L21: Data models & Relational Algebra

CS3200 Database design (fa18 s2)

https://northeastern-datalab.github.io/cs3200/

Version 11/26/2018

Announcements!

- Exam 2 comments
  - Based on feedback: Separate "select all" and negative points
    - We may have "select all" again: but minimum points is 0
    - We may have negative points again: but for MCQ only (select one single answer)
- Please don't yet submit HW7
  - We are adopting Gradescope to handle your submissions too!

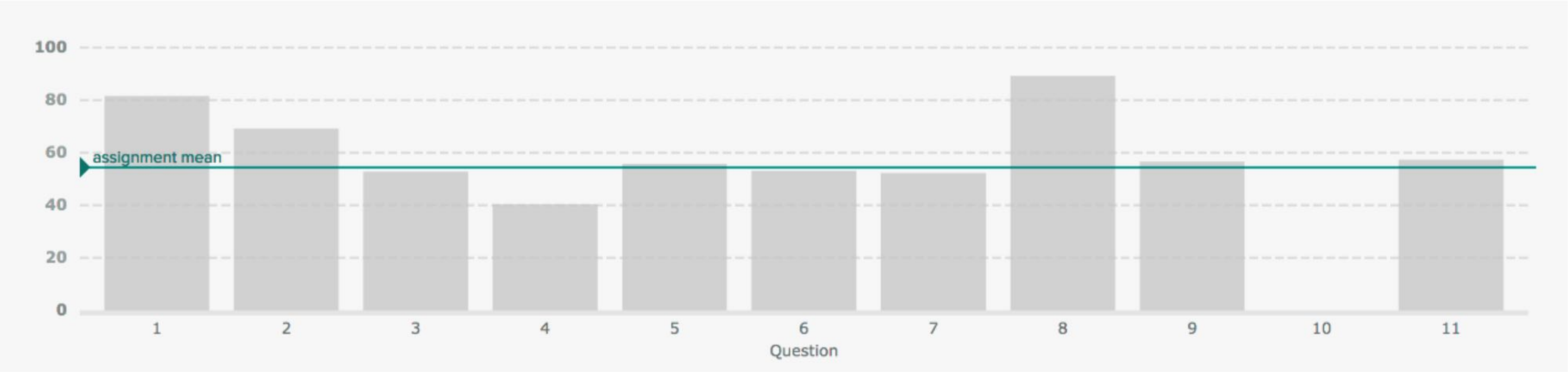
- Today
  - Exam 2 take-aways
  - Data models
  - Relational Algebra

Schedule

NoSQL				
19	R Nov 15	NoSQL		
20	M Nov 19	NoSQL	**Sadalage, Fowler**: Ch 8-11, **Harrison**	
	R Nov 22	No class: Thanksgiving		
Data models and Query Processing				
21	M Nov 26	Data Models, Relational Algebra		**Stonebraker, Hellerstein**
22	R Nov 29	Relational Algebra & Query Optimization		HW7
23	M Dec 3	Course Evaluation, Class Review		HW8, HW10, Q11 Optional PPTX (all due by Dec 5)
	R Dec 6	No class: Reading day		
	T Dec 11	**Exam 3**: 8am-10am, location: TBD		

- 4. NoSQL
  - **Sadalage, Fowler**: NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. 2012 [Safari books eBook (NEU free online access)]. Ch 8: Key-value stoes; Ch 9: Document databases; Ch 10: Column-familiy stores; Ch 11: Graph databases
  - **Harrison**: Harrison. Next Generation Databases: NoSQL, NewSQL, and Big Data. 2016 [Safari books eBook (NEU free online access)]. p43-p51: CAP, Eventual Consistency, Hashing; p61-p62: MongoDB; p65-p68: Graphs; p75-p80: Columns; p145-156: Data models; p163, 164, 166: Secondary indexing
  - **Stonebraker, Hellerstein**: Stonebraker, Hellerstein: What Goes Around Comes Around. 2005 [Online PDF. The paper is a bit dense to read but mind-opening. If you have difficulties reading it directly, feel free to browse through discussions of the paper.

From 1 = Q3 to 11 = Q13 (thus add 2 to the number)

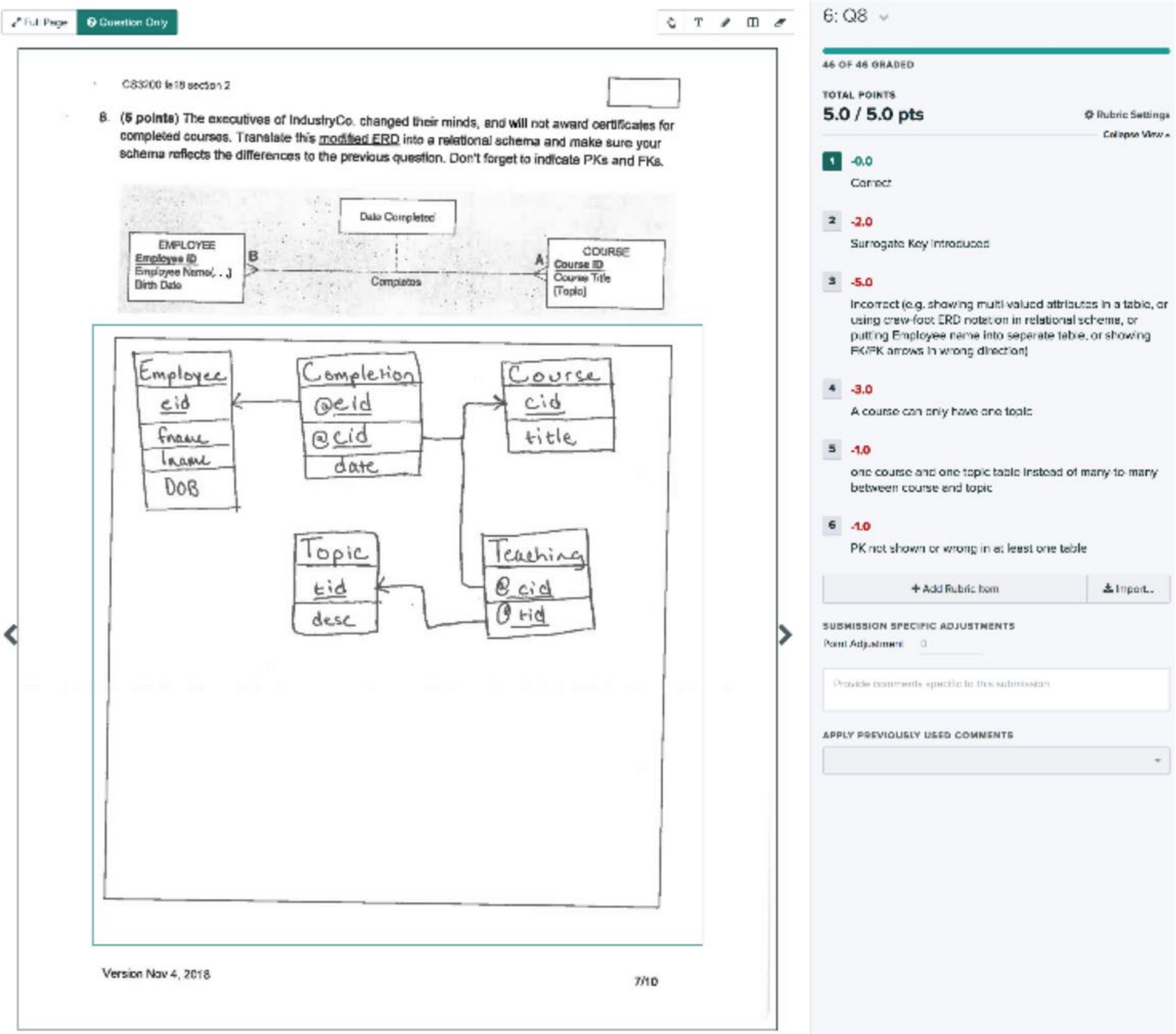


QUESTION 5

Q7 **2** / 5 pts

	– 0 pts	Correct (including the
✔	**– 3 pts**	**No implementation of topic as a multi-valued attribute (thus assuming each course has just one topic)**
	– 5 pts	Incorrect (e.g. an additional table between employee and certificate, or FK/PK not included)
	– 3 pts	Employee split up into multiple tables
	– 2 pts	Certificate with incorrect PK (e.g. composite PK of certificate number and eid and cid)
	– 4 pts	showing a multi-valued attribute in a table
	– 1 pts	Minor issue (e.g. topic as one single table)
	– 4 pts	showing relationship notation from ERD in relational schema

Grading from the "other side"



Q6

6. (**2 points**) Write a SQL DDL statement to create the schema described above. We helped you by providing a template. Just fill in the remaining necessary commands.

```
Create table employee(
    eid int,
    name varchar(100),
    manager_eid int NOT NULL,

    foreign key(manager_eid) references employee(eid),

    primary key(eid)
);
```

Q6

MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
0.0	1.0	2.0	0.8	0.75

RUBRIC	POINTS	PERCENTAGE OF STUDENTS
Correct (PK eid, FK, and not null even if the syntax is a bit off)	+ 0.0	21%
Missing "Not Null" constraint on FK	- 1.0	41%
Incorrect (e.g., PK/FK missing, or a second table, or PK used as FK)	- 2.0	32%
some non-minor error (e.g. major syntax issue, or FK not referencing another table	- 1.0	13%

Q12

12. (**max 1 point, min - 1 point**) Choose all correct statements:

- ◯ A schema that is in 1NF and whose PK is not a composite key, is also in 2NF.
- ◯ For a schema to be in 2NF, all relations need to have unary primary keys.
- ◯ A relation that has exactly one candidate key is always in BCNF.
- ◯ A relation is in BCNF if it has no partial and no transitive FDs
- ◯ A relation with a composite key can never be in 2NF.

Answer: only the first!

Also see post on Piazza: https://piazza.com/class/jj55fszwtpj7fx?cid=135 note

stop following 48 views

Actions

Reviewing Normalization

This page helped me review the basic concepts of FDs/normalization! It might be useful for homework or studying.
#pin

hw6

edit · undo good note | 10

Updated 10 days ago by Niklas Smedemark-Margulies and 2 others

followup discussions *for lingering questions and comments*

Resolved Unresolved

Wolfgang Gatterbauer 24 days ago

Thanks for posting! This is indeed a nice summary.

But please notice everyone that there are some factual and serious errors. I don't blame our student who posted this link so *please* everyone continue posting interesting links! That's great, so we can have interesting discussions on these pages :)

1) a primary key does *not* have to be a single column value (we can have composite PKs)

2) 2NF is incorrectly defined: one can have a table in 2NF with a composite PK (but no partial FD). In other words, the example in 1NF is also in 2NF.

The correct definition is that to be in second normal form (2NF), all non-key attributes must depend on the entire key.

If a relation is in 1NF and has a single column PK, then it is automatically in 2NF. But **not the converse (2nF -> single column: that is wrong!)** Thus the author of this web page must have been confused by reading the converse somewhere.
https://en.wikipedia.org/wiki/Converse_(logic)

L21: A short history of data models

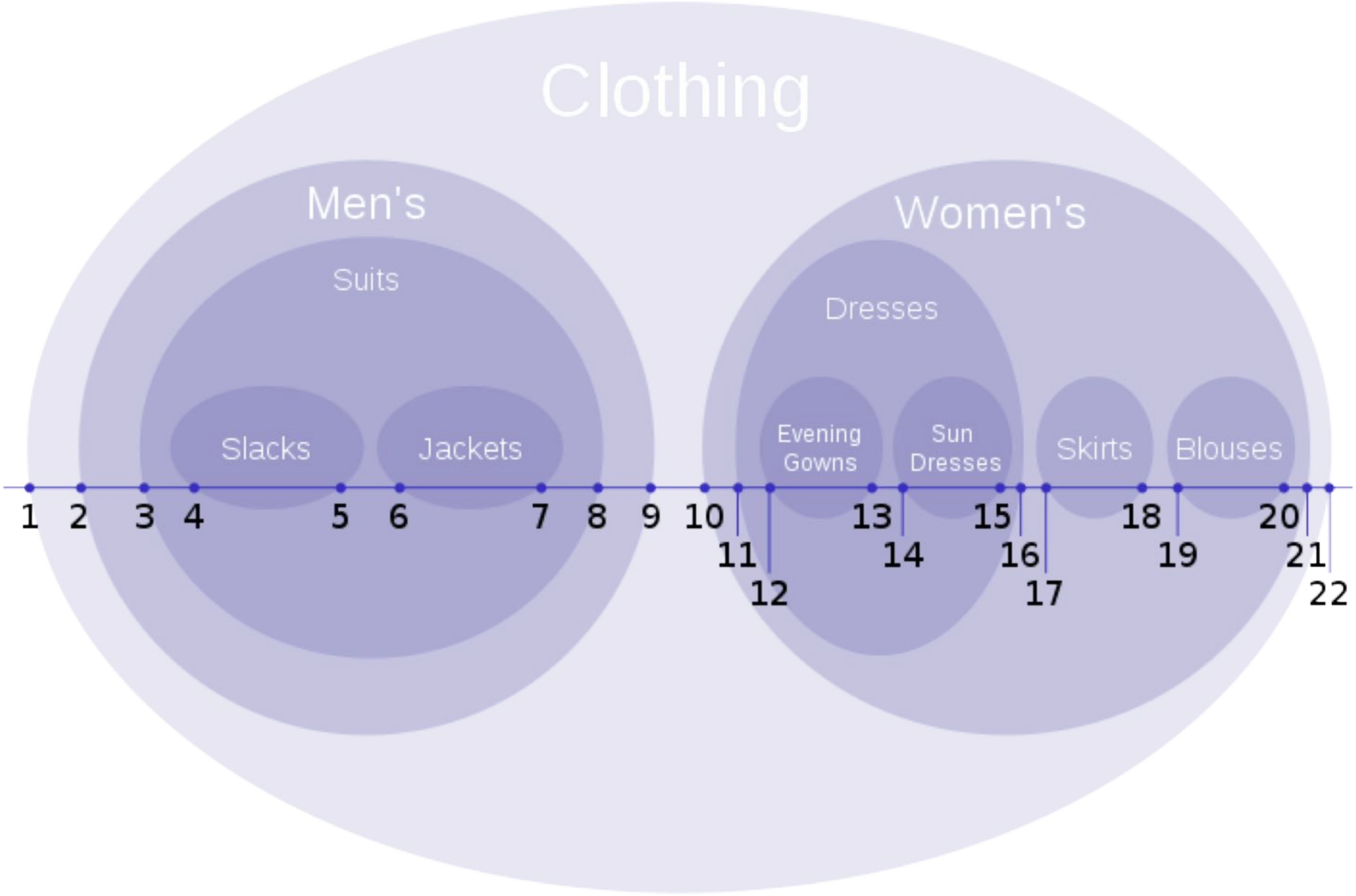
Based on article "What goes around comes around", Hellerstein, Stonebraker, 2005.
Several slides courtesy of Dan Suciu

CS3200 Database design (fa18 s2)

https://northeastern-datalab.github.io/cs3200/

Version 11/26/2018

Hierarchical data



Source: https://en.wikipedia.org/wiki/Nested_set_model

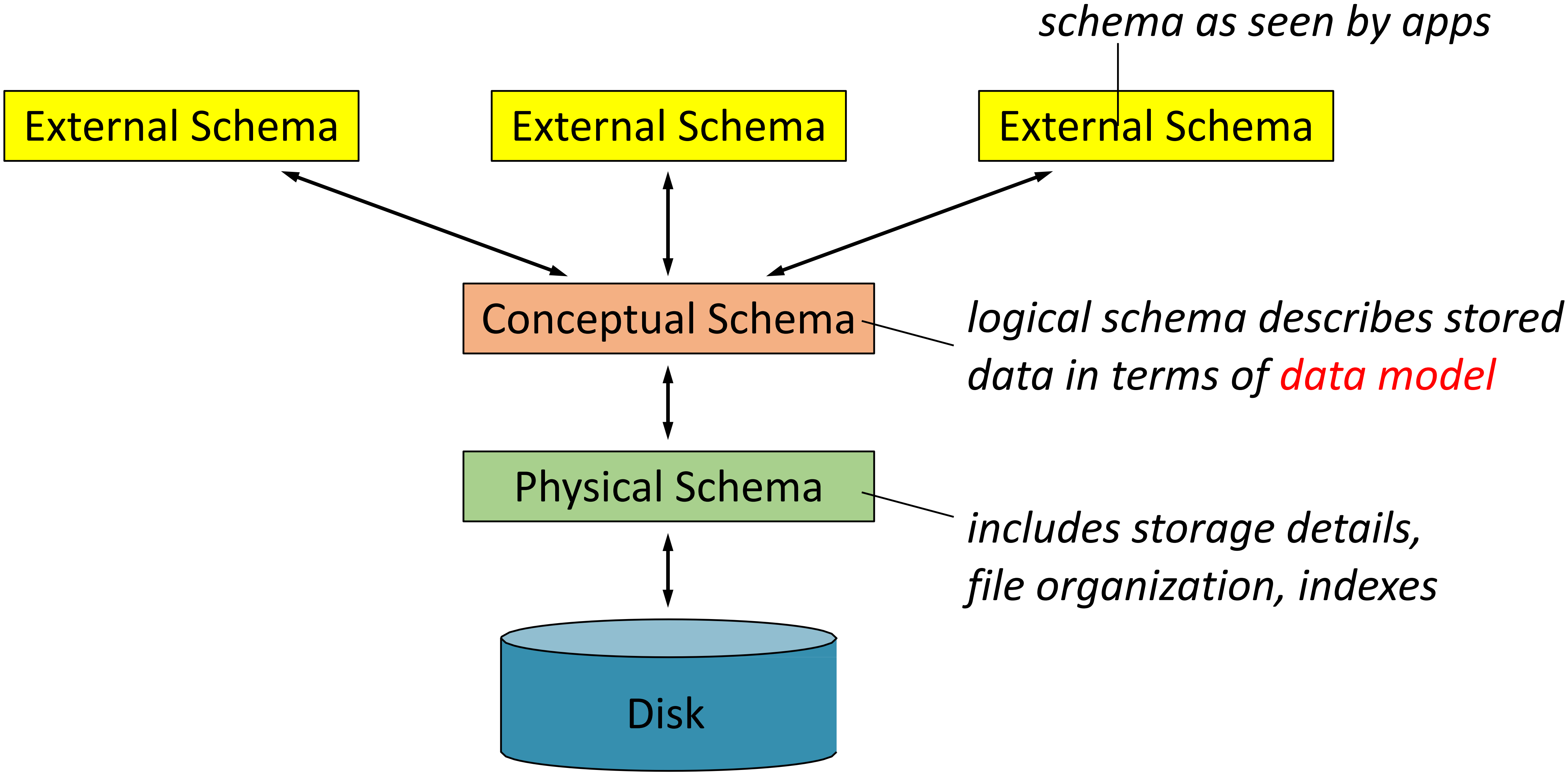
Hierarchies are powerful, but can be misleading...

“Data Model”

- Applications need to model real-world data
  - Typically includes entities and relationships between them
  - Entities: e.g. students, courses, products, clients
  - Relationships: e.g. course registrations, product purchases

- A data model enables a user to define the data using high-level constructs without worrying about many low-level details of how data will be stored on disk

Levels of Abstraction



Outline

- Different types of data
- Early data models
  - IMS
  - CODASYL
- Relational model
- Other data models: E/R Diagrams, XML

Different Types of Data

- Structured data
  - What is this?
  - Examples ?
- Semistructured data
  - What is this?
  - Examples?
- Unstructured data
  - What is this?
  - Examples?

Different Types of Data

- Structured data
- Semistructured data
- Unstructured data

Different Types of Data

- Structured data
  - All data conforms to a schema.
  - Ex: business data
- Semistructured data
  - Some structure in the data but implicit and irregular
  - Ex: resume, ads
- Unstructured data
  - No structure in data.
  - Ex: text, sound, video, images

- What is more important?

Early Proposal 1: IMS

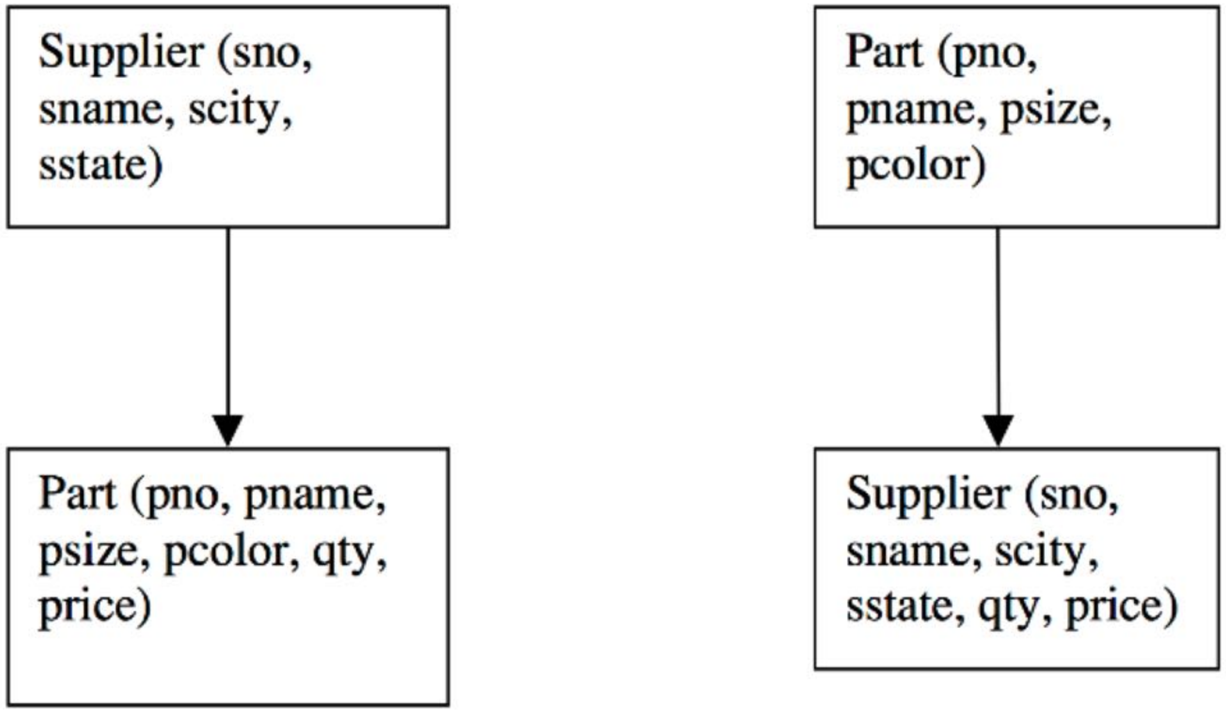
- What is it?
  - Hierarchical data model

- Record
  - **Type**: collection of named fields with data types (+)
  - **Instance**: must match type definition (+)
  - Each instance must have a **key** (+)
  - Record types must be arranged in a **tree** (-)

- **IMS database** ("IBM Information Management System") is collection of instances of record types organized in a tree

Early Proposal 1: IMS

Supplier (sno, sname, scity, sstate)
Part (pno, pname, psize, pcolor)
Supply (sno, pno, qty, price)



How does a programmer retrieve data in IMS ?

How to retrieve data? Data Manipulation Language: DL/1

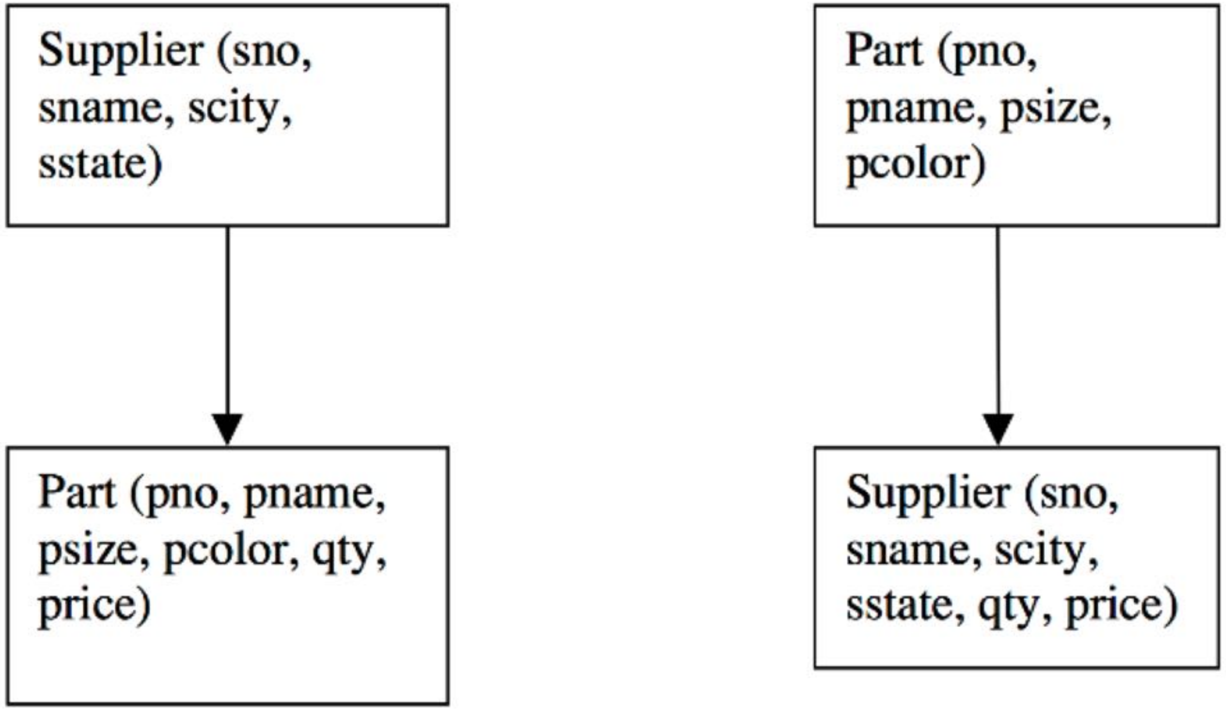
- Each record has a hierarchical sequence key (HSK)
  - Records are totally ordered: depth-first and left-to-right

- HSK defines semantics of commands:
  - get_next
  - get_next_within_parent

- DL/1 is a record-at-a-time language
  - Programmer constructs an algorithm for solving the query
  - Programmer must worry about query optimization

Early Proposal 1: IMS

Supplier (sno, sname, scity, sstate)
Part (pno, pname, psize, pcolor)
Supply (sno, pno, qty, price)



Using the first schema, one can find all the red parts supplied by Supplier 16 as:

Get unique Supplier (sno = 16)
Until failure do
    Get next within parent (color = red)
    Enddo

How is data physically stored?

Data storage

- Root records
  - Stored sequentially (sorted on key)
  - Indexed in a B-tree using the key of the record
  - Hashed using the key of the record
- Dependent records
  - Physically sequential
  - Various forms of pointers

Data Independence: What is that?

- Physical data independence
  - Applications are insulated from changes in <u>physical storage details</u>
- Logical data independence
  - Applications are insulated from changes to <u>logical structure of the data</u>
- Why are these properties important?
  - Reduce program maintenance
  - Logical database design changes over time
  - Physical database design tuned for performance

IMS Limitations

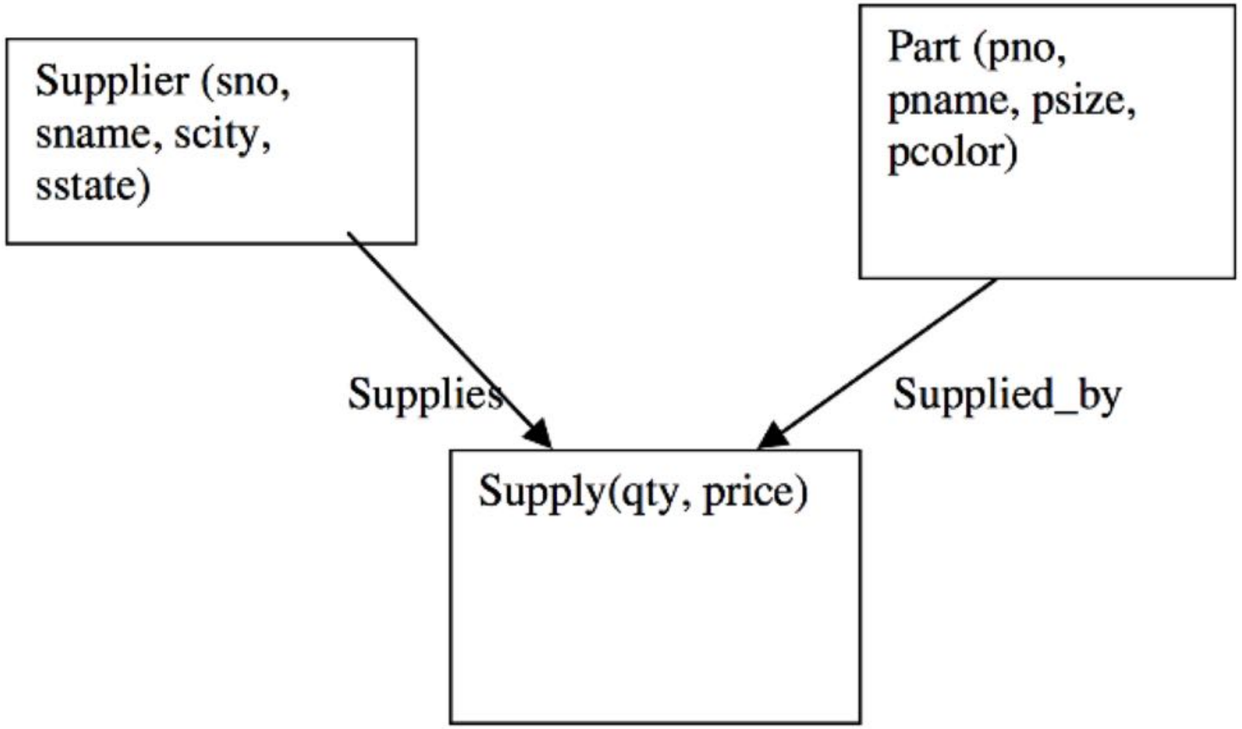
- Tree-structured data model
  - Redundant data, repetition of information (m-to-n relationships)
  - existence depends on parent, artificial structure
- Record-at-a-time user interface
  - User must specify algorithm to access data
- Very limited physical independence
  - Phys. organization limits possible operations
  - Application programs break if organization changes
- Provides some logical independence
  - DL/1 program runs on logical database
  - Difficult to achieve good logical data independence with a tree model

Early Proposal 2: CODASYL

- What is it?
  - Networked data model

- Primitives are also record types with keys (+)
- Network model is more flexible than hierarchy (+)
  - Ex: no existence dependence
- Record types are organized into network (-)
  - A record can have multiple parents
  - Arcs between records are named
  - At least one entry point to the network
- Record-at-a-time data manipulation language (-)

CODASYL example



Supplier (sno, sname, scity, sstate)
Part (pno, pname, psize, pcolor)
Supply (sno, pno, qty, price)

CODASYL Limitations

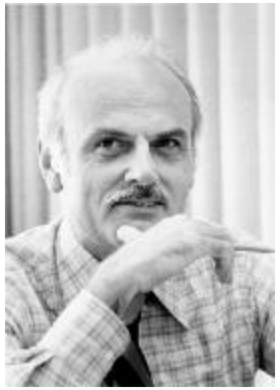
- No physical data independence
  - Application programs break if organization changes
- No logical data independence
  - Application programs break if organization changes
- Very **complex**
  - multi-dimensional space (i.e., A space of records)
- Programs must "**navigate** the hyperspace"
  - Charles Bachmann, 1973 ACM Turing Award,
Turing Lecture: "The Programmer As Navigator"
  - Access data by following pointers between records
- Load and recover as **one gigantic object**

Navigational Database Era (Early1960 – Early 1970)

- Representative Navigational Database Systems
  - Integrated Data Store (IDS), 1964, GE
  - Information Management System (IMS), 1966, IBM
  - Integrated Database Management System (IDMS), 1973, Goodrich

- CODASYL
  - Short for "Conference/Committee on Data Systems Languages"
  - Define navigational data model as standard database interface (1969)

The Birth of Relational Model



- Ted Codd
  - Born in 1923
  - PHD in 1965
  - “A Relational Model of Data for Large Shared Data Banks” in 1970
- Relational Model
  - Organize data into a collection of relations
  - Access data by a declarative language
    (i.e., tell me what you want, not how to find it)

Data
Independence

- Some early work by David L. Childs (somewhat forgotten by history)

Relational Model Overview

- Proposed by Ted Codd in 1970
- Motivation: better logical and physical data independence
- Defines logical schema only
  - No physical schema
- Set-at-a-time query language

Physical Independence

- Definition: Applications are insulated from <u>changes in physical storage details</u>

- Early models (IMS and CODASYL): No

- Relational model: Yes
  - Yes through **set-at-a-time language: algebra or calculus**
  - No specification of what storage looks like
  - Administrator can optimize physical layout

Logical Independence

- Definition: Applications are insulated from changes to logical structure of the data

- Early models
  - IMS: some logical independence
  - CODASYL: no logical independence

- Relational model
  - Yes through views (think fixed SQL queries: give me first and last name of all students)

Great Debate

- Pro relational
  - What where the arguments ?

- Against relational
  - What where the arguments ?

Great Debate

- Pro relational
  - CODASYL is too complex
  - CODASYL does not provide sufficient data independence
  - Record-at-a-time languages are too hard to optimize
  - Trees/networks not flexible enough to represent common cases
- Against relational
  - COBOL programmers cannot understand relational languages
  - Impossible to represent the relational model efficiently
  - CODASYL can represent tables
  - Transitive closure (initially) performance
  - (initially) too complex and mathematical languages
- Ultimately settled by the market place

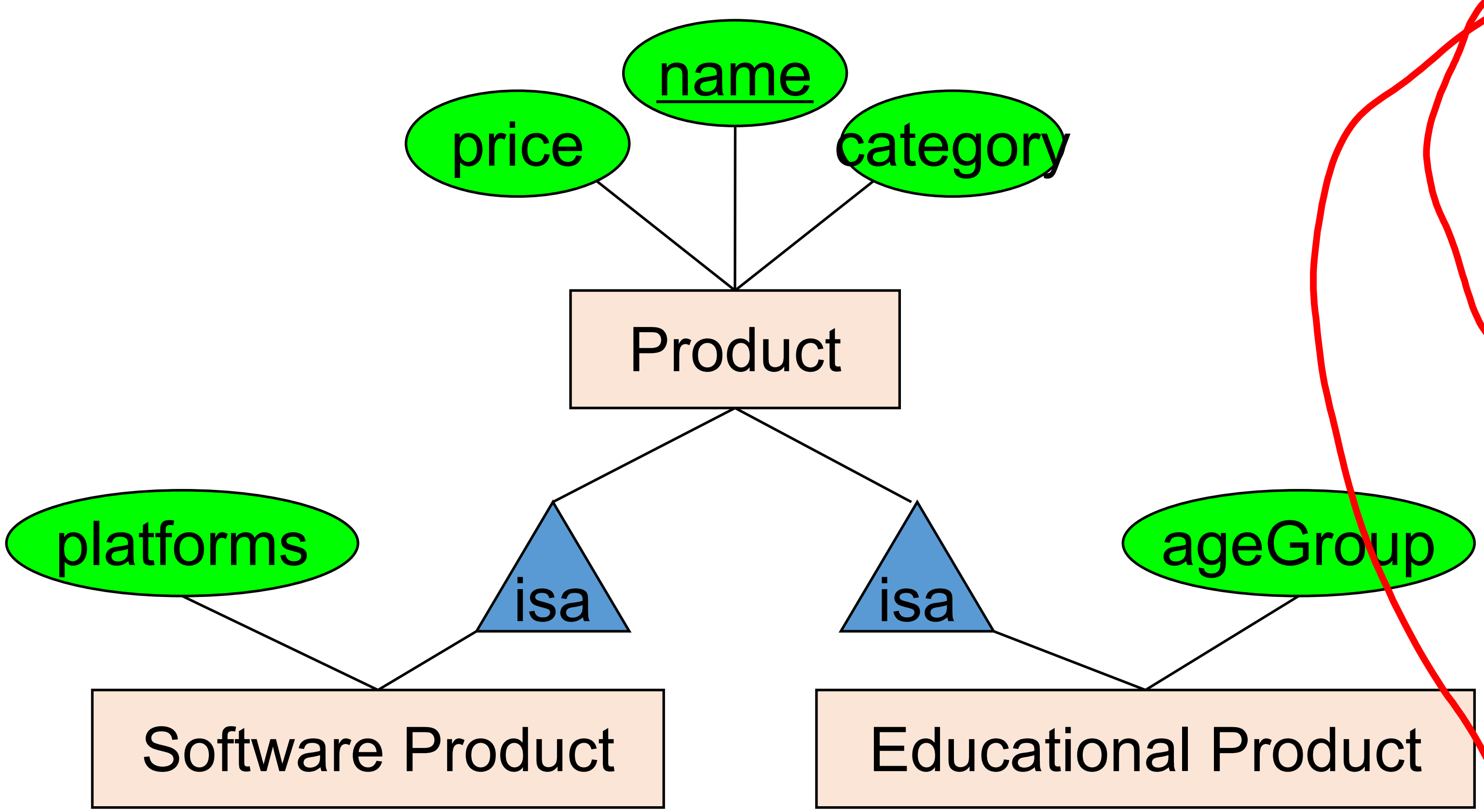
- Don Chamberlin of IBM was an early CODASYL advocate (later co-invented SQL)
  - “He (Codd) gave a seminar and a lot of us went to listen to him. This was as I say a revelation for me because Codd had a bunch of queries that were fairly complicated queries and since I’d been studying CODASYL, I could imagine how those queries would have been represented in CODASYL by programs that were five pages long that would navigate through this labyrinth of pointers and stuff. Codd would sort of write them down as one-liners. These would be queries like, "Find the employees who earn more than their managers." [laughter] He just whacked them out and you could sort of read them, and they weren’t complicated at all, and I said, "Wow." This was kind of a conversion experience for me, that I understood what the relational thing was about after that.”

Source: http://www.mcjones.org/System_R/SQL_Reunion_95/sqlr95-Prehisto.html , http://wiki.c2.com/?HistoryOfRelational

Other Data Models

- Entity-Relationship: 1970’s
  - Successful in logical database design
- Extended Relational: 1980’s
  - e.g., Aggregation
- Object-oriented: late 1980’s and early 1990’s
  - Address impedance mismatch: relational dbs !" OO languages
  - Interesting but ultimately failed (several reasons, see paper)
- Object-relational: late 1980’s and early 1990’s
  - User-defined types, ops, functions, and access methods
- Semi-structured: late 1990’s to early 2000's
  - reborn as document stores, JSON

ERD: Subclasses in ERD to Relations



Product

<u>Name</u>	Price	Category
Gizmo	99	gadget
Camera	49	photo
Toy	39	gadget

Educational Product

<u>Name</u>	ageGroup
Gizmo	toddler
Camera	adult

Software Product

<u>Name</u>	platforms
Gizmo	unix

XML Syntax

```xml
<bibliography>
    <book>    <title> Foundations… </title>
              <author> Abiteboul </author>
              <author> Hull </author>
              <author> Vianu </author>
              <publisher> Addison Wesley </publisher>
              <year> 1995 </year>
    </book>
    …
</bibliography>
```

Tags: book, title, author, …

Start tag: <book>,   end tag: </book>

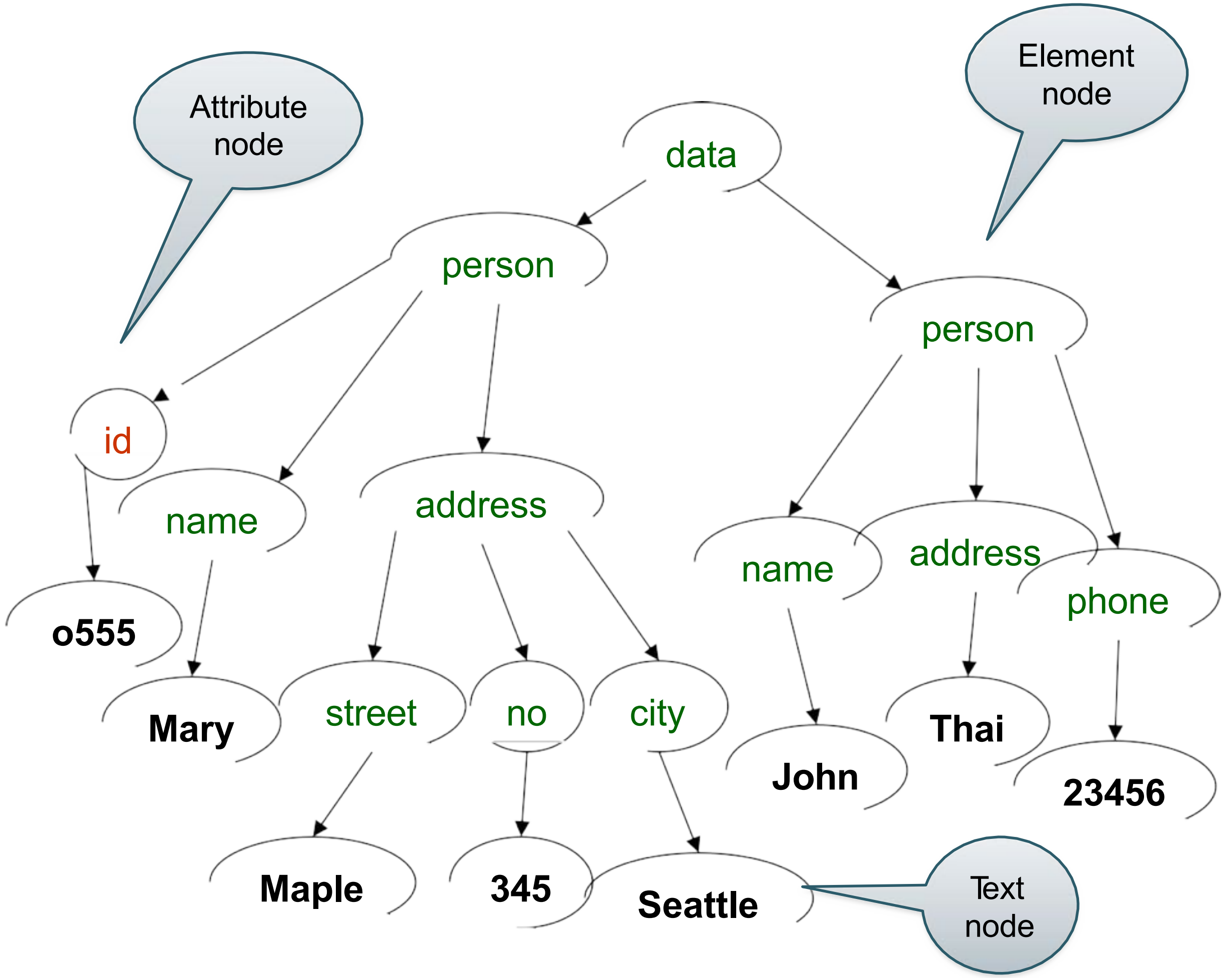
Elements: <book>…</book>,<author>…</author>

Elements are nested

An XML document: *single root element*

XML Semantics: a Tree !
DOM = Document Object Model

```
<data>
  <person id="o555" >
    <name> Mary </name>
    <address>
      <street>Maple</street>
      <no> 345 </no>
      <city> Seattle </city>
    </address>
  </person>
  <person>
    <name> John </name>
    <address>Thailand
    </address>
    <phone>23456</phone>
  </person>
</data>
```



Order matters !!!

41

XML Data

- XML is self-describing
- Schema elements become part of the data
  - Relational schema: person(name,phone)
  - In XML <person>, <name>, <phone> are part of the data, and are repeated many times
- Consequence: XML is much more flexible
- XML = semistructured data

Summary

- Data independence is desirable
  - Both physical and logical
  - Early data models provided very limited data independence
  - Relational model facilitates data independence
    - Set-at-a-time languages facilitate physical indep.
    - Simple data models facilitate logical indep.
- Flat models are also simpler, more flexible
- User should specify what they want not how to get it (declarative)
  - Query optimizer does better job than human

- New data model proposals must
  - Solve a “major pain” or provide significant performance gains