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

Frozen (I) (2013)
PG | 102 min | Animation, Adventure, Comedy | 27 November 2013 (USA)

Your rating:

7.6/10
Ratings: 7.6/10 from 369,636 users  Metascore: 74/100
Reviews: 876 user | 401 critic | 43 from Metacritic.com

When the newly crowned Queen Elsa accidentally uses her power to turn things into ice to curse her home in infinite winter, her sister, Anna, teams up with a mountain man, his playful reindeer, and a snowman to change the weather condition.

Directors: Chris Buck, Jennifer Lee
Writers: Jennifer Lee (screenplay), Hans Christian Andersen (inspired by the story "The Snow Queen" by), 4 more credits »
Stars: Kristen Bell, Idina Menzel, Jonathan Groff | See full cast and crew »

Won 2 Oscars. Another 76 wins & 53 nominations. See more awards »
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

An Entity-Relationship Diagram (ERD)
More Detailed?

Student
- Name
- ID
- Phone
- Address

Course
- Name
- Number
- Semester
- Year

Took
- Semester
- Year

Lecturer
- Name
- ID
- Phone
- Address
- emp#
Type Inheritance?

- Student
  - Took
    - Year
    - Semester
  - Course
    - Name
    - Number
  - Lecturer
    - emp#
  - Person
    - ISA
    - ISA
    - Phone
    - ID
    - Name
    - Address
Option 1: Single Table

**StudentCourseRegistry**

<table>
<thead>
<tr>
<th>sID</th>
<th>sName</th>
<th>sAddr</th>
<th>cNum</th>
<th>cName</th>
<th>cLecturer</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Avia</td>
<td>Haifa</td>
<td>363</td>
<td>DB</td>
<td>Anna</td>
<td>95</td>
</tr>
<tr>
<td>1234</td>
<td>Avia</td>
<td>Haifa</td>
<td>319</td>
<td>PL</td>
<td>Barak</td>
<td>82</td>
</tr>
<tr>
<td>2345</td>
<td>Boris</td>
<td>Nesher</td>
<td>319</td>
<td>PL</td>
<td>Barak</td>
<td>73</td>
</tr>
</tbody>
</table>

**Advantages?**

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? Course w/o 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
Relational Design

Option 1: Single Table

**StudentCourseRegistry**

<table>
<thead>
<tr>
<th>sID</th>
<th>sName</th>
<th>sAddr</th>
<th>cNum</th>
<th>cName</th>
<th>cLecturer</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Avia</td>
<td>Haifa</td>
<td>363</td>
<td>DB</td>
<td>Anna</td>
<td>95</td>
</tr>
<tr>
<td>1234</td>
<td>Avia</td>
<td>Haifa</td>
<td>319</td>
<td>PL</td>
<td>Barak</td>
<td>82</td>
</tr>
<tr>
<td>2345</td>
<td>Boris</td>
<td>Nesher</td>
<td>319</td>
<td>PL</td>
<td>Barak</td>
<td>73</td>
</tr>
</tbody>
</table>

Option 2: Multiple Tables

**Student**

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Avia</td>
<td>Haifa</td>
</tr>
<tr>
<td>2345</td>
<td>Boris</td>
<td>Nesher</td>
</tr>
</tbody>
</table>

**Course**

<table>
<thead>
<tr>
<th>number</th>
<th>name</th>
<th>lecturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>363</td>
<td>DB</td>
<td>Anna</td>
</tr>
<tr>
<td>319</td>
<td>PL</td>
<td>Barak</td>
</tr>
</tbody>
</table>

**Took**

<table>
<thead>
<tr>
<th>sID</th>
<th>cNum</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>363</td>
<td>95</td>
</tr>
<tr>
<td>1234</td>
<td>319</td>
<td>82</td>
</tr>
<tr>
<td>2345</td>
<td>319</td>
<td>73</td>
</tr>
</tbody>
</table>

Drawback: join required more often...

How can we formalize what “goodness” means?
Need to understand the connection between sID and sName, etc.
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
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)
Which Constraints Should Hold Here?

**Student**

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Avia</td>
<td>Haifa</td>
</tr>
<tr>
<td>2345</td>
<td>Boris</td>
<td>Nesher</td>
</tr>
</tbody>
</table>

**Course**

<table>
<thead>
<tr>
<th>number</th>
<th>name</th>
<th>lecturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>363</td>
<td>DB</td>
<td>Anna</td>
</tr>
<tr>
<td>319</td>
<td>PL</td>
<td>Barak</td>
</tr>
</tbody>
</table>

**Took**

<table>
<thead>
<tr>
<th>sID</th>
<th>cNum</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>363</td>
<td>95</td>
</tr>
<tr>
<td>2345</td>
<td>319</td>
<td>73</td>
</tr>
</tbody>
</table>

- A student cannot get two grades for the same course
- Grade must be > 53 (check constraint)
- No two tuples have the same ID (key constraint)
- Courses with the same number have the same name (functional dependency)
- sID is a Student.ID; cNum is a Course.number (referential constraint)
Querying: Which Courses Avia Took?

<table>
<thead>
<tr>
<th>S</th>
<th>C</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>name</td>
<td>addr</td>
</tr>
<tr>
<td>1234</td>
<td>Avia</td>
<td>Haifa</td>
</tr>
<tr>
<td>2345</td>
<td>Boris</td>
<td>Nesher</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assembly

```assembly
... mov $1, %rax mov $1, %rdi mov $message, %rsi mov $13, %rdx syscall mov $60, %rax xor %rdi, %rdi ...
```

Python

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

QL

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

Logic (RC)

```sql
\{\langle x \rangle | \exists y, n, z, l, g
 [S(y,n,'Avia') \land C(z,x,l) \land T(y,z,g)]\}
```

Logic Programming (Datalog)

```sql
Q(x) ← S(y,n,‘Avia’),C(z,x,l),T(y,z,g)
```

SQL

```sql
π_{C.name}(σ_{S.name='Avia', number=cNum, ID=sID}(S×C×T))
```

Algebra (RA)
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

```
<students>
  <student id="100026">
    <name>Joe Average</name>
    <age>21</age>
    <major>Biology</major>
    <results>
      <result course="Math 101" grade="C-"/>
      <result course="Biology 101" grade="C+"/>
      <result course="Statistics 101" grade="D"/>
    </results>
  </student>
  <student id="100078">
    <name>Jack Doe</name>
    <age>18</age>
    <major>Physics</major>
    <major>XML Science</major>
    <results>
      <result course="Math 101" grade="A"/>
      <result course="XML 101" grade="A-"/>
      <result course="Physics 101" grade="B+"/>
      <result course="XML 102" grade="A"/>
    </results>
  </student>
</students>
```
Other Data Models: RDF (Linked Data)

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">
    <rdf:Description rdf:about="http://example.org/courses/6.001">
        <s:students>
            <rdf:Bag>
                <rdf:li rdf:resource="http://example.org/students/Amy"/>
                <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
                <rdf:li rdf:resource="http://example.org/students/Johann"/>
                <rdf:li rdf:resource="http://example.org/students/Maria"/>
                <rdf:li rdf:resource="http://example.org/students/Phuong"/>
            </rdf:Bag>
        </s:students>
    </rdf:Description>
</rdf:RDF>
```

http://www.w3.org/TR/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, ...

http://geekandpoke.typepad.com/
HISTORICAL OVERVIEW
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
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

\[(r_1[2], r_1[3]): \forall p_1, r_1 = \forall p_2, r_2 \exists p_3, r_3 \left( (r_1[1] = r_3[1]) \land (r_3[3] = r_2[1]) \right) \]

Applying the reduction procedure of Section 4.1, we obtain the following defining equations:

\[
S_i = R_i \quad (i=1,2,3) \\
S = S_1 \circ S_2 \circ S_3 \\
T_3 = S[1=6] \land S[8=4]
\]
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

M. Stonebraker
E. Wong
• 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)
  – **PODS**: ACM Symp. on Principles of Database Systems (since 1982)
  – **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)
  – **EDBT**: Intl. Conference on Extending Database Technology (since 1988)

• Journals:
  – **TODS**: ACM Transactions on Database Systems (since 1976)
  – **VLDBJ**: The VLDB Journal (since 1992)
  – **SIGMOD REC**: ACM SIGMOD Record (since 1969)
<table>
<thead>
<tr>
<th>System Design</th>
<th>Database Security</th>
<th>Further XML</th>
<th>Entity Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Distributed, storage, in-memory, recovery</td>
<td>• ER models, normal forms, dependency</td>
<td>• Query eval / optimize</td>
<td></td>
</tr>
<tr>
<td>Query Languages</td>
<td>Database Views</td>
<td>Database Privacy</td>
<td>Information Extraction from Web/text</td>
</tr>
<tr>
<td>• Codasyl, SQL, recursion, nesting</td>
<td>• View-based access</td>
<td></td>
<td>• Utilizing crowd input in databases</td>
</tr>
<tr>
<td>Schema Design</td>
<td>System Optimization</td>
<td>Data Models</td>
<td>Social Networks &amp; Social Media</td>
</tr>
<tr>
<td>• ER models, normal forms, dependency</td>
<td>• Caching &amp; replication</td>
<td>• Streaming data</td>
<td></td>
</tr>
<tr>
<td>Transaction &amp; concur.</td>
<td>Indexing</td>
<td>• Graph data</td>
<td></td>
</tr>
<tr>
<td>DB Performance</td>
<td>Clustering</td>
<td>Benchmarking</td>
<td>Crowdsourcing</td>
</tr>
<tr>
<td>• Query process &amp; opt.</td>
<td></td>
<td>• Data Integration</td>
<td>• Utilizing crowd input in databases</td>
</tr>
<tr>
<td>• Evaluation methods</td>
<td></td>
<td>• Interoperability</td>
<td></td>
</tr>
<tr>
<td>Data Models</td>
<td>Heterogeneity</td>
<td>DB Uncertainty</td>
<td></td>
</tr>
<tr>
<td>• OO, geo, temporal</td>
<td>• Data Integration</td>
<td>• Inconsistency &amp; cleaning</td>
<td></td>
</tr>
<tr>
<td>Logic</td>
<td>Interoperability</td>
<td>• Probabilistic DB</td>
<td></td>
</tr>
<tr>
<td>• Deductive (Datalog)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Integrity/constraints</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incompleteness (null)</td>
<td>Analytics (OLAP)</td>
<td>DB &amp; IR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Multimedia, DNA</td>
<td>• DB for search</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Text, XML</td>
<td>• Search for DB</td>
<td></td>
</tr>
<tr>
<td>Data Models</td>
<td>Mining &amp; Discovery</td>
<td>Schema Matching &amp; Discovery</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Discovering association rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Models</td>
<td>Data Exchange</td>
<td>Data Exchange</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ranking &amp; personalization</td>
<td></td>
</tr>
<tr>
<td>Data Models</td>
<td></td>
<td></td>
<td>Cloud Databases</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Column Stores</td>
</tr>
</tbody>
</table>

* Based on SIGMOD session topics from DBLP
Turing Awards for DB Technology

1973
CHARLES WILLIAM BACHMAN
United States – 1973
CITATION
For his outstanding contributions to database technology.

1981
EDGAR F. ("TED") CODD
United States – 1981
CITATION
For his fundamental and continuing contributions to the theory and practice of database management systems.

1998
JAMES ("JIM") NICHOLAS GRAY
United States – 1998
CITATION
For seminal contributions to database and transaction processing research and technical leadership in system implementation.

2014
MICHAEL STONEBRAKER
United States – 2014
CITATION
For fundamental contributions to the concepts and practices underlying modern database systems.
THE COURSE
1. Database meta modeling & design
2. Relational Databases
   – Querying: SQL, Algebra, Logic
   – Integrity & design theory
3. Additional models
   – XML
   – Semantic Web (RDF)
   – NoSQL
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, nimo@campus.technion.ac.il
  – Hadar Frenkel
    • Taub 315, hadarlh@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%)