Hierarchical (IMS) (late 60s-70s)

+ facilitates simple data manipulation language (DL/I)
- Information is repeated
- Existence depends on parents
- no physical data independence (can’t tune physical level without tuning app)
- Not much logical data independence either (can’t tune schema without changing app (think views))

Lessons from hierarchical:

Lesson 1. Physical and logical data independence are highly desirable

Lesson 2. Tree structured data models are very restrictive

Lesson 3. It’s a challenge to provide sophisticated logical reorganizations of tree structured data

Lesson 4. Record-at-a-time user interface forces manual query optimization (hard!)

Directed graph (CODASYL) (70s)

+Yeah! Graphs, not trees!

+ Can model many-to-many relationships

- Still no physical data independence
- Much more complex than IMS
- Lesson 5: Directed graphs are more flexible than hierarchies, but more complex
- Lesson 6: Loading and recovering directed graphs is more complex than hierarchies

Relational (70s-early 80s)

1. Store the data in a simple data structure (tables)
2. Access it through a high level set-at-a-time DML
3. No need for a physical storage proposal

- Lots of good arguing by various sides “the great debate”
- Non-technical factor: CODASYL systems were not portable → not porting to first microprocessors (VAX) (whoops)

Lessons from Relational:

Lesson 7: Set-at-a-time languages are good; offer improved physical data independence

Lesson 8: logical data independence is easier with a simple data model than with a complex one

Lesson 9: Technical debates are usually settled by the elephants of the marketplace, and often for reasons having little to do with the technology

Lesson 10: query optimizers can beat all but the best record at a time DBMS application programmers

Discussion Questions

- (Michael) How could this rift between theoretical academics and practical implementations be addressed from a research perspective? (i.e. how can academics prove that their work is worthy of industry’s funding?)

ER (70s)

- Response to normalization
- Standard wisdom: create table, then normalize. Problems for DBAs:
  - – 1. Where do I get initial tables
  - – 2. can’t understand functional dependences
- Lesson 11: Functional dependencies are too difficult for mere mortals to understand. Another reason for KISS

Extended Relational (80s)

- How many features must relational databases have…
  - Set valued attributes
  - Aggregation
  - Generalization
  - And many, many more

Lesson 12: unless there is a big performance or functionality advantage, new constructs will go nowhere

Semantic (late 70’s and 80’s) (SDM)

- Similar ideas, but more radical; change whole model to be semantically richer.

- Lots of machinery, little benefit. Died without a trace.

Object-oriented (late 80’s and early 90’s)

+Support OO languages

- market failure: no leverage, no standards, some versions had reliance on C++

Lesson 13: Packages will not sell to users unless they are in “major pain”

Lesson 14: Persistent languages will go nowhere without support of PL community

Discussion Questions

- (Michael) Limitations in practicality (hardware) prevented ideas from coming to fruition. What other aspects or challenges may cause a great idea to be proposed or explored “in the wrong era”?

Object-relational (late 80s and early 90s)

- OO + R

+ Some commercial success

+ put some code in DBMS

- no standards

Lesson 14: OR puts code in DB which makes for fast adaptability

Lesson 15: Widespread adoption of new technology requires either standards and/or an elephant pushing hard

Discussion Questions

- (Sarah/Sid) How do we decide which research is worth revisiting?
  - Who makes these decisions? Industry or academia
  - What spurs these types of decisions?
  - Is it worth it?

XML (late 90s to - ?)

- Semantic heterogeneity
- Schema later: best for semi-structured… authors claim there aren't that many of these
- XML Schema:
  - Can be hierarchical, as in IMS
  - Can have links to other records as in CODASYL & SDM
  - Can have set-based attributes as in SDM
  - Can inherit from other records, as in SDM
  - Even more complexity!

Three visions of the future of XML Schema:

- XML schema fails because of excessive complexity
- A “data-oriented” subset of XML Schema will be proposed that is vastly simpler
- “It will become popular. Within a decade, all problem with IMS and CODASYL that motivated Codd to invent the relational model will resurface. At that time some enterprising researcher, call him Y, will ‘dust off’ Codd’s original paper, and there will be a replay of ‘the Great Debate’ Presumably it will end the same way as the last one. Moreover, Codd won the Turing award in 1981 for his contribution. In this scenario, Y will win the Turing award circa 2015”.

Lessons from XML

Lesson 16: Schema-later is probably a niche market

Lesson 17: XQuery is pretty much OR SQL with a different syntax

Lesson 18: XML will not solve semantic heterogeneity either inside or outside the enterprise

Discussion Questions

- (Rachel) The authors claim that XML still doesn't solve the semantic heterogeneity problem. So What's the semantic heterogeneity problem (in plain terms) and what is missing from the XML approach?
- (Rachel) In the future, which of the following do you think will occur:

  1)XML Schema will fail because of its complexity

  2) A “data-oriented” subset of XML Schema will be proposed that is vastly simpler

  3) XML will become popular and replay of the “Great Debate”

Debate

- One side represents academia
- One side represents industry
- Rachel is a new investor who is interested in propelling database research into the future
- Convince her that your side deserves the funding
  - Try to work arguments from the paper into your explanation