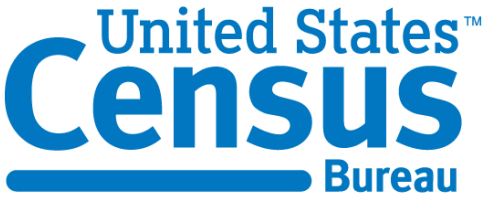
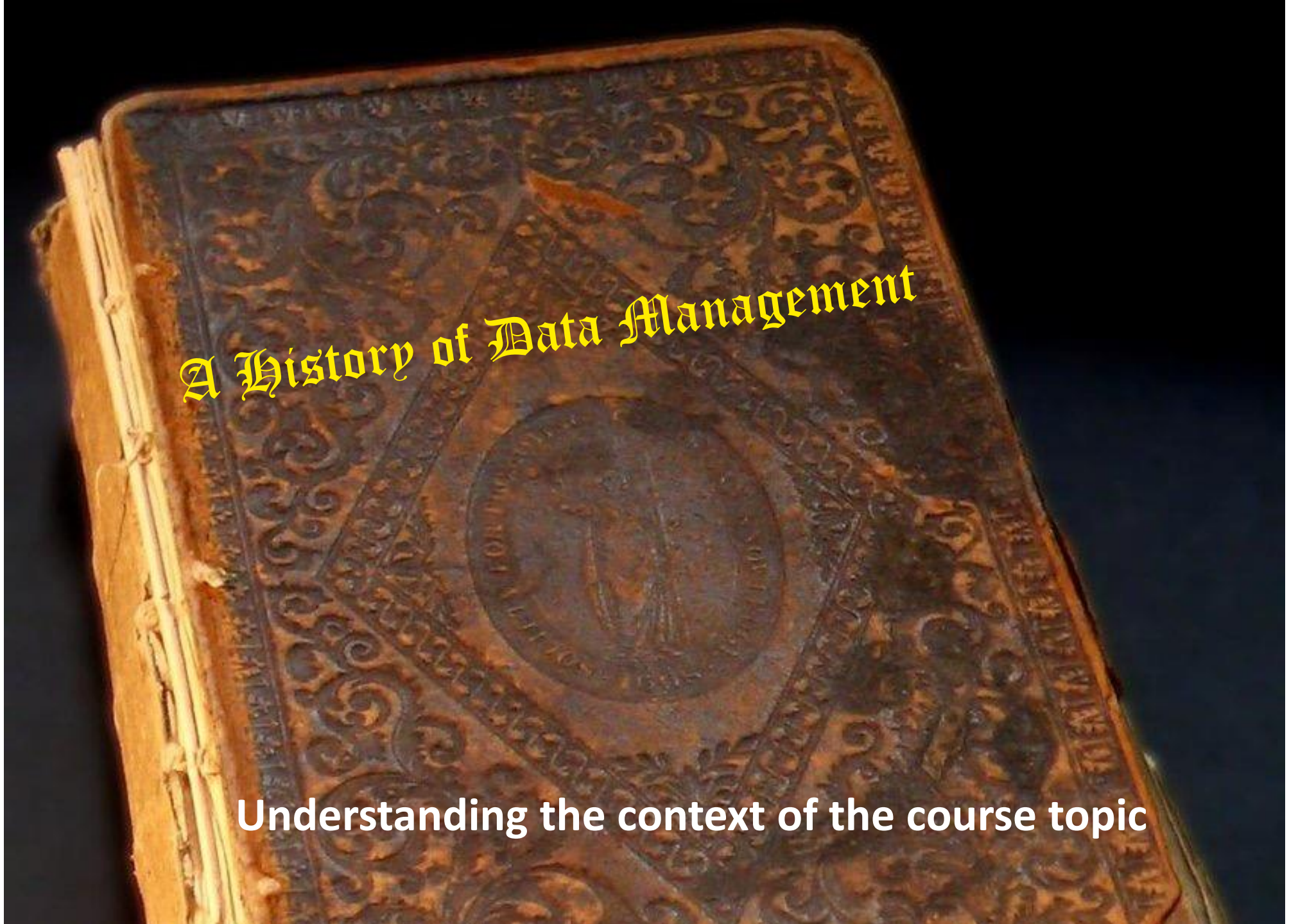
COMP 430/533
Intro. to Database Systems

Course overview

Databases central to modern life





Understanding the context of the course topic

Ancient History: 1950’s-1970’s

Computing in the 1950's

An exciting time:

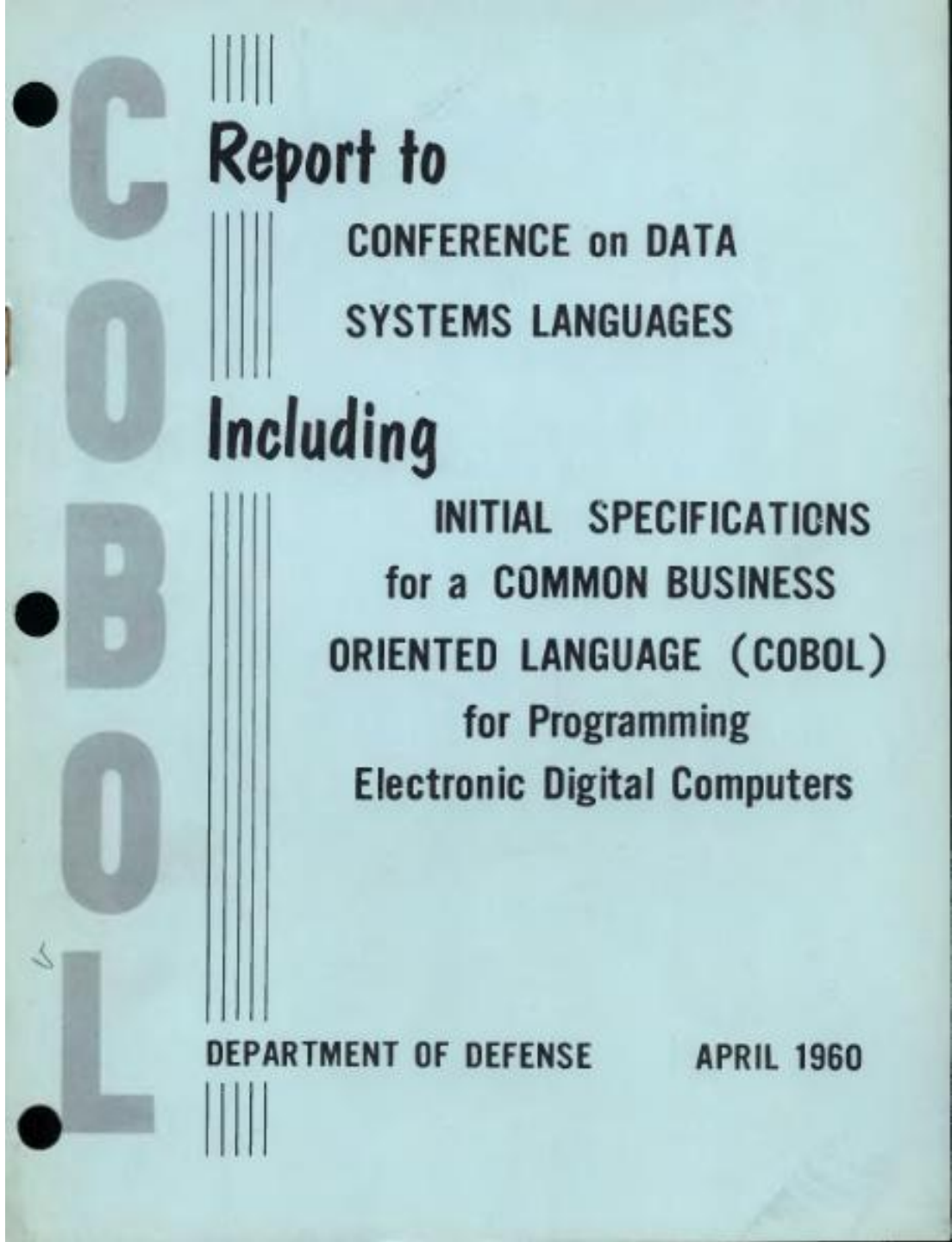
- FORTRAN – John Backus, 1957
- LISP – John McCarthy, invented 1958, implemented 1962

But hardware limited:

- Data management was almost inconceivable.

CODASYL (1959)

First **CO**nference on **DA**ta **SY**stems and **L**anguages



COBOL Sample (Part 1)

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. SAMPLE.
000030 AUTHOR. J.P.E. HODGSON.
000040 DATE-WRITTEN. 4 February 2000
000041
000042* A sample program just to show the form.
000043* The program copies its input to the output,
000044* and counts the number of records.
000045* At the end this number is printed.
000046
000050 ENVIRONMENT DIVISION.
000060 INPUT-OUTPUT SECTION.
000070 FILE-CONTROL.
000080     SELECT STUDENT-FILE ASSIGN TO SYSIN
000090         ORGANIZATION IS LINE SEQUENTIAL.
000100     SELECT PRINT-FILE ASSIGN TO SYSOUT
000110         ORGANIZATION IS LINE SEQUENTIAL.
000120
000130 DATA DIVISION.
000140 FILE SECTION.
000150 FD   STUDENT-FILE
000160     RECORD CONTAINS 43 CHARACTERS
000170     DATA RECORD IS STUDENT-IN.
000180 01   STUDENT-IN                PIC X(43).
000190
000200 FD   PRINT-FILE
000210     RECORD CONTAINS 80 CHARACTERS
000220     DATA RECORD IS PRINT-LINE.
000230 01   PRINT-LINE                PIC X(80).
000240
```

```
000250 WORKING-STORAGE SECTION.
000260 01  DATA-REMAINS-SWITCH      PIC X(2)
000261                 VALUE SPACES.
000262 01  RECORDS-WRITTEN          PIC 99.
000270
000280 01  DETAIL-LINE.
000290     05 FILLER                PIC X(7)
000291                 VALUE SPACES.
000300     05 RECORD-IMAGE          PIC X(43).
000310     05 FILLER                PIC X(30)
000311                 VALUE SPACES.
000312
000313 01 SUMMARY-LINE.
000314     05 FILLER                PIC X(7)
000315                 VALUE SPACES.
000316     05 TOTAL-READ            PIC 99.
000317     05 FILLER                PIC X
000318                 VALUE SPACE.
000319     05 FILLER                PIC X(17)
000320                 VALUE 'Records were read'.
000321     05 FILLER                PIC X(53)
000322                 VALUE SPACES.
000323
```

Data arranged in records.
Data described declaratively, separate from “code”.

COBOL Sample (Part 2)

```
000330 PROCEDURE DIVISION.
000331
000340 PREPARE-SENIOR-REPORT.
000350     OPEN INPUT STUDENT-FILE
000360          OUTPUT PRINT-FILE.
000361     MOVE ZERO TO RECORDS-WRITTEN.
000370     READ STUDENT-FILE
000380         AT END MOVE 'NO' TO DATA-REMAINS-SWITCH
000390     END-READ.
000400     PERFORM PROCESS-RECORDS
000410         UNTIL DATA-REMAINS-SWITCH = 'NO'.
000411     PERFORM PRINT-SUMMARY.
000420     CLOSE STUDENT-FILE
000430         PRINT-FILE.
000440     STOP RUN.
000450
000460 PROCESS-RECORDS.
000470     MOVE STUDENT-IN TO RECORD-IMAGE.
000480     MOVE DETAIL-LINE TO PRINT-LINE.
000490     WRITE PRINT-LINE.
000500     ADD 1 TO RECORDS-WRITTEN.
000510     READ STUDENT-FILE
000520         AT END MOVE 'NO' TO DATA-REMAINS-SWITCH
000530     END-READ.
000540
000550 PRINT-SUMMARY.
000560     MOVE RECORDS-WRITTEN TO TOTAL-READ.
000570     MOVE SUMMARY-LINE TO PRINT-LINE.
000571     WRITE PRINT-LINE.
000572
```

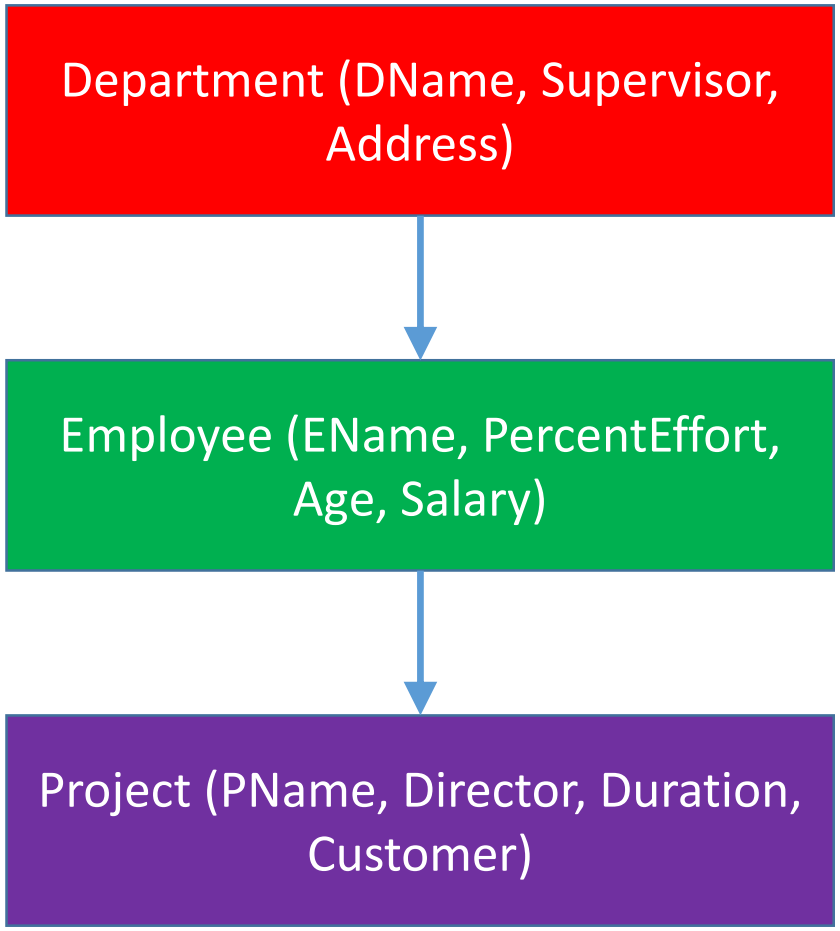
Code processes a record at a time.
English-like syntax.
Keywords capitalized.

Information Management System (1966)

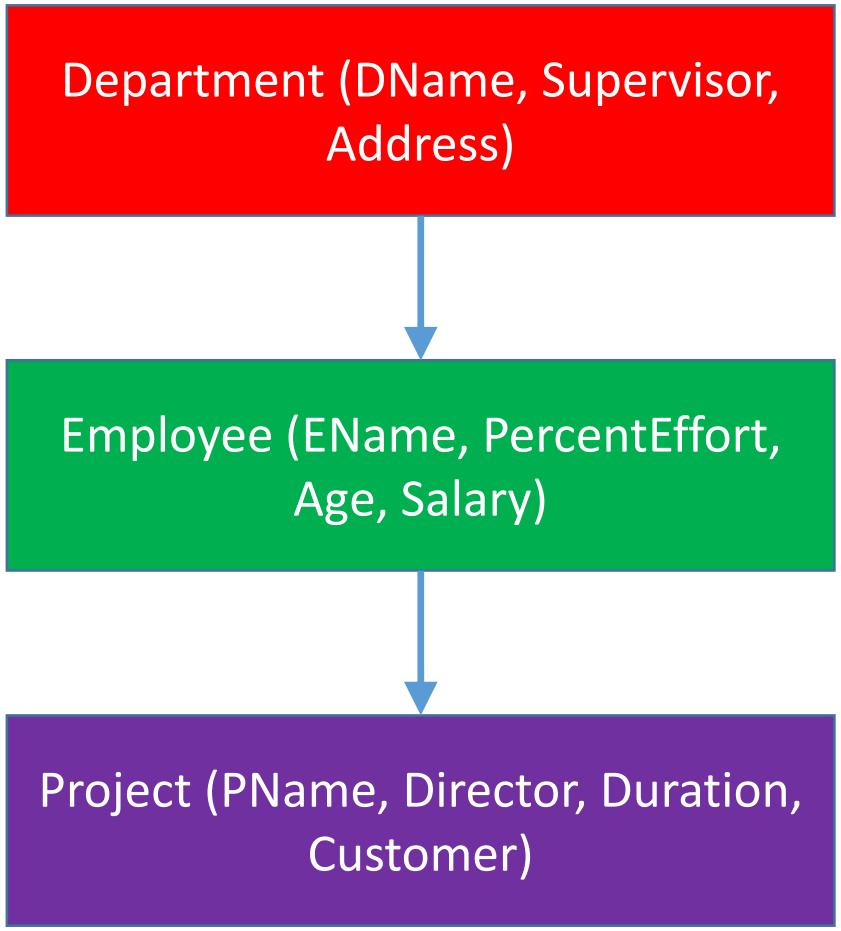
Developed by IBM for Saturn V

Used first serious data model – hierarchical

Hierarchical Data Model



Hierarchical Data Model

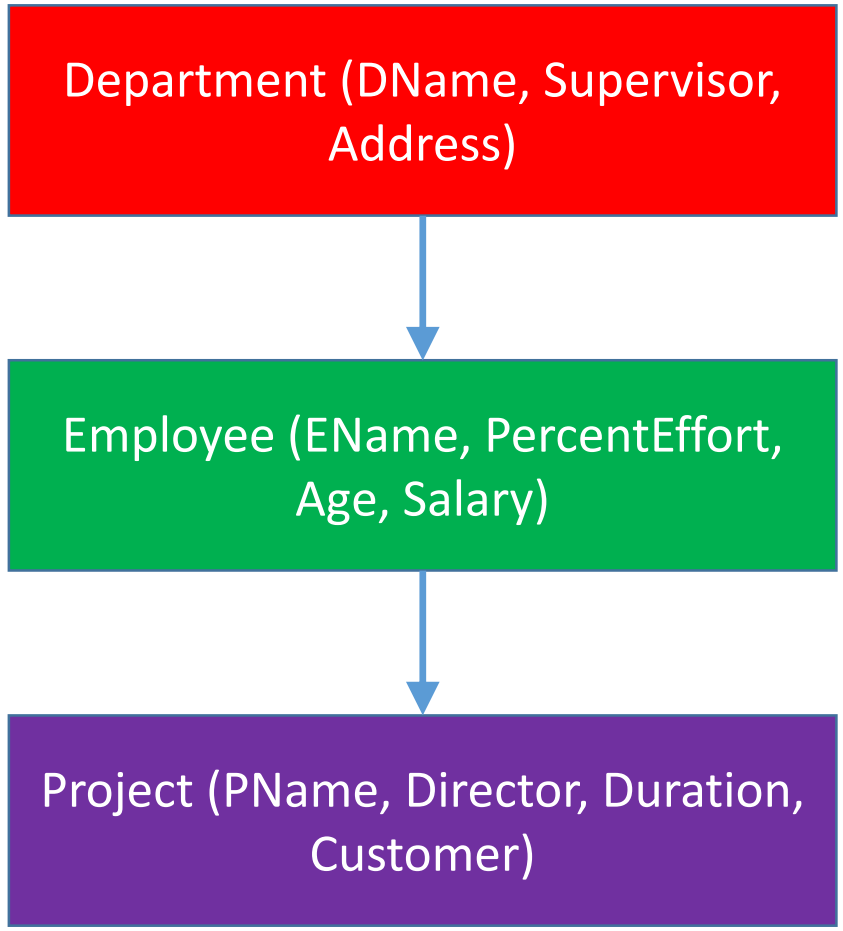


Which employees supervised by Swarat work on Pliny?

DL/1 code:

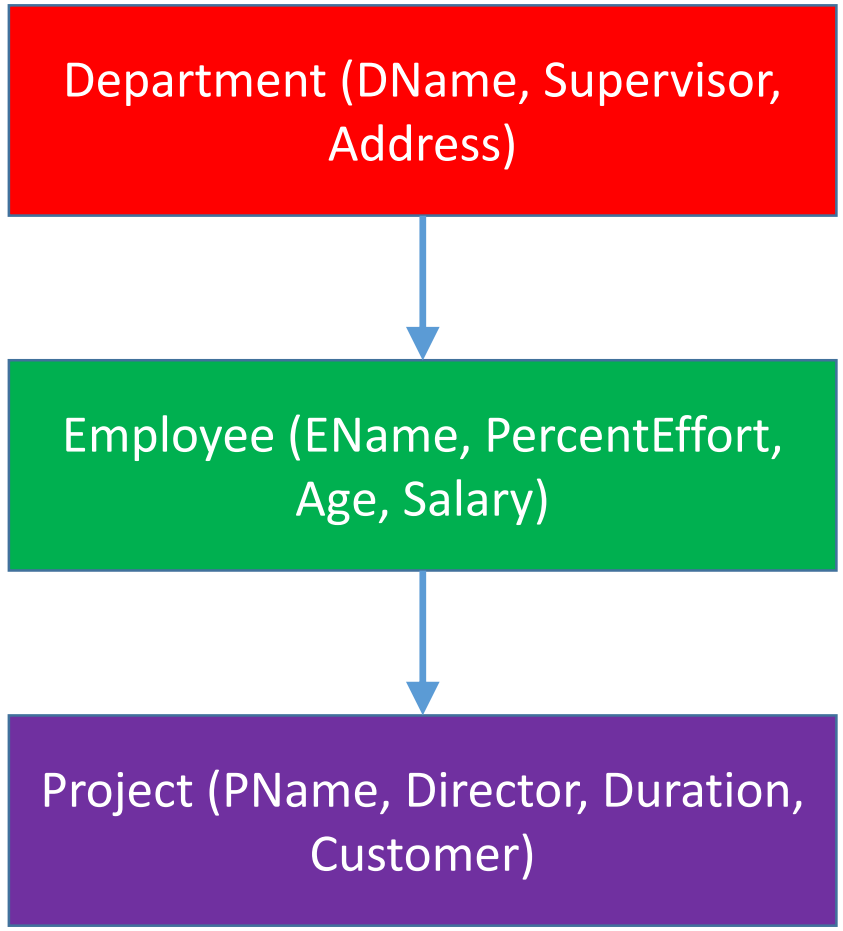
```
Get unique Department (Supervisor = "Swarat")
  Until failure do
    Get next within parent
      Until failure do
        Get next within parent (PName = "Pliny")
        ...
      Enddo
  Enddo
```

Hierarchical Data Model – directional



Can search from Departments to Projects, but not the other way.

Hierarchical Data Model – redundancy

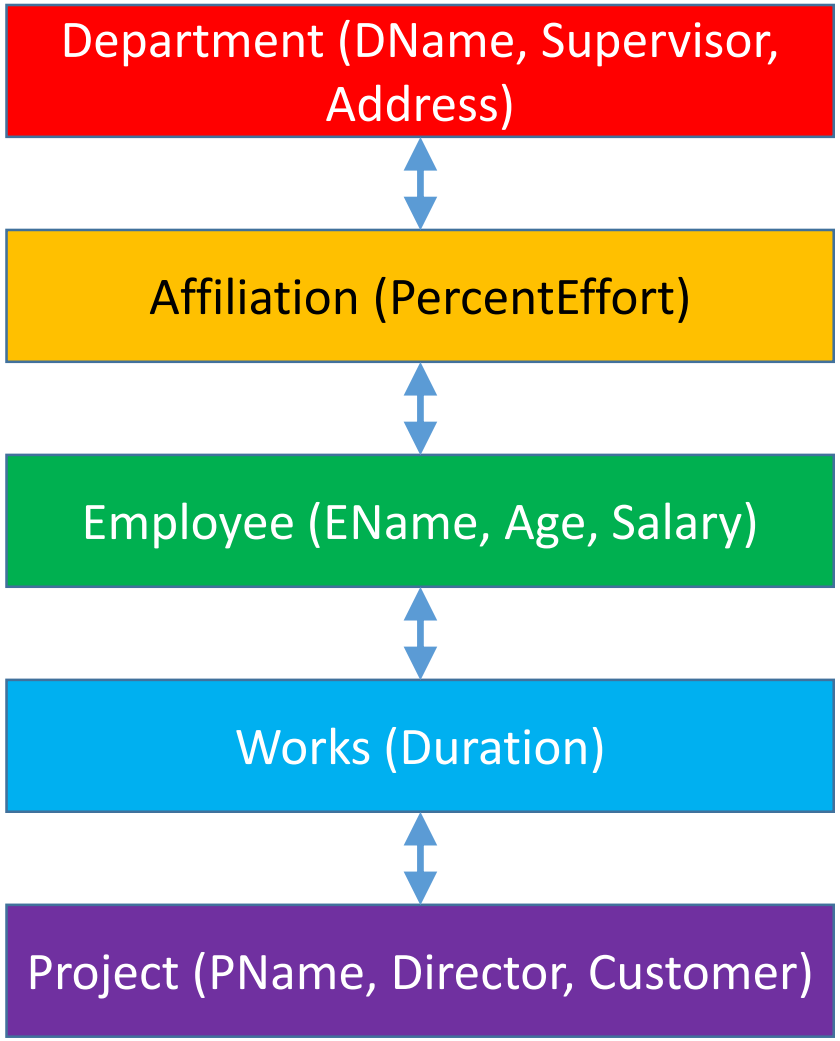


Assume:

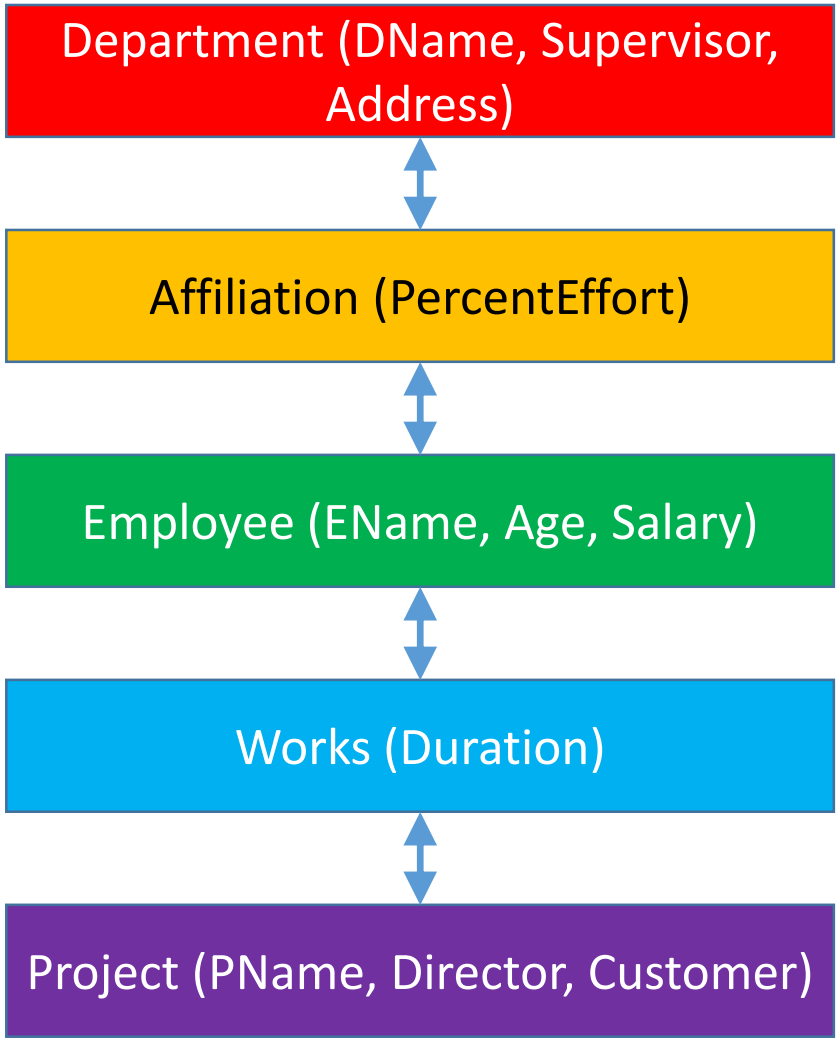
- Employees can work on multiple projects.
- Projects can have multiple employees.

Leads to redundancy – Project info repeated for each of its employees.

CODASYL Network Data Model (1969)



CODASYL Network Data Model (1969)

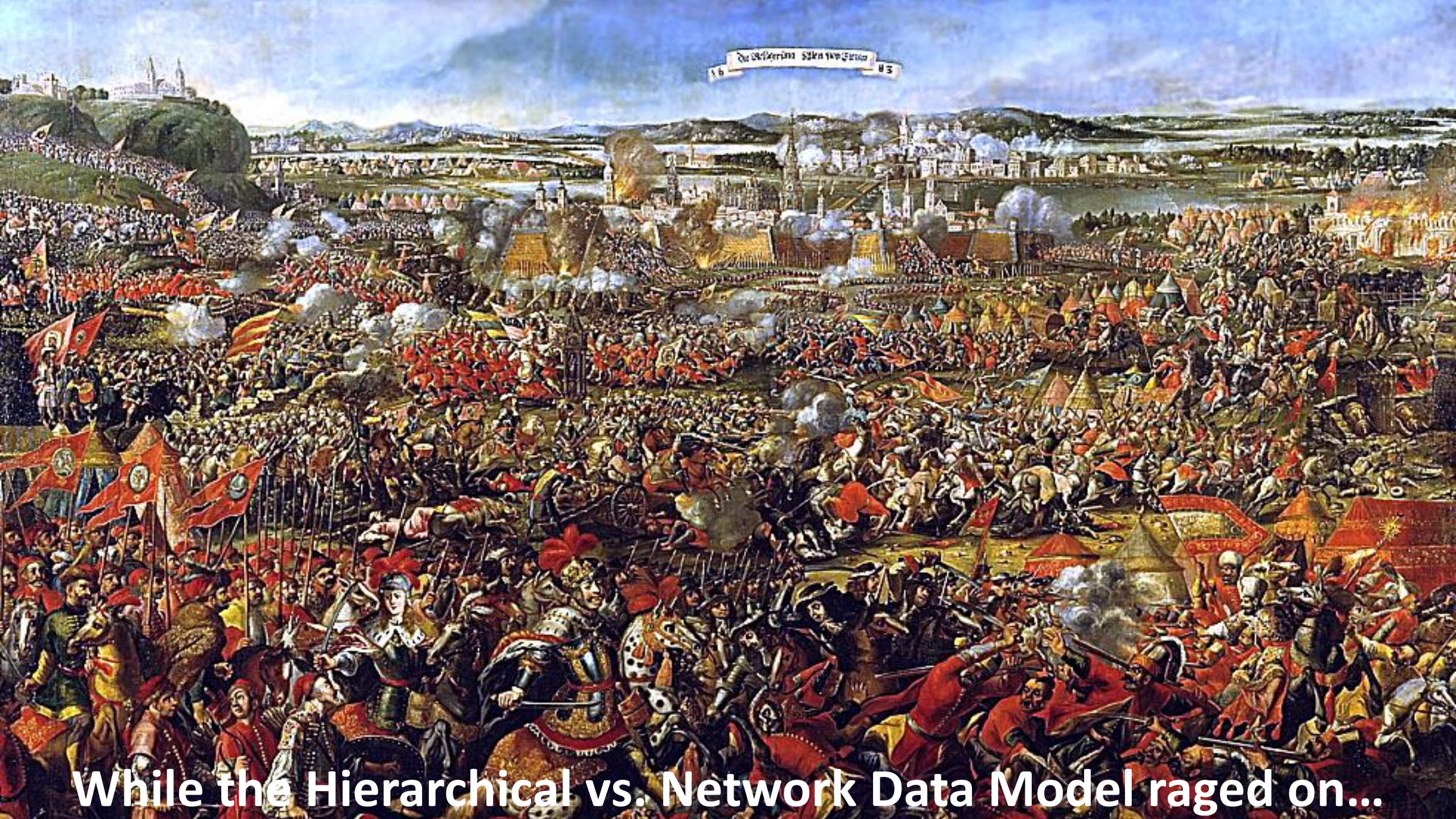


Which employees supervised by Swarat work on Pliny?

Search ↑ good when few employees on Pliny project.

Search ↓ good when few employees supervised by Swarat.

However, code uses one particular access pattern.



While the Hierarchical vs. Network Data Model raged on…

Relational Model (1970)

Department (DName, Supervisor, Address)

AffiliatedWith (DName, EName, PercentEffort)

Employee (EName, Age, Salary)

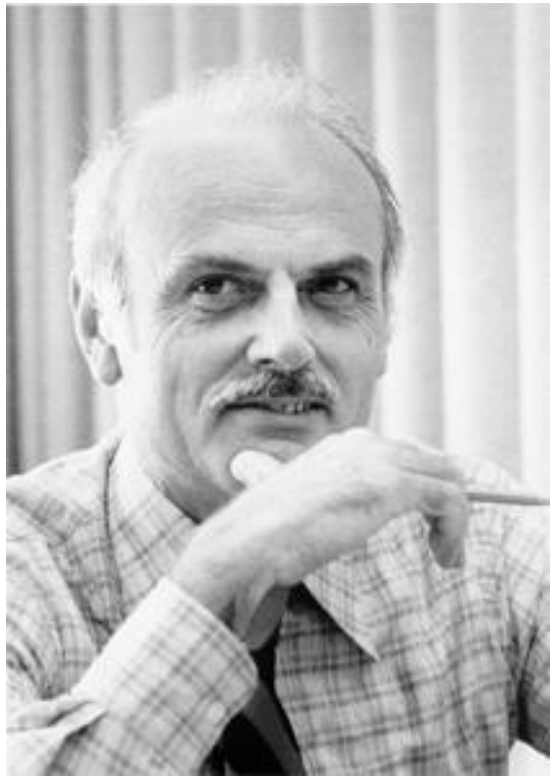
WorksFor (Ename, Pname, Duration)

Project (PName, Director, Customer)

Edgar F Codd:

"A relational model of data for large shared data banks." *Communications of the ACM* 13.6 (1970)

1981 Turing Award



Relational Model

Department (DName, Supervisor, Address)

AffiliatedWith (DName, EName, PercentEffort)

Employee (EName, Age, Salary)

WorksFor (Ename, Pname, Duration)

Project (PName, Director, Customer)

Which employees supervised by Swarat work on Pliny?

Declarative query:

{e.EName | Employee(e) and
exists (w,a,d) (WorksFor(w) and
AffiliatedWith(a) and
Department(d) and
e.EName = w.EName and
w.PName = “Pliny” and
a.EName = e.Ename and
a.DName = d.DName and
d.Supervisor = “Swarat”) }

Relational Model Dominates

- 1969: Published internally at IBM
- 1970: Published externally
- 1974: IBM's System R project begins
- 1979: Oracle RDBMS to market – market cap: $0B to ~$160B now
- 1980: IBM RDBMS to market
- Dominates market and mindshare

Turing Awards: 1981 (Cobb), 2014 (Stonebraker)

Meanwhile… Two kinds of DB content/use

Transactional (OLTP)	Analytical (OLAP)
• E.g., retail sales	• E.g., analyze sales trends
• Track all data – CRUD	• Analyze historical data
• Frequent updates	• Periodic updates
• Flexible queries	• Predetermined queries
• Do everything well	• Optimize for specific uses

Modern History: 1980’s-Present

Object-Oriented DBs (1980s)

Persistent objects

Then: Didn't catch on.

- Simple objects fit well into relational model.
- Lost advantages of declarative queries. Largely repeats hierarchical and network data models.

Now: Idea has returned with more complex objects.

- Same data representation in application & DB.

XML DBs

HTML-like data tags

XML popular for data interchange

- Semi-structured data model – more flexible than relational
- Declarative query language (XQuery)

- Similar to hierarchical model

MapReduce (2008)

Not really a DB, but an algorithm framework for large data sets

- Uses first-order functions – fits well with modern PLs

- Flexible data format – no schemas
- No separate SQL language
- Slow, but easily parallelized

Spark, Hadoop, et al.

MapReduce + many other bulk data operations + some SQL ops

- OO library with first-order functions – fits well with modern PLs
- Effective caching in distributed RAM – avoids repeated I/O of traditional MapReduce

Course Pragmatics

Structure

Classes:	lecture + activities		
Assignments:	9	70% total	7.8% each
Exams:	2	30% total	15% each

Each has additional requirements for grad version.

Information

Canvas – www.canvas.edu

- Schedules & notes
- **Go read Honor Code & course policies!**
- Assignments & grades

Piazza for discussion

- Also linked within Canvas
- **Go sign up!**

Install software before next class

Lots of details, but shouldn’t be difficult.

Software overview

PostgreSQL – a common database management system

- It's a database course!
- Installed on your own computer to avoid any shared server issues.

Jupyter Notebook – a browser-based system for mixing text instructions and runnable code

- Used for in-class exercises.
- Also convenient for beginning assignments.
- This is implemented on top of Python, so we need that too.

PostgreSQL – Installing

www.enterprisedb.com/products-services-training/pgdownload

- Download version 9.6.x. Includes a server, a client, and various tools.
- Use the default options, except…
  - **Choose and remember the password for superuser "postgres".**
  - Running Stack Builder is unnecessary.
- Don't add the EDB Language Pack, since that will add an older version of Python than we want.

Alternate: Everything should work with other PostgreSQL distributions (www.postgresql.org/download/), but we haven't tried them.

PostgreSQL – running

There will be multiple ways that we run SQL code in PostgreSQL:

- From its “shell”: (Windows: Start, Mac: Applications) / PostgreSQL / SQL Shell
- From its “manager”: (Windows: Start, Mac: Applications) / PostgreSQL / pgAdmin
- From within the Jupyter application that you will install.
- From within applications that you will write.

PostgreSQL – setting up course account

We'll create a namespace & account specifically for coursework.

- Separate from anything else you use PostgreSQL for.
- We'll share code that accesses this local namespace (*schema*) & account, so we'll all use the same names.
- **Choose and remember a password for this account.**

Run the SQL shell.

Login with defaults, except for your `postgres` password.

Type the following, except replacing your password.

```
CREATE SCHEMA ricedb;
CREATE USER ricedb PASSWORD 'yourpassword';
GRANT ALL ON SCHEMA ricedb to ricedb;
GRANT ALL ON ALL TABLES IN SCHEMA ricedb to ricedb;
```

Anaconda (Python & Jupyter) – installing

- Uninstall Anaconda 2, if you have it.
  - No need to uninstall Python 2.x, if you have it.

- www.continuum.io/downloads
  - Download version with Python 3.x. Includes Jupyter as a package.
  - Use the default installation.

If you insist: Download Python 3 (www.python.org) and Jupyter (jupyter.org) without Anaconda. Find the installed application `jupyter-notebook` and make it easy to use (e.g., as a shortcut or in your path).

Anaconda – running

You will need to run the Anaconda Navigator.

- (Windows: Start / Anaconda, Mac: Applications) / Anaconda Navigator
- In Windows, to run it with administrator privileges, use right-click / Run as Administrator.

psycopg2 library – installing

This integrates PostgreSQL & Jupyter.

Run Anaconda Navigator with administrator/root privileges.

- Under Environments, show All libraries.
- Click in the box next to psycopg2, and click on Apply.

More details: https://pypi.python.org/pypi/psycopg2

ipython-sql library – installing

This makes SQL easier to use in Jupyter.

Run the shell with administrator/root privileges:

- Windows: Search for Command. Right-click / Run as Administrator, Mac: Applications / Terminal.
- `conda install -c conda-forge ipython-sql=0.3.6`

More details: pypi.python.org/pypi/ipython-sql

Jupyter – setting up

1. Choose a directory that will contain the course “notebooks”.
   - I use `C:\Users\greiner\OneDrive\Courses\COMP 430\2017 Spring\notebooks`
2. Create a configuration file.
   - Run the shell with administrator/root privileges:
     - Windows: Search for Command. Right-click / Run as Administrator, Mac: Applications / Terminal.
   - `jupyter notebook --generate-config`
3. Find your configuration file.
   - Windows: `C:\Users\USERID\`, Mac: `/Users/USERID/`
   - Previous step added a directory there: `.jupyter`
   - Previous step added a configuration file `jupyter_notebook_config.py` in that.
4. Edit the configuration file. Uncomment and complete the line with the notebook path.
   - Windows: Since backslash is the string-escape character, double all the backslashes in the path.
   - I use `c.NotebookApp.notebook_dir = u'C:\\Users\\greiner\\OneDrive\\Courses\\COMP 430\\2017 Spring\\notebooks'`

Jupyter – running

Run within Anaconda Navigator.

- Under Home, click on Jupyter.

Run on its own.

- Windows: Start / Anaconda / Jupyter Notebook

Sample Jupyter notebook – installing & using

- Download the Jupyter notebook 01a-introduction.ipynb from Canvas.
- Put it in the folder that you told Jupyter you would be using.
- Run Jupyter Notebook.
  - In the browser tab for Jupyter, click on 01a-introduction.ipynb.
  - In the browser tab for this notebook, follow the directions.

This will confirm that you have set up everything correctly.

Also, as stated before…

- Read course policies on Canvas.
- Sign up for course on Piazza.

Acknowledgements

Thanks to Chris Ré (Stanford) and Chris Jermaine (Rice) for ideas used in this course's slides, exercises, and assignments. Others are cited where appropriate.