

Information Retrieval

DBMSs generally do precise matching (although `like`/regexps)

Information retrieval systems do approximate matching.

E.g. documents containing specified words (Google, etc.)

Also introduces notion of "quality" of matching
(e.g. tuple T_1 is a *better* match than tuple T_2)

Quality also implies *ranking* of results.

Much activity in incorporating IR ideas into DBMS context.

Goal: support database exploration better.

Multimedia Data

Data which does not fit the "tabular model":

- image, video, music, text, ... (and combinations of these)

Research problems:

- how to specify queries on such data? ($image_1 \cong image_2$)
- how to "display" results? (synchronize components)

Solutions to the first problem typically:

- extend notions of "matching"/indexes for querying

- require sophisticated methods for capturing data features

Sample query: find other songs *like* this one?

Uncertainty

Multimedia/IR introduces approximate matching.

In some contexts, we have approximate/uncertain data.

E.g. witness statements in a crime-fighting database

"I think the getaway car was red ... or maybe orange ..."

"I am 75% sure that John carried out the crime"

Work by Jennifer Widom at Stanford on the *Trio* system

- extends the relational model (ULDB)
- extends the query language (TriQL)

Stream Management Systems

Makes one addition to the relational model

- *stream* = infinite sequence of tuples, arriving one-at-a-time

Applications: news feeds, telecomms, monitoring web usage, ...

RDBMSs: run a variety of queries on (relatively) fixed data

StreamDBs: run fixed queries on changing data (stream)

Approaches:

- *window* = relation formed from a stream via a rule
- *stream data type* = build new stream-specific operations

Semi-structured Data

Uses *graphs* rather than tables as basic data structure tool.

Applications: complex data representation, via "flexible" objects, e.g. XML

Graph nature of data changes query model considerably.

(e.g. Xquery language, high-level like SQL, but different operators, etc.)

Implementing graphs in RDBMSs is often inefficient.

Research problem: query processing for XML data.

Dispersed Databases

Characteristics of dispersed databases:

- very large numbers of small processing nodes
- data is distributed/shared among nodes

Applications: environmental monitoring devices, "intelligent dust", ...

Research issues:

- query/search strategies (how to organise query processing)
- distribution of data (trade-off between centralised and diffused)

Less extreme versions of this already exist:

- grid and cloud computing
- database management for mobile devices

Looking ahead

Every so often, DBMS researchers meet to consider the field:

- Laguna Beach, 1989 ... http://doi.acm.org/10.1145/382272.1367994
- Asilomar, 1998 ... http://doi.acm.org/10.1145/306101.306137
- Claremont, 2008 ... http://doi.acm.org/10.1145/1462571.1462573
- Beckman, 2016 ... http://doi.acm.org/10.1145/2845915

Regular attendees: Rakesh Agrawal (IBM), Phil Bernstein (MS), Mike Carey (BEA), Stefano Ceri (Pisa), David deWitt (MS), Michael Franklin (UCB), Hector Garcia-Molina (Stanford) Jim Gray (MS), Laura Haas (IBM), Alon Halevy (Google) Joe Hellerstein (UCB), Mike Lesk (Bell), David Maier (PSU), Raghu Ramakrishnan (Yahoo), Avi Silberschatz (Bell), Rick Snodgrass (Arizona), Mike Stonebraker (UCB/MIT), Jeff Ullman (Stanford), Jennifer Widom (Stanford), ...

Beyond COMP3311

COMP9315 Database Systems Implementation

- comprehensive study of DBMS internals

COMP9318 Data Warehousing and Data Mining
- data summarisation/discovery techniques

COMP9319 Web Data Compression and Search
- Web compression and searching algorithms

COMP6714 Information Retrieval and Web Search
- finding information in unstructured text

Course Review/Exam Preview

COMP3311 Course Aims

At the end of this course you should be able to:

- develop a (correct, non-redundant) data model for a simple application
- implement data models using SQL database management systems
- devise effective queries to answer information requests on a database
- enhance the power of the database via stored procedures and triggers
- implement PHP scripts to efficiently access a database
- understand the overall architecture of modern relational DBMSs
- understand foundations of transaction and query processing

Syllabus Overview

1. Data modelling and database design
   - Entity-relationship (ER) design, relational data model
   - Relational theory (algebra, dependencies, normalisation)
2. Database application development
   - SQL for querying and data definition (PostgreSQL's version)
   - PostgreSQL, `psql` (an SQL shell), PLpgSQL (procedural SQL)
   - PHP (DB access)
3. DBMS technology
   - Performance tuning, catalogues, access control
   - DBMS architecture, query processing, transactions

Things in gray will **not** be examined.

Assessment Summary

Your final mark/grade will be determined as follows:

```
a1      = mark for assignment 1            (out of 15)
a2      = mark for assignment 2            (out of 15)
quizzes = mark for top 3 on-line quizzes   (out of 10)
examP   = mark for exam (practical)        (out of 30)
examW   = mark for exam (written)          (out of 30)

exam    = examP + examW              (out of 60)
okExam  = examP >= 12 && examW >= 12 (after scaling)

mark   = a1 + a2 + quizzes + exam
grade  = HD|DN|CR|PS  if mark >= 50 && okExam
       = FL           if mark <  50 && okExam
       = UF           if !okExam
```

Final Exam

D-day: Tuesday 7 May

- held in computer labs e.g., K17, Physics, ElecEng labs
- 3-hour morning session for some students
- 3-hour afternoon session for the rest
- random allocation to sessions/labs
  - I know about clash students
  - if you have a preference, let me know (there will be an online form available soon)

Session/lab allocations on-line by end of next week

... Final Exam

3 hours, 90 marks 10 questions (5 Written, 5 Prac)

Past Exams are attached to the course web site.

Differences between 19s1 and older exams:

- 06s2 exam was way too long (old style)
- 12s1 had 12 questions (6+6)

19s1 exam has more emphasis on "practicality" than "theory"

... Final Exam

Written Questions: (worth 50% of exam mark, hurdle 12/30)

- cover a range of topics (incl. PLpgSQL and PHP)
- are like online exercises (but no recycling)
- partial marks for partial solutions
- no marks for writing nothing or "I don't know"
- negative marks for writing irrelevant/rubbish

If you don't know the answer, don't write anything.

... Final Exam

Prac Questions: (worth 50% of exam mark, hurdle 12/30)

- are like SQL questions from previous exams
- vaguely like Assigt SQL, but DB much simpler/smaller
- write SQL queries/views (using **SQLite**)
- no PLpgSQL or PHP in prac exam
- some (not many) marks for incomplete solutions

... Final Exam

Exam working environment:

- using standard CSE exam environment
- the usual collection of editors (e.g. pico, gedit, vim, emacs)
- `xfig` and `dia` for drawing ER diagrams, if any
- `sqlite3` for doing the SQL prac questions
- from within the web browser, you will have
  - exam paper (instructions and all questions)
  - SQLite3, PostgreSQL, PHP docs (copied from course web site)
- all answers will be typed/drawn into files
- submit answers via `give` (special exam version)

Revision

Sources for revision material:

- COMP3311 Course Notes, Theory Exercises
- *Fundamentals of Database Systems*, Elmasri/Navathe
- *Database System Concepts*, Silberschatz/Korth/Sudarshan
- *Database Management Systems*, Ramakrishnan/Gehrke
- *Database Systems: Complete Book*, Garcia-M/Ullman/Widom
- *Database Systems: App-oriented*, Kifer/Bernstein/Lewis
- SQLite/PostgreSQL/PHP Documentation (learn to navigate)

... Revision

Strategy for revision:

- attempt the *past exams* (available on Web)

  (no questions from past exams will be repeated on this year's exam)

- review *exercises* and assignments

  (no exercise/assignment questions will be repeated on the exam)

- come to the tuts (Tue & Wed on week 10, and also Tue on week 11) to resolve problems

  (Check the Web timetable for tut times and locations.)

Supplementary Exams

Suplementary Exams are only available to people who

- are absent from the Final Exam with *good reason*
  (good = documented, serious, clearly affects ability to do exam)
- have performed well during the rest of the semester

If you are awarded a Supp Exam ...

- you **must** make yourself available for it
- non-attendance at the Supp ⇒ mark of 0 for the exam

Assessment

Assessment is about determining how well *you* understand the syllabus of this course.

If you can't *demonstrate your understanding*, you won't pass.

In particular, we don't pass people just because ...

- please, please, ... my parents will be ashamed of me

- please, please, ... I tried *really hard* in this course
- please, please, ... I'll be excluded if I fail COMP3311
- please, please, ... if I fail this, I can't do COMP9xxx
- etc. etc. etc.

... Assessment

Of course, assessment isn't a "one-way street" ...

- I get to assess you in the final exam
- you get to assess me in MyExperience

MyExperience evaluations are online now.

Telling me good things is fine.

Telling me things I did wrong is helpful.

Some Thoughts ...

You need to learn for life, not just the exam.

In particular, learn to find answers for yourself.

No single correct answer. (Solutions range from poor to excellent)

Take *pride* in your work. (Aim for quality, not just correctness)

Finally ...

Good Luck with the Exams ... and Life ...

Produced: 18 Apr 2019