Data Models and Database Management Systems (DBMSs)
Data Models in the 1960s, 1970s, and 1980s

Hierarchical

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1956</td>
<td>Vern Watts was IMS's chief architect for many years. Watts joined IBM in 1956 and worked at IBM's Silicon Valley development labs until his death on April 4, 2009.</td>
</tr>
</tbody>
</table>

Network (Graph)

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1966</td>
<td>IBM began the designed of IMS with Rockwell and Caterpillar for the Apollo program, where it was used to inventory the very large bill of materials (BOM) for the Saturn V moon rocket and Apollo space vehicle.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1968</td>
<td>The first &quot;IMS READY&quot; message appeared on an IBM 2740 terminal in Downey, California, on 14 August 1968.</td>
</tr>
</tbody>
</table>

Relational – Schema (Model) - first

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1965</td>
<td>CODASYL (Conference / Committee on Data Systems Languages) formed a List Processing Task Force.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1969</td>
<td>the network CODASYL database model was published.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980s</td>
<td>interest in CODASYL gradually faded due to growing interest in relational databases.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>Dr. E. F. Codd develops the relational database model.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1978</td>
<td>IBM develops the predecessor to SQL.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979</td>
<td>Relational Software, Inc. (later renamed Oracle) releases the first relational DBMS, Oracle.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1982</td>
<td>IBM releases their first RDBMS, SQL/DS (SQL/Data System).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985</td>
<td>IBM released DB2 (Database 2).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>Microsoft releases SQL Server.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1989</td>
<td>ANSI publishes first SQL standards (ANSI/ISO SQL-89, or SQL1).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
</table>
Hierarchical DBMS Example
(A Contrived Hierarchical DBMS using HTML)

Dr. Philip Cannata

Queries

Show all employees:
$('*emp')$

Show employee with id=7876:
$('*#7876')$

Show all departments:
$('*dept')$

Show department of employee with id=7876:
$('*#7876').parent().andSelf()$

Show department of employees:
$('*emp').parent().andSelf()$

Show all employees with their department name:
var r=$('*emp').each(function( index, element ) {console.log( $(this).attr('empno'), $(this).attr('ename'), $(this).attr('job'), $(this).attr('mgr'), $(this).attr('hiredate'), $(this).attr('sal'), $(this).attr('comm'), $(this).parent().attr('dname') });$

Show all employees with their manager’s name:
Hierarchical DBMS Example
(A Contrived Hierarchical DBMS using HTML)

### Queries

Show all employees with their manager’s name:

```javascript
var a = $('emp').each(function() {
  try {
    console.log( "EMPLOYEE: " + $(this).attr('ename') + " MANAGER " + $("#" + $(this).attr("mgr")).attr('ename') );
  } catch(e) {} });
```
Hierarchical DBMS Example

(A Second Contrived Hierarchical DBMS using HTML)

<body>

<ol>
  <li>Show all managers: $(‘mgr’)</li>
  <li>Show all employees with their manager’s name:
      var r=$("emp").each(function(index, element){console.log($(this).attr('empno'), $(this).attr('ename'), $(this).attr('job'), $(this).attr('mgr'), $(this).attr('hiredate'), $(this).attr('sal'), $(this).attr('comm'), $(this).parent().parent().attr('ename')})</li>
</ol>

Dr. Philip Cannata
Dr. E. F. Codd

1. Humans only know how to store data on Disk or in Memory in the form of Mathematical Relations (Tables).
2. So, the Data Model should be Relational (i.e., Tables).
3. Also, DBMSs should have an ad hoc query language.

A mathematical Relation is a subset of the cross-product of a set of Domains. e.g.,

Example 1

| < | 0 | 1 |
|   | 0 | 2 |
|   | 0 | 3 |
|   | 1 | 2 |
|   | 1 | 3 |
|   | 2 | 3 |

Example 2

<table>
<thead>
<tr>
<th>D1: People</th>
<th>D2: Ages</th>
<th>D3: Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phil</td>
<td>60</td>
<td>M</td>
</tr>
<tr>
<td>Phil</td>
<td>60</td>
<td>F</td>
</tr>
<tr>
<td>Phil</td>
<td>65</td>
<td>M</td>
</tr>
<tr>
<td>Phil</td>
<td>65</td>
<td>F</td>
</tr>
<tr>
<td>Rita</td>
<td>60</td>
<td>M</td>
</tr>
<tr>
<td>Rita</td>
<td>60</td>
<td>F</td>
</tr>
<tr>
<td>Rita</td>
<td>65</td>
<td>M</td>
</tr>
<tr>
<td>Rita</td>
<td>65</td>
<td>F</td>
</tr>
</tbody>
</table>
Dr. Philip Cannata

Traditional Data Management in the 1980s, 1990s, and 2000s

Hierarchical

Network (Graph)

Relational – Schema (Model) - first

1970 Dr. E. F. Codd develops the relational database model.
1978 IBM develops the predecessor to SQL.
1979 Relational Software, Inc. (later renamed Oracle) releases the first relational DBMS, Oracle.
1982 IBM releases their first RDBMS, SQL/DS (SQL/Data System).
1985 IBM released DB2 (Database 2).
1987 Microsoft releases SQL Server.
1989 ANSI publishes first SQL standards (ANSI/ISO SQL-89, or SQL1).

1956 Vern Watts was IMS's chief architect for many years. Watts joined IBM in 1956 and worked at IBM's Silicon Valley development labs until his death on April 4, 2009.

1965 CODASYL (Conference / Committee on Data Systems Languages) formed a List Processing Task Force.
1969 The network CODASYL database model was published.

1980s interest in CODASYL gradually faded due to growing interest in relational databases.

1966 IBM began the designed of IMS with Rockwell and Caterpillar for the Apollo program. It was used to inventory the very large bill of materials (BOM) for the Saturn V moon rocket and Apollo space vehicle.

1968 The first "IMS READY" message appeared on an IBM 2740 terminal in Downey, California, on 14 August 1968.

1969 The network CODASYL database model was published.

1982 IBM releases its first RDBMS, SQL/DS (SQL/Data System).
1985 IBM released DB2 (Database 2).
1987 Microsoft releases SQL Server.
1989 ANSI publishes first SQL standards (ANSI/ISO SQL-89, or SQL1).

1956 Vern Watts was IMS's chief architect for many years. Watts joined IBM in 1956 and worked at IBM's Silicon Valley development labs until his death on April 4, 2009.

1966 IBM began the designed of IMS with Rockwell and Caterpillar for the Apollo program. It was used to inventory the very large bill of materials (BOM) for the Saturn V moon rocket and Apollo space vehicle.

1968 The first "IMS READY" message appeared on an IBM 2740 terminal in Downey, California, on 14 August 1968.

1965 CODASYL (Conference / Committee on Data Systems Languages) formed a List Processing Task Force.
1969 The network CODASYL database model was published.

1980s interest in CODASYL gradually faded due to growing interest in relational databases.
There were many other Data Models, but most were not implemented as DBMSs.

**ER Data Model is still used in Data Modeling**

Introduced by Peter Chen in 1976 - "The Entity Relationship Model: Toward a Unified View of Data" for entity–relationship modeling.”

---

**Diagram:**

- **One-to-Many (1:M) Relationship:**
  - A PAINTER can paint many PAINTINGS; each PAINTING is painted by one PAINTER.
  - **Diagram:**
    - PAINTER (1)
    - PAINTING (M)
    - PAINTER paints PAINTING

- **Many-to-Many (M:N) Relationship:**
  - An EMPLOYEE can learn many SKILLS; each SKILL can be learned by many EMPLOYEES.
  - **Diagram:**
    - EMPLOYEE (M)
    - SKILL (N)
    - EMPLOYEE learns SKILL

- **One-to-One (1:1) Relationship:**
  - An EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.
  - **Diagram:**
    - EMPLOYEE (1)
    - STORE (1)
    - EMPLOYEE manages STORE
There were many other Data Models but, most were not implemented as DBMSs. **IDEF1X Data Model is still used in Data Modeling**
There were many other Data Models, but most were not implemented as DBMSs. 
IE Data Model is still used in Data Modeling
There were many other Data Models, but most were not implemented as DBMSs.
UML Data Model is still used in Data Modeling
There were many other Data Models, but most were not implemented as DBMSs. **OO DBMSs were implemented but failed**
There were many other Data Models, but most were not implemented as DBMSs. LDAP was implemented and still exists today as Directory Servers (e.g., Active Directory)
There were many other Data Models, but most were not implemented as DBMSs. EAV was implemented
EAV Evolved into RDF and other NoSQL DBMSs

Resource Description Framework (RDF)

Contents: Specifications | Bookmarks (Intro * Articles) | Projects and Applications | Developer tools | Schemas | Related Technologies | Timeline

Nearby: Semantic Web Advanced Development | RDF Validator | Resource Guide | Scratchpad

The Resource Description Framework (RDF) integrates a variety of applications from library catalogs and world-wide directories to syndication and aggregation of news, software, and content to personal collections of music, photos, and events using XML as an interchange syntax. The RDF specifications provide a lightweight ontology system to support the exchange of knowledge on the Web.

The W3C Semantic Web Activity Statement explains W3C's plans for RDF, including the RDF Core WG, Web Ontology and the RDF Interest Group.

RDF Specification Development

The RDF Specifications build on URI and XML technologies. The RDF suite of specifications consist of:

- RDF/XML Syntax Specification (Revised)
  W3C Recommendation
  Dave Beckett, ed.
- RDF Vocabulary Description Language 1.0: RDF Schema
  W3C Recommendation
  Dan Brickley, R.V. Guha, eds.
- RDF Primer
Data Management Today

NoSQL – Schema(Model)-later

- Couchbase
- Neo4j
- InfiniteGraph
- MongoDB
- MarkLogic
- Key-Value Databases
- Redis
- Hypertable
- Riak
- SimpleDB
- Aerospike
- Oracle Graph RDF
- W3C

Wide Column Stores

- Hadoop
- Cassandra
- Amazon DynamoDB
- Apache HBase

Graph Databases

- Neo4j
- InfiniteGraph

Document Databases

- MongoDB
- Couchbase

Relational (YeSQL) – Schema(Model)-first

- Oracle 12c
- IBM DB2
- MySQL
- SQL Server

Big Data – Schema(Model)-never

- 40 Zettabytes of 6,600,000,000,000,000,000,000,000,000,000,000 bytes of data is created each day
- 9 billion people have cell phones
- What company is the 2.5 Quintillion bytes of data created each day
- If 100 Terabytes of data filled 333 homes
- The New York Stock Exchange trades 1 TB of data information during each trading session
- By 2016, it is projected there will be 18.3 billion network connections – almost 3.5 connections per person in earth

The FOUR V's of Big Data

- Volume
- Variety
- Velocity
- Veracity

- 4.4 million IT jobs
- 27% of respondents on how much of their data was accurate

- 30 billion pieces of content are shared on Facebook every month
- 600 million tweets are sent per day by about 300 million active users
- 4 billion hours of video are watched on YouTube each month
- 1 in 3 business leaders don’t trust the information they use to make decisions
- Pure data quality costs the US economy around $3.1 trillion a year
- As of 2011, the global size of data in healthcare was estimated to be 130 exabytes (100,000,000,000,000,000,000 bytes). By 2020, it is expected that 2.5 quintillion bytes of data will be created and stored each day.

Dr. Philip Cannata
Data Management Today

NoSQL – Schema(Model)-later

Big Data – Schema(Model)-never

https://training.edb.utexas.edu/node/2329

http://www.lynda.com/NoSQL-tutorials/NoSQL-SQL-Professionals/368756-2.html?
srchtrk=index:1%Alinktypeid:2%Aq:nosql%Apage:1%As:relevance%Asa:true%Aproducttypeid:2

I don’t endorse all of the statements in this course but it does represent what many people are preaching.
# Example Comparison: RDBMS vs. Hadoop

<table>
<thead>
<tr>
<th></th>
<th>Traditional RDBMS</th>
<th>Hadoop / MapReduce</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Size</strong></td>
<td>Gigabytes (Terabytes)</td>
<td>Petabytes and greater</td>
</tr>
<tr>
<td><strong>Access</strong></td>
<td>Interactive and Batch</td>
<td>Batch – NOT Interactive</td>
</tr>
<tr>
<td><strong>Updates</strong></td>
<td>Read / Write Many Times</td>
<td>Write Once, Read Many Times</td>
</tr>
<tr>
<td><strong>Structure</strong></td>
<td>Static Schema</td>
<td>Dynamic Schema</td>
</tr>
<tr>
<td><strong>Integrity</strong></td>
<td>High (ACID)</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Scaling</strong></td>
<td>Nonlinear</td>
<td>Linear</td>
</tr>
<tr>
<td><strong>Query Response Time</strong></td>
<td>Can Be Near Immediate</td>
<td>Has Latency (Due to Batch Processing)</td>
</tr>
</tbody>
</table>
cs347 Data Management

NoSQL – Schema(Model)-later

Big Data – Schema(Model)-never

Relational (YeSQL) – Schema-first

The Complexity goes Somewhere!
DataStax – Graph Databases

TechRadar: Database Management 2014

Demo of Neo4j Cypher Graph Database Language

I’ll be using the Apex 5 REST interface for this demo.
Demo of Neo4j Cypher Graph Database Language

The Apex 5 REST interface is similar to POSTMAN.
Demo of Neo4j Cypher Graph Database Language


Notice: I didn’t create an emp “table” before this insert.
Demo of Neo4j Cypher Graph Database Language

"CREATE (:dept { DEPTNO : 20, DNAME : 'RESEARCH', LOC : 'DALLAS' })"
"MATCH (a:emp),(b:dept) WHERE a.deptno = 20 AND b.deptno = 20 CREATE (a)-[:employees]->(b)"

Creating a Relationship.
Demo of Neo4j Cypher Graph Database Language

Query: "MATCH(a:emp) RETURN a"

Results:

Clob001

"MATCH(a:emp) RETURN a"
"MATCH(a:emp)<-[employees]-(b:dept) WHERE b.deptno = 20 RETURN b, a.ename, a.job, a.mgr, a.deptno, a.hiredate"
I’ll load this behind the scenes and then do a few more examples.
SIM DDL for the emp/dept Database


CREATE (:dept { DEPTNO : 10, DNAME : 'ACCOUNTING', LOC : 'NEW YORK'})
CREATE (:dept { DEPTNO : 20, DNAME : 'RESEARCH', LOC : 'DALLAS'})
CREATE (:dept { DEPTNO : 30, DNAME : 'SALES', LOC : 'CHICAGO'})
CREATE (:dept { DEPTNO : 40, DNAME : 'OPERATIONS', LOC : 'BOSTON'})

MATCH (a:emp),(b:dept) WHERE a.deptno = 10 AND b.deptno = 10 CREATE (a)<-[[:employees]]-(b)
MATCH (a:emp),(b:dept) WHERE a.deptno = 20 AND b.deptno = 20 CREATE (a)<-[[:employees]]-(b)
MATCH (a:emp),(b:dept) WHERE a.deptno = 30 AND b.deptno = 30 CREATE (a)<-[[:employees]]-(b)
MATCH (a:emp),(b:dept) WHERE a.deptno = 40 AND b.deptno = 40 CREATE (a)<-[[:employees]]-(b)
MATCH (a:emp)<-[[:employees]]-(b:dept) WHERE b.deptno = 20 RETURN b, a.name, a.job, a.mgr, a.deptno, a.hiredate
MATCH (a:emp),(b:dept) WHERE a.deptno = 10 AND b.deptno = 10 CREATE (a)-[:dept]->(b)
MATCH (a:emp),(b:dept) WHERE a.deptno = 20 AND b.deptno = 20 CREATE (a)-[:dept]->(b)
MATCH (a:emp),(b:dept) WHERE a.deptno = 30 AND b.deptno = 30 CREATE (a)-[:dept]->(b)
MATCH (a:emp),(b:dept) WHERE a.deptno = 40 AND b.deptno = 40 CREATE (a)-[:dept]->(b)
MATCH (a:emp)-[:dept]->(b:dept) RETURN b.dname, a.name

Creating Relationships.
Demo of Neo4j Cypher Graph Database Language

"MATCH(a:emp)-[:dept]->(b:dept) RETURN b.dname, a.ename"
Demo of Neo4j Cypher Graph Database Language

```
MATCH(a:emp)<-[employees]-(b:dept) WHERE b.deptno = 20 RETURN b, a.ename, a.job, a.mgr, a.deptno, a.hiredate
```

Dr. Philip Cannata
cs347 Data Management

NoSQL – Schema(Model)-later

Big Data – Schema(Model)-never

The Complexity goes Somewhere!
All Data Models are Ignoring the First Data Model
Except for SIM, which was implemented in the 1980s and still exists at UT with Me

Based on the 1981 paper “Database Description with SDM: A Semantic Database Model” by Michael Hammer and Dennis McLeod, see class calendar for a pdf version of the paper.
Data Management

Dr. Philip Cannata, phil.cannata@oracle.com, office hours: MW 3:00 - 3:45 in GDC 5.402.
TA: Chia-Chen Hsu, cchsu@utexas.edu, office hours: TBD.
Proctor: Chris Timeus, timaeus@utexas.edu, office hours: TBD.

Canvas link.

We will be using Piazza.

Make sure you are registered with piazza for this course and be sure to check and read your piazza email several times a day because this will be the primary means of communication outside of class time. Dr. Cannata will not alter his procedure of always sending all of his messages to the entire class.

Students with disabilities link.

Students with disabilities may request appropriate academic accommodations from the Division of Diversity and Community Engagement, Services for Students with Disabilities, 512-471-6259, and the link above.

Schema-first (i.e., a Relational Model is required) DBMS

Oracle emp/dept (scott/tiger) Relational database:

Cut and paste from this link into SQLDeveloper to create the emp and dept tables in your Oracle user account.

Schema-later DBMSs

Contrived Hierarchical emp/dept HTML databases:

Load these into Chrome and then turn on the JavaScript Console (option-command-j on the Mac) to view the Elements structure.

- HTML DBMS #1
- HTML DBMS #2

emp/dept Neo4j database:

The emp/dept neo4j DDL can be viewed at this link. You can cut and paste the sql from this link into SQLDeveloper to see how I generated the neo4j emp/dept DDL from the Relational emp/dept database.

emp/dept SIM database:

The emp/dept SIM DDL can be viewed at this link.
SIM DDL for the emp/dept Database

Creating Relationships.
Revealing a Secret

Query: "MATCH(a:emp) RETURN a"

Db: jdbc:oracle:thin:@129.152.144.84:1521/PDB1.usuniversi01134.oraclecloud.internal

Mode: rdf_mode

Returndimensions: False

Returnfor: neo4j_python

Model: F2015

Results:

Clob001

["EMPNO", "DEPTNO", "COMM", "JOB", "MGR", "ENAME", "HIREDATE", "SAL"], [7900, 30, 0, "CLERK", 7698, "JAMES", "03-DEC-81", 950], [7698, 30, 0, "MANAGER", 7839, "JONES", "02-APR-81", 2975], [7782, "MILLER", "23-JAN-82", 1300], [7876, 20, 0, "CLERK", 7788, "ADAMS", "12-JAN-83", 1100], [7788, 20, 0, "ANALYST", 7566, "SCOTT", "09-DEC-82", 3000], [7654, 30, 1400, ""], [..]
ReL is a research system that I’ve built with the help of my students. ReL is a fork of Jython, which is a python interpreter written in Java that incorporates various Data Models. In this case, ReL converts Neo4j/Cypher into SIM and then converts SIM to SPARQL. ReL is also being run as my RESTful Server.

{"query": u"MATCH(a:emp) RETURN a"}

**ReL statement is:** MATCH(a:emp) RETURN a

**SIM is:** FROM EMP RETRIEVE *;

**SPARQL is:** SELECT DISTINCT empno, deptno, comm, job, mgr, ename, hiredate, sal
from table(
    sem_match('select * where {
        GRAPH <emp_SCHEMA> { ?indiv rdf:type :emp }
    }',
    SEM_MODELS('F2015_CS370_SIM'), null,
    SEM_ALIASES(SEM_ALIAS('', '#'), null)
    ('EMPNO', 'DEPTNO', 'COMM', 'JOB', 'MGR', 'ENAME', 'HIREDATE', 'SAL'), (7900, 30, 0, 'CLERK', 7698, 'JAMES', '03-DEC-81', 950), (7698, 30, 0, 'MANAGER', 7839, 'BLAKE', '01-MAY-81', 2850)) . . .
Revealing a Secret

When I loaded the full Cypher, this was built under the covers by ReL.
"MATCH(a:emp)<-[:employees]-(b:dept) WHERE b.deptno = 20 RETURN b, a.ename, a.job, a.mgr, a.deptno, a.hiredate"
Revealing a Secret

{"query": u"MATCH(a:emp)<-[[:employees]]-(b:dept) WHERE b.deptno = 20 RETURN b, a.ename, a.job, a.mgr, a.deptno, a.hiredate"}

**ReL statement is:** MATCH(a:emp)<-[[:employees]]-(b:dept) WHERE b.deptno = 20 RETURN b, a.ename, a.job, a.mgr, a.deptno, a.hiredate

**SIM is:** FROM DEPT RETRIEVE *, ENAME OF employees, JOB OF employees, MGR OF employees, DEPTNO OF employees, HIREDATE OF employees WHERE DEPTNO = 20;

**SPARQL is:** SELECT DISTINCT deptno, dname, loc, x0_1, x1_1, x2_1, x3_1, x4_1
from table(
  sem_match('select * where {
    GRAPH <dept_SCHEMA> { ?indiv rdf:type :dept }

    OPTIONAL {
      ?indiv :employees ?x0_0 .
      ?x0_0 :ename ?x0_1 .
      ?indiv :employees ?x0_0 .
      ?x0_0 :job ?x1_1 .
      ?indiv :employees ?x0_0 .
      ?x0_0 :mgr ?x2_1 .
      ?indiv :employees ?x0_0 .
      ?x0_0 :deptno ?x3_1 .
      ?indiv :employees ?x0_0 .
      ?x0_0 :hiredate ?x4_1 .
    }
  }
},
SEM_MODELS('F2015_CS370_SIM'), null,
SEM_ALIASES( SEM_ALIAS('', '#')), null )

(('DEPTNO', 'DNAME', 'LOC', 'X0_1', 'X1_1', 'X2_1', 'X3_1', 'X4_1'), (20, 'RESEARCH', 'DALLAS', 'ADAMS', 'CLERK', 7788, 20, '12-JAN-83'), (20, 'RESEARCH', 'DALLAS', 'SCOTT', 'ANALYST', 7566, 20, '09-DEC-82')) ...
How is the Neo4j/Cypher to SIM translation done in ReL?

MATCH (e:emp)-[:DEPTNO]->(d:dept)-[:ORGNUM]->(o:org) WHERE e.empno = 111
RETURN e.empno, e.ename, e.sal, d.dname, o.oname

In CS345, you learn to parse this kind of statement and build an Abstract Syntax Tree (AST):

![Diagram of an Abstract Syntax Tree]

You use a set of formal grammars to do this.

You also learn how to traverse the AST and output something like SIM (I teach how to do this using a “Visitor” pattern).

This work was done by Carson Ball as a CS370 project this past Summer. Carson has had cs345, cs375, and cs347. I would hire him!
Revealing another Secret

Query: "select * from emp"

Db: jdbc:oracle:thin:@129.152.144.84:1521/PDB1.usuniversi01134.oraclecloud.internal

Mode: rdf_mode

Returndimensions: False

Returnfor: python

Model: F2015

Results:

Others are Starting to Discover this Secret

DataStax – Graph Databases
http://www.datastax.com/2015/03/qa-cassandra-and-titandb-insights-into-datastaxs-graph-strategy - minute 29:00 to the end
But there’s More that Others are Not Discovering Yet

Here’s some RDF/RDFa/OWL magic to do class/subclass inferencing (entailment).

CREATE (:customer { CUSTNO : 1111, CNAME : 'PHIL'});

INSERT INTO F2015_CS370_SIM DATA VALUES (1000, SDO_RDF_TRIPLE_S('F2015_CS370_SIM:<SCHEMA>', '#person', 'rdf:type', 'rdfs:Class'));

BEGIN
  sem_apis.create_entailment(
    'emp_dept_entailment',
    sem_models('F2015_CS370_SIM'),
    sem_rulebases('owl2r1'),
    SEM_APIES.REACH_CLOSURE
  );
END;
/

BEGIN
  -- NGDL
  sem_apis.create_entailment(
    'emp_dept_entailment',
    sem_models('F2015_CS370_SIM'),
    sem_rulebases('owl2r1'),
    SEM_APIES.REACH_CLOSURE,
    options => 'LOCAL_NG_INF=T,ENTAIL_ANYWAY=T'
  );
END;
/

SELECT v1 "empno", v3 "comm", v4 "job", v5 "mgr", v6 "ename", v7 "hiredate", v8 "sal", v9 custno, v10 cname
FROM TABLE(SEM_MATCH('SELECT . WHERE {
  GRAPH ?g { ?s1 rdf:type :person }
  OPTIONAL { ?s1 :empno ?v1 }
  OPTIONAL { ?s1 :deptno ?v2 }
  OPTIONAL { ?s1 :comm ?v3 }
  OPTIONAL { ?s1 :job ?v4 }
  OPTIONAL { ?s1 :mgr ?v5 }
  OPTIONAL { ?s1 :ename ?v6 }
  OPTIONAL { ?s1 :hiredate ?v7 }
  OPTIONAL { ?s1 :sal ?v8 }
  OPTIONAL { ?s1 :custno ?v0 }
  OPTIONAL { ?s1 :cname ?v10 }
  }
  ,
  SEM_MODELS('F2015_CS370_SIM'), SDO_RDF_Rulebases('owl2r1'),
  SEM_ALIASSES( SEM_ALIAS('', '#'), null )
);
class/subclass Inferencing is just the beginning

For more information on Inferencing, see Chapter 9 of the Learning SPARQL book (see the Books and Papers Tab on the class website).

Dr. Philip Cannata
cs347 Data Management

NoSQL – Schema(Model)-later

Big Data – Schema(Model)-never

The Complexity goes Somewhere!
Useful Free Books

See the Books and Papers Tab on the class website.