Database Management Systems
Course 236363
Lecture 1: Introduction

Course Lecturer

- Ph.D. HebrewU (DB uncertainty & search, XML)
- Prof. Technion since Jan 2015
- Past positions:
  - IBM Research in Almaden (San Jose, CA)
  - CS Theory group, text-analytics group
  - LogicBlox (Berkeley, CA)
  - Logic-based solution for data management & analysis
- Research:
  - Next generation DB: Infrastructure for text analytics, DB uncertainty, data cleaning, ML & DB
  - Graph mining, graph IR

What is a Database?

- Simply, a persistent (cross session) repository of data
  - Models differ in what “data” means, especially how to separate content from structure; Tables? Graphs? Trees? Objects? Maps?
- Data Base Management System (DBMS): A software system for creating, maintaining, updating, and querying the database
  - General purpose—not for any specific application
  - Interacts with a user (e.g., DBA) or an application
- Challenges:
  - Modeling (data, languages, consistency, security)
  - Engineering
  - Efficiency & scalability

Why are Databases Needed?

- Facilitate (save time & skills)
  - Program in high levels of abstraction (concepts, entities, relationships, etc.)
  - No need for in-house implementation
  - Storage, disk, persistency, recovery, security, algs, etc.
  - Easier to accommodate architecture changes
  - Democratize data management (not only experts)
- Boost performance (often)
  - Adopt optimization & hardware utilization programmed already by the database vendor
- Safer software
  - The chance of bugs & security leaks reduces dramatically (past users suffered for us)

Where are you using databases, directly or indirectly, in your life?
The IMDb Application

Steps in Database Setup

- Requirement analysis
  - What information needs to be stored?
  - How will it be used?
- Conceptual database design
  - Define/describe/discuss the semantic modeling of data in the application
- Logical database design
  - Translate conceptual design into a database schema
- Physical database design
  - Translate the database schema into a physical storage plan on available hardware (done by DBMS)

Faculty Example

- Design a database for the faculty’s administrative assistants
- Several types of entities
  - Student: student name, id, address
  - Course: name, catalogue number, lecturer
  - Lecturer? Faculty? Building? Academic track?
  - Depending on the application needs
- Various relationships
  - Student took course (and got a grade)

Data Modeling

More Detailed?

Type Inheritance?
Which Constraints Should Hold Here?

Integrity Constraints

- Schema-level (data-independent) specifications on how records should behave beyond the relational structure
  - (e.g., students with the same ID have the same name, take the same courses, etc.)
  - DBMS guarantees that constraints are always satisfied, by disabling actions that cause violations

Option 1: Single Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Hall</th>
<th>sID</th>
<th>Name</th>
<th>Course.number</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Boris</td>
<td>Haifa</td>
<td>319</td>
<td>Nesher</td>
<td>DB</td>
<td>73</td>
</tr>
<tr>
<td>2345</td>
<td>Boris</td>
<td>Haifa</td>
<td>319</td>
<td>PL</td>
<td>Barak</td>
<td>82</td>
</tr>
</tbody>
</table>

Drawbacks:
- Cost & Redundancy: Why should the student's address be stored in each course she takes?
- Incompleteness: What about students that do not take any courses? If we use an external web students?
- Harder to maintain: If a student changes address, need to update all records of relevant tuples; risk inconsistency or require more expensive controls
- Harder to maintain: If we wish to add the a semester column, every app will need to update its schema assumption

Option 2: Multiple Tables

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Hall</th>
<th>number</th>
<th>Lecturer</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Boris</td>
<td>Haifa</td>
<td>319</td>
<td>Barak</td>
<td>73</td>
</tr>
<tr>
<td>2345</td>
<td>Boris</td>
<td>Haifa</td>
<td>319</td>
<td>PL</td>
<td>82</td>
</tr>
</tbody>
</table>

Relational Design

Option 1: Single Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Hall</th>
<th>number</th>
<th>Lecturer</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Boris</td>
<td>Haifa</td>
<td>319</td>
<td>Barak</td>
<td>73</td>
</tr>
<tr>
<td>2345</td>
<td>Boris</td>
<td>Haifa</td>
<td>319</td>
<td>PL</td>
<td>82</td>
</tr>
</tbody>
</table>

Option 2: Multiple Tables

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Hall</th>
<th>number</th>
<th>Lecturer</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Boris</td>
<td>Haifa</td>
<td>319</td>
<td>Barak</td>
<td>73</td>
</tr>
<tr>
<td>2345</td>
<td>Boris</td>
<td>Haifa</td>
<td>319</td>
<td>PL</td>
<td>82</td>
</tr>
</tbody>
</table>

Why Schema-Level Constraints?

- Maintenance: consistency assured w/o custom code
- Development complexity: no reliance on consistency tests
  - But exceptions need to be handled
- Optimization: operations may be optimized if we know that some constraints hold
  - (e.g., once a sought student ID is found, you can stop; you won’t find it again)

Querying: Which Courses Avia Took?

```sql
SELECT C.name, T.grade FROM S, T, C WHERE S.sID = 'Avia' AND S.ID = T.sID AND T.cNum = C.number
```

```python
for v in T:
    for c in C:
        if s.name='Avia' and c.number=t.cNum:
            print s.name
```

```assembly
mov rdi, s.ID
mov rax, C.number
mov rsi, T.cNum
mov rdx, S.sID
cmp rax, rdx
je exit_loop
```

Logic (RCA)

```python
logic1 = lambda x,y,z : (x in S and y in T and z in C)
logic2 = lambda x,y,z : (x in S and y in T and z in C and x in R)
```

Logic Programming (Datalog)

```python
logic1 = lambda x,y,z : (x in S and y in T and z in C)
logic2 = lambda x,y,z : (x in S and y in T and z in C and x in R)
```

Algebra (RA)

```python
logic1 = lambda x,y,z : (x in S and y in T and z in C)
logic2 = lambda x,y,z : (x in S and y in T and z in C and x in R)
```
What is a Query Language?

- A language for specifying how desired information is retrieved/derived from the database
- Usually, does not change the database
  - At least not the user-defined tables
- Specialized to the database model
  - As opposed to a general programming language
- In contrast, a Data Definition Language (DDL) is a language for manipulating (creating / updating / deleting) schemas and data

“Goodness” of a Query Language

- **Simple**
  - Users: easier to use
  - DBMS: easier to implement, easier to optimize
- **High-level**
  - Declare what, not program how
  - Users: easier, less control
  - DBMS: more flexibility, more responsibility
- **Expressive**
  - NOT: predefined queries; YES: ops w/ composition
  - Users: better
  - DBMS: harder to implement/optimize

Other Data Models: XML

```xml
<?xml version="1.0"?>
<students id="100026">
  <student id="100078">
    <name>Jack</name>
    <age>21</age>
    <major>Physics</major>
  </student>
  <student id="100026">
    <name>Joe</name>
    <age>21</age>
    <major>Math</major>
  </student>
</students>
```

Other Data Models: RDF (Linked Data)

http://www.w3.org/2004/REC-rdf-primer-20040210/

NoSQL Databases

- Really, “no general relations”
- A collection of restricted/specialized database models to allow for scalability / distribution
  - Key-value store: specialized for hash tables
  - Document store: similar to key-value, but values have an internal structure (e.g., XML, JSON)
  - Graph databases: specialized for graphs with “nodes” and “edges;” queries tailored to traversal

Why NoSQL rather than RDB?

- Same as “Why DB rather than PL”?
  - Translate to scalable execution plans
  - Utilize modern hardware architectures
    - Multi core, multi CPU, multi machines (cluster), Hadoop (HDFS+Map-Reduce), etc.
  - Easier to develop / phrase queries
**Why Now?**

- **Social trends**
  - Online activity: social nets, blogs, ...
- **Technological trends**
  - Mobile computing, "wearable computing," "Internet of Things," ...
- **Business trends**
  - Social-network marketing, cloud computing, ...

---

**Pre-Relational Databases**

- Cross-app solutions for data store/access proposed already in the 1960s
  - **Examples:**
    - The CODASYL committee standardized a network data model (Codasyl Data Model)
      - A network of entities linked to each other, very similar to object-oriented models
    - **Integrated Data Stores** (Charles Bachman)
    - IBM’s IMS, driven by the Apollo program
      - Hierarchical data model; focused mainly on storage interface; low-level access to retrieve record segments

---

**Codd’s Vision (1)**

- 1970: Codd invents the relational database model
  - Idea:
    - Data stored as a collection of relations, connected by keys
    - Relations conform to a schema
    - Questions via a query language over the schema
    - System translates queries into actual execution plans
  - Principle: separate logical from physical layers
  - Work done in IBM San Jose, now IBM Almaden

---

**Codd’s Vision (2)**

- 1970-1972: Codd introduced the relational algebra and the relational calculus
  - Algebraic and logical QLs, respectively
  - Proves their equal expressive power

---

**Codd Catches On (1)**

- 1973: Michael Stonebraker and Eugene Wong implement Codd’s vision in INGRES
  - Commercialized in 1983
  - Evolved to Postgres (now PostgreSQL) in 1989
Codd Catches On (2)

- 1974: A group from the IBM San Jose lab implements Codd’s vision in System R, which evolved to DB2 in 1983
- SQL initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce
- 1977: Influenced by Codd, Larry Ellison founds Software Development Labs
  - Becomes Relational Software in 1979
  - Becomes Oracle Systems Corp (1982), named after its Oracle database product

Publication Venues for DB Research

- Conferences:
  - SIGMOD: ACM Special Interest Group on Management of Data (since 1975)
  - VLDB: Intl. Conf. on Very Large Databases (since 1975)
  - ICDE: IEEE Intl. Conf. on Data Engineering (since 1984)
  - ICDT: Intl. Conference on Database Theory (since 1986)
- Journals:
  - TODS: ACM Transactions on Database Systems (since 1976)
  - VLDBJ: The VLDB Journal (since 1990)
  - SIGMOD REC: ACM SIGMOD Record (since 1969)

Selected Database Research Topics*

<table>
<thead>
<tr>
<th>Database Security</th>
<th>Data Models</th>
<th>Data Exchange</th>
<th>Data Models</th>
<th>Query Analysis &amp; Optimization</th>
<th>Data Models</th>
<th>Data Models</th>
<th>Data Models</th>
<th>Data Models</th>
<th>Data Models</th>
<th>Data Models</th>
<th>Data Models</th>
<th>Data Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Access</td>
<td>View</td>
<td>Graph</td>
<td>In-Memory</td>
<td>Data Integrity</td>
<td>Key-Value</td>
<td>Relational</td>
<td>XML</td>
<td>In-Memory</td>
<td>Key-Value</td>
<td>Relational</td>
<td>XML</td>
<td>In-Memory</td>
</tr>
<tr>
<td>Security</td>
<td>Access</td>
<td></td>
<td></td>
<td>Security</td>
<td>XML</td>
<td>SQL</td>
<td></td>
<td>Access</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Access</td>
</tr>
<tr>
<td>Query</td>
<td>Analysis</td>
<td></td>
<td></td>
<td>Security</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Analysis</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Analysis</td>
</tr>
<tr>
<td>Privacy</td>
<td>Security</td>
<td></td>
<td></td>
<td>Security</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Privacy</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Privacy</td>
</tr>
<tr>
<td>Integrity</td>
<td>Security</td>
<td></td>
<td></td>
<td>Security</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Integrity</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Integrity</td>
</tr>
<tr>
<td>Database</td>
<td>Security</td>
<td></td>
<td></td>
<td>Security</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Database</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Database</td>
</tr>
<tr>
<td>Management</td>
<td>Security</td>
<td></td>
<td></td>
<td>Security</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Management</td>
<td>SQL</td>
<td>SQL</td>
<td></td>
<td>Management</td>
</tr>
</tbody>
</table>

Turing Awards for DB Technology

- 1973: CHARLES WILLIAM BACHMAN
  - 1981: EDGAR F. (“ED”) Codd
  - 1998: JAMES (“JAY”) NICHOLAS GRAY
  - 2014: MICHAEL STONEBRAKER

Core Topics

1. Database meta modeling & design
2. Relational Databases
   - Querying: SQL, Algebra, Logic
   - Integrity & design theory
3. Additional models
   - XML
   - Semantic Web (RDF)
   - NoSQL

THE COURSE
FYI: Complementary Courses

- 234322 Information Storage Systems
  - Used to be “File Systems”
  - Relevant content: files and disks, secondary-memory computation, database indexes, query-plan optimization (single node, MR), concurrency control, database recovery
- 236605 Uncertainty in Databases
  - (Advanced Topics in CS, Spring semester)
  - Nulls and missing information, inconsistent databases, probabilistic databases

Course Staff

- Lecturer
  - Benny Kimelfeld
  - Taub 703, bennyk@technion.ac.il
  - Reception on Tuesdays, 3:30-4:30
- TAs
  - Roni Licher (in charge)
    - Taub 225, ronili@cs.technion.ac.il
  - Nimrod Raifer
    - Taub 412, nimrodcampus.technion.ac.il
  - Hadar Frenkel
    - Taub 315, hadarh@gmail.com

Course Requirements

- Exam (70%), home assignments (30%)
- Four assignments
  - 2 x dry (5% each)
  - 1 x wet (10%)
  - 1 x wet+dry (10%)