Course Wrap-Up
Admin stuff

• We will post A3 results as soon as they’re ready, (once the deadline passes and we get to marking)

• Please make sure to submit all under your A3 folder, no subfolders and such

• Marks for in-class exercises and A2 to be released very soon
  • Keep monitoring Piazza

• Office hours will continue as scheduled (Thursday 2-4pm), even if Tuesday is the last day of classes
  • Come with questions about anything you might have unclear when prepping for the exam
Preparing for the exam

• Re-solve parts of the assignments where you didn’t get full marks.
• For topics you aren’t fully confident in, re-do the lecture prep and in-class exercises.
• Go over solutions for in-class exercises
• Make up your own queries in RA, SQL, XQuery to hit on query types and language features you need practise in.
• Solve old tests and finals.
  • Publicly available on the UofT repository
  • Don’t count just on these for prepping though
The Final

• Comprehensive (covers the whole term), including:
  • RA
  • SQL and JDBC
  • DTDs, XML, XQuery
  • FD theory and normalization
  • ER modelling and DB design
The Final

• You need to know the syntax of each language.
• You don’t need to memorize function/method APIs. We will provide what you need and/or be forgiving when marking those details.
• SQL: views are always welcome, as long as correct.
• Comments are usually unnecessary, unless we say otherwise.
• Questions may be similar to previous tests and final exams, but don’t count on that!
The Final

• Typically, it’s about 24 pages long, but
  • A page towards the end is empty (for rough work)
  • Last 2 pages: the schemas for reference (you can detach this last sheet, for convenience – do not detach anything else though)
• Page 1 is the cover
  • Lots of empty space to fill in your answers
• So it’s really 20 pages, with lots of white space
• You need 40% on the final to pass the course, regardless of the rest of the term marks
The Final

• When and where:
  • [http://www.artsci.utoronto.ca/current/exams/dec15](http://www.artsci.utoronto.ca/current/exams/dec15)

<table>
<thead>
<tr>
<th>Course</th>
<th>Section</th>
<th>Date</th>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC343H1F</td>
<td>A - MC</td>
<td>TUE 15 DEC</td>
<td>EV 7:00 - 10:00</td>
<td>BN 2N</td>
</tr>
<tr>
<td>CSC343H1F</td>
<td>ME - Z</td>
<td>TUE 15 DEC</td>
<td>EV 7:00 - 10:00</td>
<td>BN 2S</td>
</tr>
<tr>
<td>CSC369H1F</td>
<td>A - T</td>
<td>FRI 11 DEC</td>
<td>EV 7:00 - 10:00</td>
<td>BN 3</td>
</tr>
<tr>
<td>CSC369H1F</td>
<td>U - Z</td>
<td>FRI 11 DEC</td>
<td>EV 7:00 - 10:00</td>
<td>ST VLAD</td>
</tr>
<tr>
<td>CSC373H1F</td>
<td></td>
<td>THU 17 DEC</td>
<td>EV 7:00 - 10:00</td>
<td>BN 3</td>
</tr>
</tbody>
</table>

• Clara Benson Building (BN), 320 Huron Street
• Next to Athletic Centre!
• EV == evening! So, it’s at 7-10PM, not AM!
Course wrap-up - Lessons learned

• What do I take away from this course?
  • Data models are important
  • Relational model: concept of relation/table
  • Schema vs. instance!
  • Keys, integrity constraints
  • RA: foundation of SQL
  • SQL: DML, DDL, expressive power/limitations
  • Embedded SQL: more control, addresses SQL limitations by combining it with a conventional language
  • XML/DTD + Xpath/Xquery – not all data fits a rigid schema; unstructured data needs another representation
  • Database design theory: client requirements => E/R diagram => relational schema.
  • FD theory helps bring schema into a normal form
Expressive power

• $\text{RA} \subseteq \text{SQL} \subseteq \text{Xquery}$
• You can sometimes do surprisingly much with what appears to be very little
Theme: expressive power

- Expressive power vs Computational complexity
  - XQuery can’t do all the optimizations a relational DBMS can do.
  - SQL general assertions vs intra-table constraints.
  - SQL triggers are a tradeoff.

- Expressive power vs Language complexity
  - RA vs SQL vs XQuery:
    How complex are expressions/statements in the language?
  - But is some of the difference just good/bad design?
Theme: classic tradeoffs

• Time vs space
  • For long-running queries where results might be reused, store snapshots of computed results to save time.
    • Time saved in future queries vs. space taken up
    • Materialized views (also, keep up-to-date issue)
  • The DBMS stores indices to speed up operations
    • B-trees, R-trees, etc. speed up operations – no longer need to search the entire table, but takes up space.

• Time vs time
  • Tradeoff time to build and maintain indices against speed-up of other operations
  • Sequential vs random I/O: are indexes always better?
Theme: other tradeoffs

• Rigidity of SQL vs Flexibility of XML
• No redundancy vs Preserve dependencies
• No redundancy vs Performance
Why should I care?

• In the era of Big Data, having knowledge of database systems is really important! E.g.,
  • What type of data do I have? Structured vs. unstructured
  • How do I start designing a database?
  • How do I optimize my design?
  • How do I use a database system?
• Database systems are all part of a bigger picture
CSC443

• “Database System Technology”
• Takes the perspective of building a DBMS.
• Internals of a DBMS
• Topics like:
  • Memory management - bufferpool
  • Indices – e.g., Btrees
  • Query optimization – produce good query plans
  • Managing storage – row-oriented, column-oriented, etc.
  • Concurrency control
  • Transaction management – ACID properties
  • Tuning for performance
  • Types of workloads: OLTP, OLAP, etc.
  • Data mining
Trends in DB Research

• Managing huge amounts of data: approximate querying, statistical methods, self-tuning, power management
• NoSQL technologies
• Stream processing
• Data mining
• Data privacy and security
• Different kinds of data, e.g., spatial, temporal, data from sensors, social network data, graph databases
• New languages and interfaces
• Visualization of data
• Top-tier database conferences: VLDB, SIGMOD, ICDE, EDBT, CIDR, CIKM, SIGSPATIAL GIS, IEEE BigData
NoSQL – why should I check it out?

DO YOU HAVE ANY EXPERTISE IN SQL?

NO

DOESN'T MATTER. WRITE: "EXPERT IN NO SQL."

Leverage the NoSQL boom
BigData – no “one-size fits all”

• Relational databases are not always a good fit
• No “one-size fits all”
  • Typical OLTP database: less than 1TB of data
  • Google: 900 TB of search engine data (mostly unstructured!)
  • Youtube: 80PB video data/year
  • Scientific data
    • US department of energy: 3.5PB
  • New application domains
    • Stream processing
    • Social media (graph data)
Need for scale and performance

• Scaling up
  - Issues with scaling up when the dataset is just too big
  - RDBMS not designed to be distributed
    • Parallel DBMSs exist though: partitioning + replication strategies, distributed query processing, joins across multiple nodes, etc.
  - Cost effective strategy: ‘scaling out’ or ‘horizontal scaling’

• Some applications need very few database features
  - SQL may be too heavy-weight; does not need fancy indexing, just fast lookup by primary key
    • You might have heard about Key-value stores!
  - High scalability: need to be able to handle when traffic spike happens!
  - High availability: need to be able to handle crashes; graceful recovery
    • Need to continue to provide service - Cost of downtime is extremely high!
    • Facebook down in 2010 for 3 hours => $1M in lost ad revenues
    • Paypal down in 2009 for 4 hours (network H/W failure) => $7.2M in lost transactions
Need for flexible data model

• Relational schema: too rigid
  • No way to change dynamically
  • Need a DB admin to “stop the world” and change the schema, migrate the data in the new structures, etc.

• Many applications’ data: no fixed structure
  • Log processing
  • Stream processing
  • Graph processing

• Data model should not restrict data access!
NoSQL advantages

• Cheap, easy to implement (open source)

• Data is replicated to multiple nodes (availability and fault-tolerance) and can be partitioned
  • Down nodes easily replaced
  • No single point of failure

• Can scale up and down

• Doesn't require a schema
  • Not really.. :)}
What are we sacrificing instead?

• Decades of database optimizations (carefully-designed query optimizers, indexing, etc.)

• Joins

• ACID Transactions

• SQL, powerful expressive query language (mostly)

• Easy integration with other applications that support SQL
NoSQL categories

- **Document store**
  - Key
  - Document (collection of key-values)

- **Column store**
  - Key
  - CF1: C1
  - CF1: C2
  - CF2: C1
  - CF3: C1

- **Key-value store**
  - Key
  - Binary Data

- **Graph store**
  - Node 1
    - Key
    - Properties
  - Node 2
    - Key
    - Properties
  - Relationship 1
    - Key
    - Properties
# Examples

<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document store</td>
<td>CouchDB, MongoDB</td>
</tr>
<tr>
<td>Column store</td>
<td>Cassandra, HBASE</td>
</tr>
<tr>
<td>Key-value store</td>
<td>Redis, Riak</td>
</tr>
<tr>
<td>Graph store</td>
<td>InfiniteGraph, Neo4j</td>
</tr>
</tbody>
</table>
Should I be using NoSQL Databases?

• NoSQL data storage systems makes sense for applications that need to deal with very large semi-structured data
  • Log Analysis
  • Social Networking Feeds

• Most of us work on organizational databases, which are not that large and have low update/query rates
  • Regular relational databases are the right solution for most such applications
Concluding remarks

• I hope you’ve enjoyed the course.
• I hope you’ve found it interesting and that it will help you in your future career / job prospects

• Good luck with the final exam!
  • Don’t panic! You’ve got this!
  • This is your chance to show us what you’ve learned
  • We WANT to give you the credit that you’ve earned

• Thank you for a great class, it’s been a real pleasure teaching you this term!
The end..