Modeling Reality

- The real world:
  - Objects
  - Relationships (between objects)
  - Operations (changing aspects of objects and relationships)
- Example: Bank
  - Objects: customers, branches, employees
  - Relationships: customer-account, account-branch
  - Operations: add customer, add account, deposit, withdrawal, balance, prediction

DBMS: faithful & efficient modeling of aspects of the real world for a specific operation

So, 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? Objects? Maps?
- Database 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, query, consistency, security)
  - Engineering
  - Efficiency & scalability
What Services do Databases Provide?

- Centralized information management
  - Conceptual and physical data manager
- Core operations
  - Access control
  - A “smart” query processor
  - Transaction processing, ACID
  - Recovery
- Interoperability
  - Uniform data access across various platforms
  - Logical-physical independence

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 (2013)
- 25% of 408 votes. Variety, Adventure, Canada
- Katherine Bell, Stephen Root, Jonathan Rhys Meyers
- "Adapted from the story "Four Chris Claus", the story follows the adventures of a young girl who is forced to leave her home in order to save her brother and sister."
- "Written and directed by David Goodwin, the film stars Katherine Bell as Anna, Stephen Root as Kristoff, and Jonathan Rhys Meyers as Hans. The story is set in the fictional kingdom of Arendelle, which has been comically "Frozen" by a powerful ice magic, and Anna must travel across the kingdom to save her sister Elsa from the curse."

- [Watch trailer]
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?

Relational Design

Option 1: Single Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>City</th>
<th>Code</th>
<th>Course</th>
<th>Lecturer</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?

- **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

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

<table>
<thead>
<tr>
<th>ID</th>
<th>sName</th>
<th>sAddr</th>
<th>cNum</th>
<th>cName</th>
<th>Lecturer</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

<table>
<thead>
<tr>
<th>ID</th>
<th>sName</th>
<th>sAddr</th>
<th>cNum</th>
<th>cName</th>
<th>Lecturer</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>2345</td>
<td>Boris</td>
<td>Nesher</td>
<td>319</td>
<td>PL</td>
<td>Barak</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 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.)
  - Schema-level: talks about object and relationship types; not concrete instances
- 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?

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?

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

Algebra (RA)
```
π C.name ((σ S.name = 'Avia' AND S.ID = T.sID
AND T.cNum = C.number)(S ⨉ C ⨉ T))
```

Logic Programming (Datalog)
```
\[ \{ (sID, cNum, grade) \mid sID = 1234, cNum = 363, grade = 95 \} \]
```

Logic (RC)
```
\[ \{ (sID, cNum, grade) \mid sID = 1234, cNum = 363, grade = 95 \} \]
```

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
<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>
    <results>
      <result course="Math 101" grade="A"/>
      <result course="Physics 101" grade="B+"/>
      <result course="XML 102" grade="A"/>
    </results>
  </student>
</students>
```

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/triples with “nodes” and “edges,” queries tailored to traversal
RDF: Triples of the Semantic Web

http://dbpedia.org/page/Technion__Israel_Institute_of_Technology

<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:dbp="http://dbpedia.org/property/"
         xmlns:dbo="http://dbpedia.org/ontology/"
         xmlns:per="http://dbpedia.org/resource/Peretz_Lavie"
         xmlns:tei="http://dbpedia.org/resource/Technion_Israel_Institute_of_Technology">
  <rdf:Description rdf:about="http://dbpedia.org/resource/Technion_Israel_Institute_of_Technology">
    <dbp:students rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">13253</dbp:students>
    <dbp:president rdf:resource="http://dbpedia.org/resource/Peretz_Lavie"/>
  </rdf:Description>
</rdf:RDF>

RDF Example from DBPedia

http://dbpedia.org/resource/Technion_Israel_Institute_of_Technology

There's More! Linked!

http://lod-cloud.net/
Main Course Topics

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

FYI: Complementary Courses

• 236510 Database Systems Implementation
  – Concurrency control, recovery, query processing, distributed DBs & replication, in-depth acquaintance with a commercial system
• 236378 Principles of Managing Uncertain Data
  – Nulls and missing information, inconsistent databases, probabilistic databases
• 234322 Information Storage Systems
  – Used to be “File Systems”
  – Files & disks, secondary-memory computation, DB index, query-plan optimization (single node, MR), concurrency control, recovery

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

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)
  - **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)
Selected Database Research Topics*

<table>
<thead>
<tr>
<th>System Design</th>
<th>Database Security</th>
<th>Entity Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Distributed, storage, recovery</td>
<td>• VIEWS-based access</td>
<td>• Information Extraction from Web/DB</td>
</tr>
<tr>
<td>Query Languages</td>
<td>• INCREMENTAL maintain</td>
<td>Further XML</td>
</tr>
<tr>
<td>• CODASYL, SQL, recursion, nesting</td>
<td>• Caching &amp; replication</td>
<td>• Query &amp; optimize</td>
</tr>
<tr>
<td>Schema Design</td>
<td>System Optimization</td>
<td>Data Models</td>
</tr>
<tr>
<td>• ER models, normal forms, dependency</td>
<td>• Caching &amp; replication</td>
<td>• Streaming data</td>
</tr>
<tr>
<td>Transaction &amp; concurrency</td>
<td>• Inconsistency</td>
<td>Data Models</td>
</tr>
<tr>
<td>• Query process &amp; opt.</td>
<td>• Interoperability</td>
<td>• Graph data</td>
</tr>
<tr>
<td>• Evaluation methods</td>
<td>• Interoperability</td>
<td>DB Uncertainty</td>
</tr>
<tr>
<td>Data Models</td>
<td>Gradually</td>
<td>• Probabilistic DB</td>
</tr>
<tr>
<td>• O/O, graph, temporal</td>
<td>• Data Integration</td>
<td>CB &amp; IR</td>
</tr>
<tr>
<td>Logic</td>
<td>• Deductive (Datalog)</td>
<td>• DB for search</td>
</tr>
<tr>
<td>• Incomplete</td>
<td>• Recursion, nesting</td>
<td>• Search for DB</td>
</tr>
<tr>
<td>Incompleteness</td>
<td>Mining &amp; Discovery</td>
<td>Further XML</td>
</tr>
<tr>
<td>• transactional</td>
<td>• Discovering association rules</td>
<td>• Data Models</td>
</tr>
</tbody>
</table>

1980 1990 2000

* Based on SIGMOD session topics from DBLP

Turing Awards for DB Technology

1973
CHARLES WILLIAM BACHMAN

For the outstanding contributions to database technology.

1981
EDGAR F. (“TED”) COOKE

For his fundamental and continuing contributions to the theory and practice of database management systems.

1998
JAMES (“Jim”) NICHOLAS GRAY

For fundamental contributions to database and transaction processing research and technical leadership in system implementation.

2014
MICHAEL STONEBRAKER

For fundamental work in the development of the concepts and practice underlying modern database systems.