What Goes Around Comes Around

Shiqi Liu

2017/1/23

INDEX

1
2
3

IMS

CODASYL

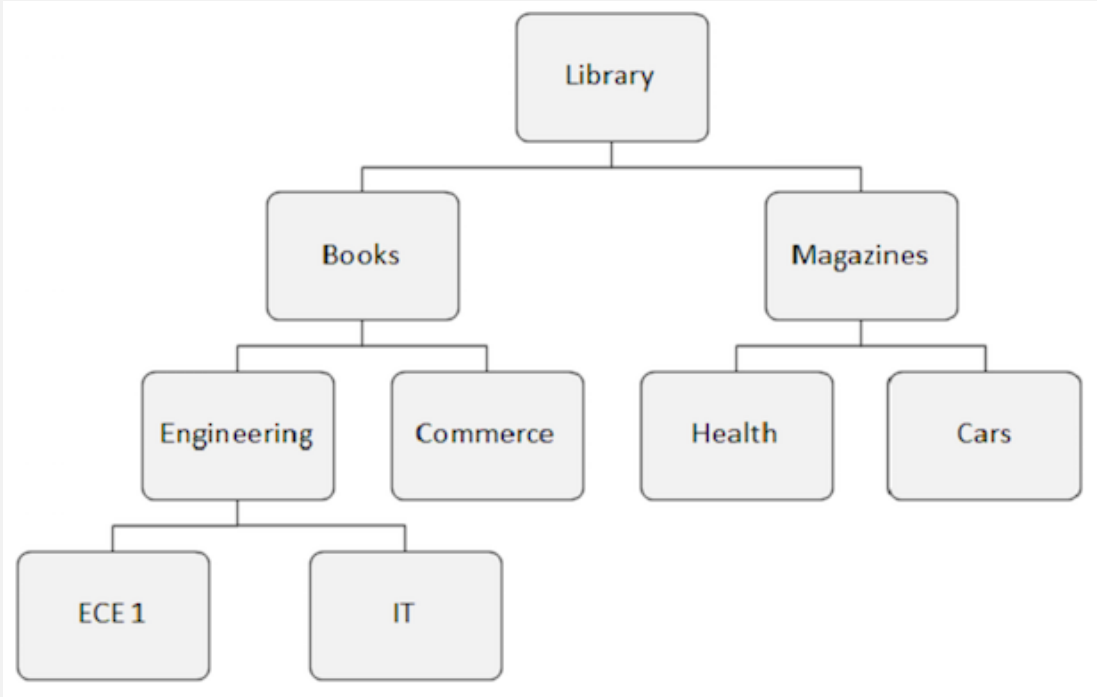
Relational


MetLife
IMS

1 Hierarchical data model

1 Record Type

2 Tree & Dependency



1 Hierarchical data model

Suppplier

16 General Supply Boston Ma
24 Special Supply Detroit Mi

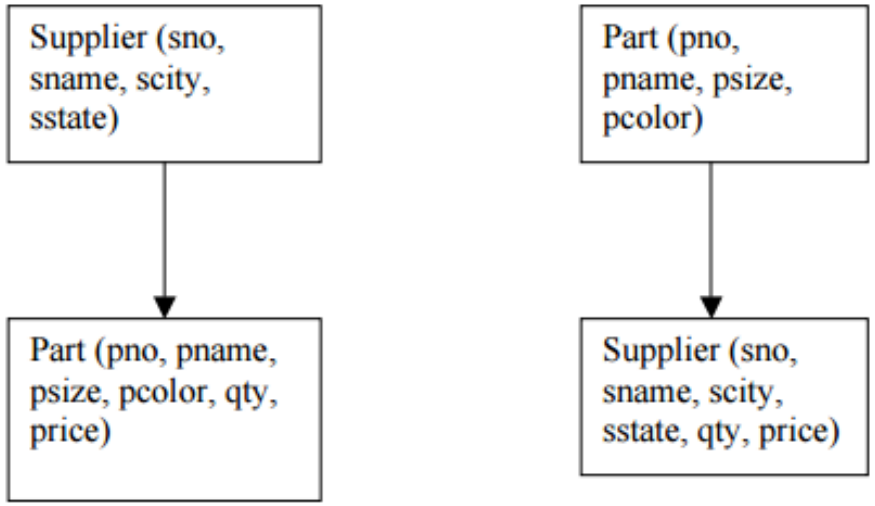
Part

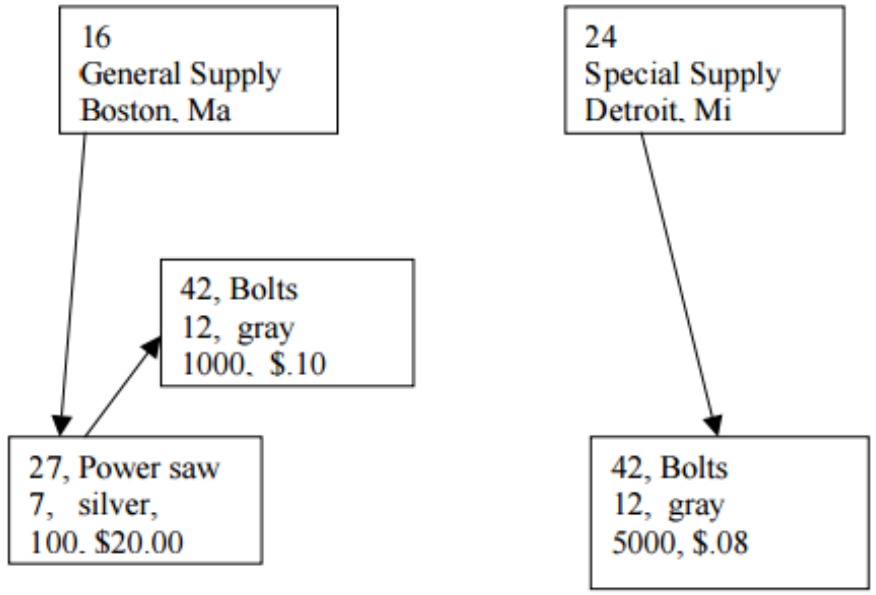
27 Power saw 7 silver
42 bolts 12 gray

Supply

16 27 100 $20.00
16 42 1000 $.10
24 42 5000 $.08

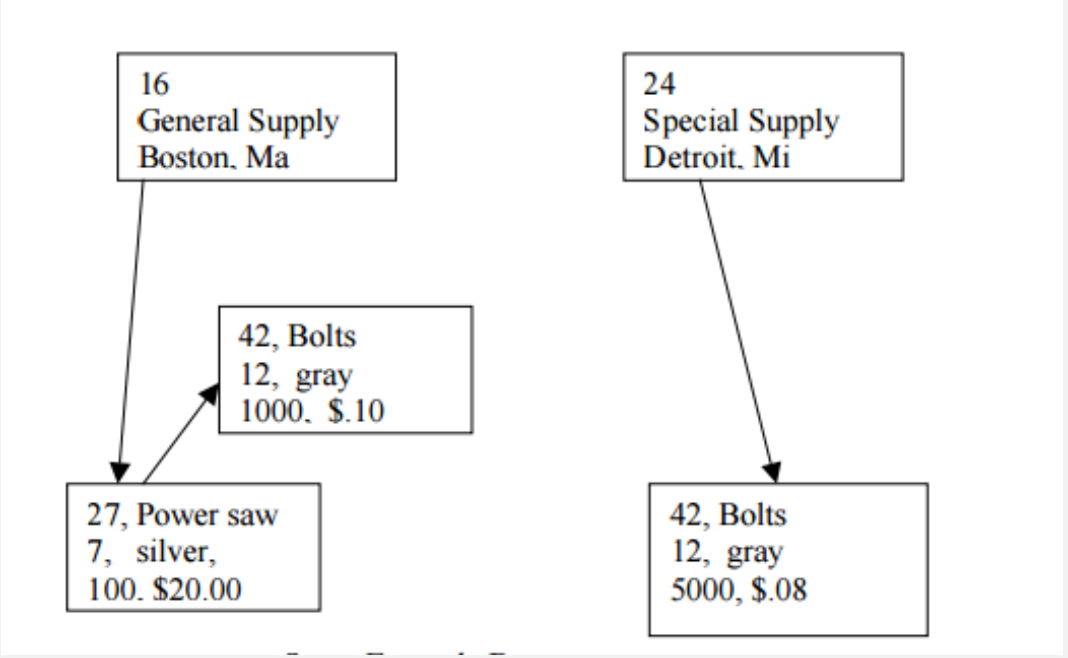
Some Sample Data
Figure 2





1 Hierarchical data model

Problem:



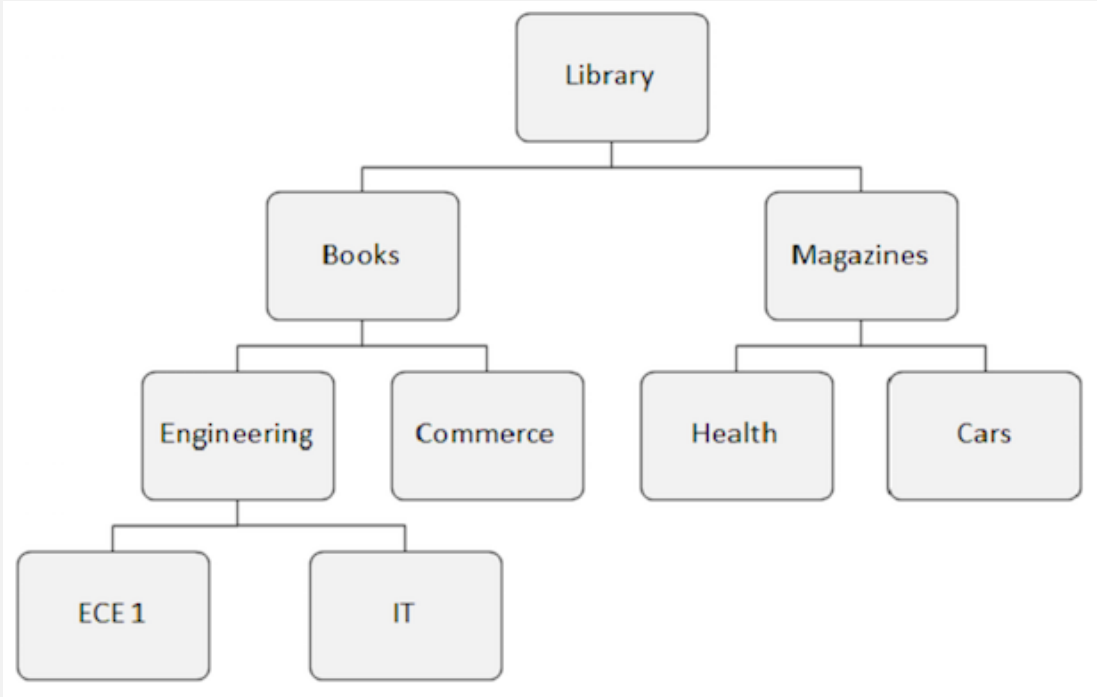
Repeated, Redundancy and Inconsistent

What about the parts that currently no one supplies?

1 Hierarchical data model

1 Tree & Dependency

2 Record Type

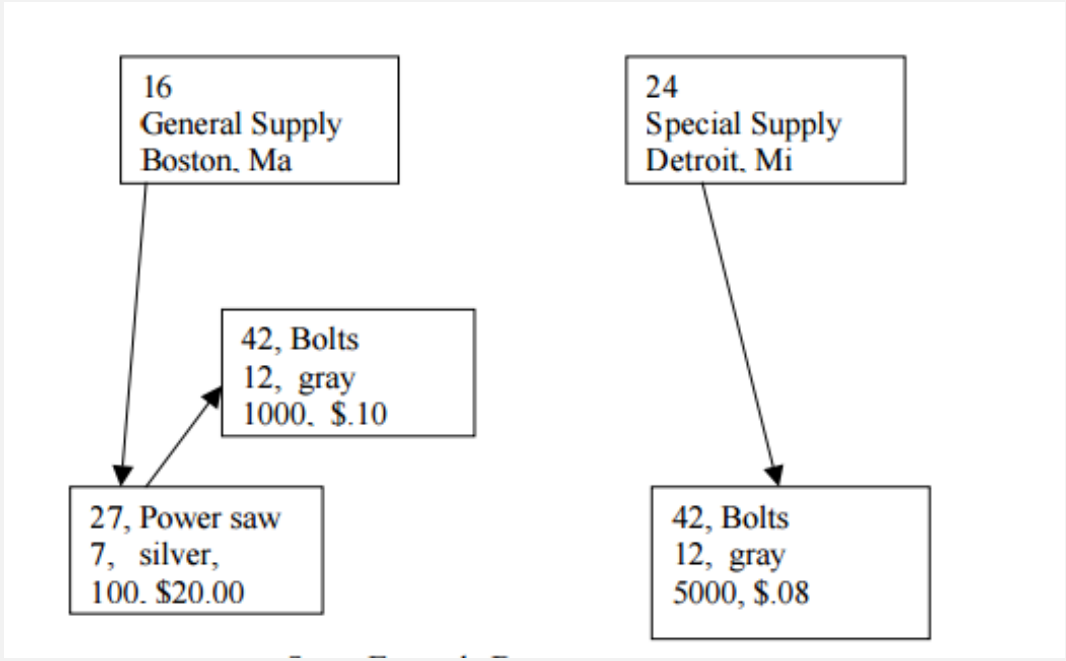


Record-at-a-time Language

2 DL/I Programming Language

Find red parts supplied by supplier 16

```
Get unique Supplier (sno = 16)
Until no-more {
        Get next within parent (color = red)
}
```

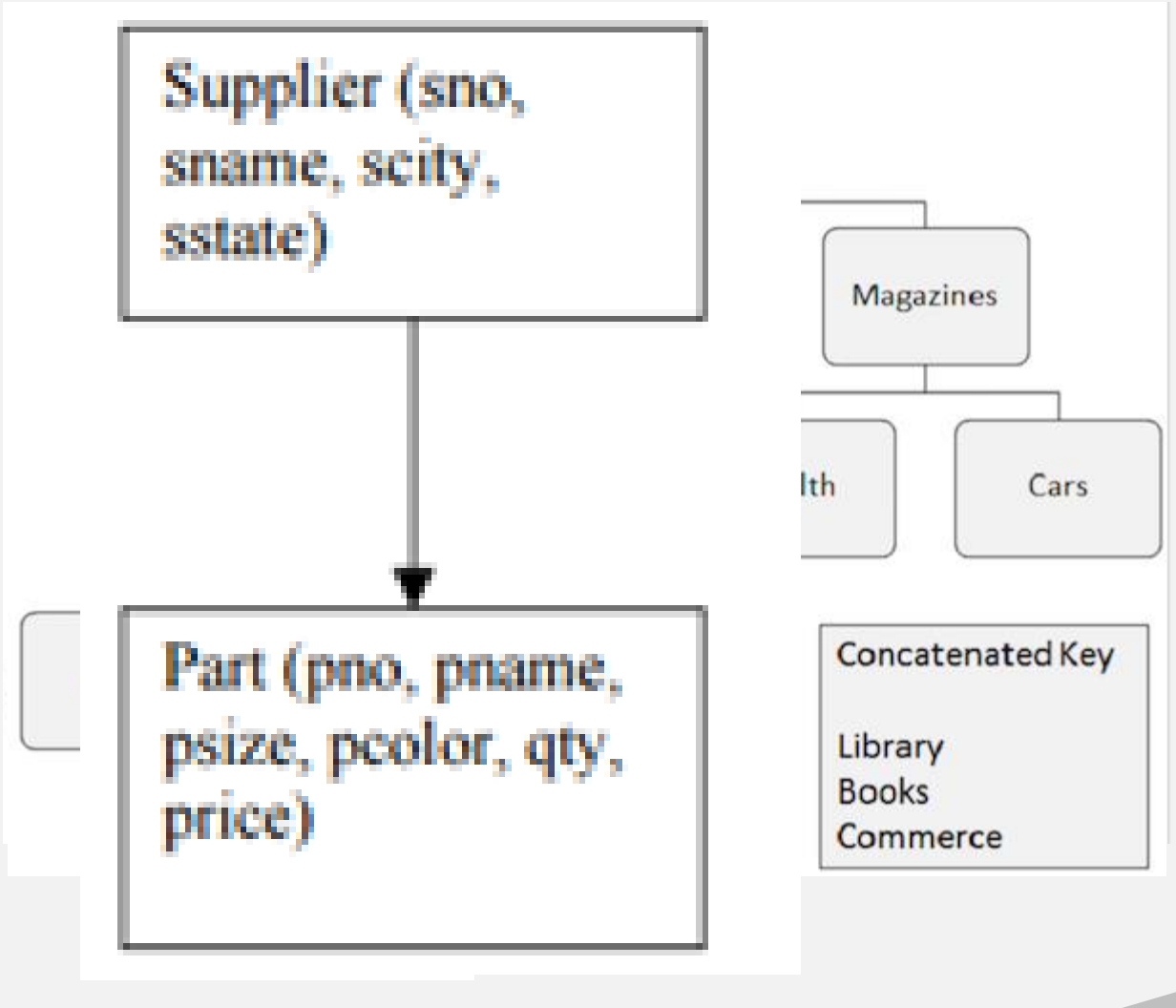


2 DL/I Programming Language

Sequential Access

Random Access

```
Get unique Supplier (sno = 16)
Until no-more {
        Get next within parent (color = red)
}
```



Problems

DL/I Programming Language

Physical data independence

What is physical data independence?

Why is it important?

Why IMS doesn’t have it?

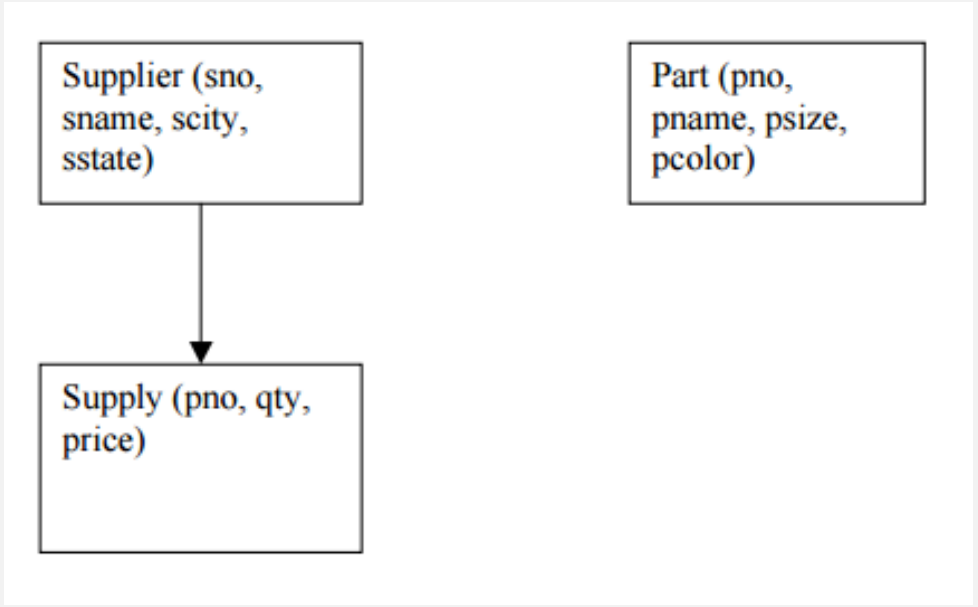
Logical data independence

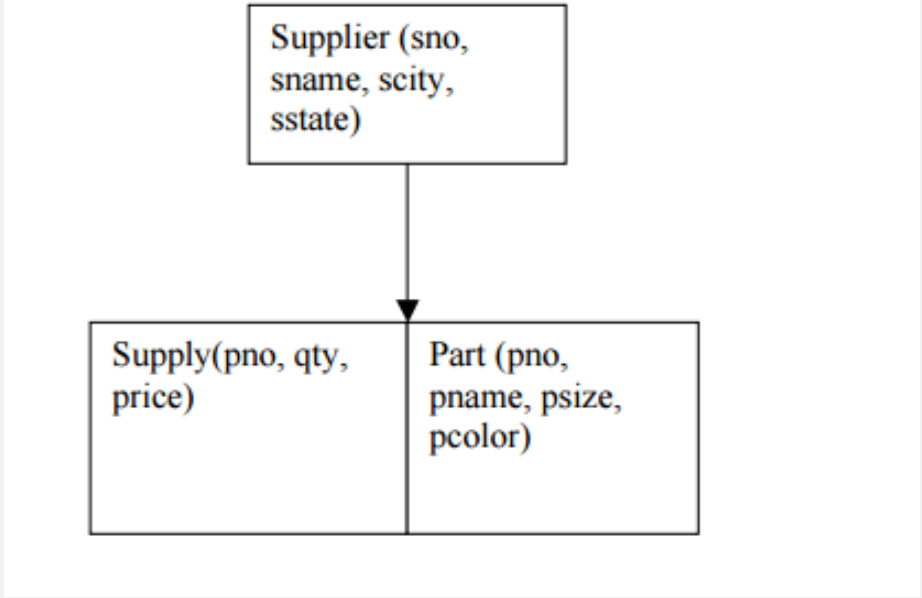
What is logical data independence?

Why is it important?

Why IMS doesn’t have it?

3 View


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


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

Lesson Learned

Lesson 1: Physical and logical data independence are highly desirable

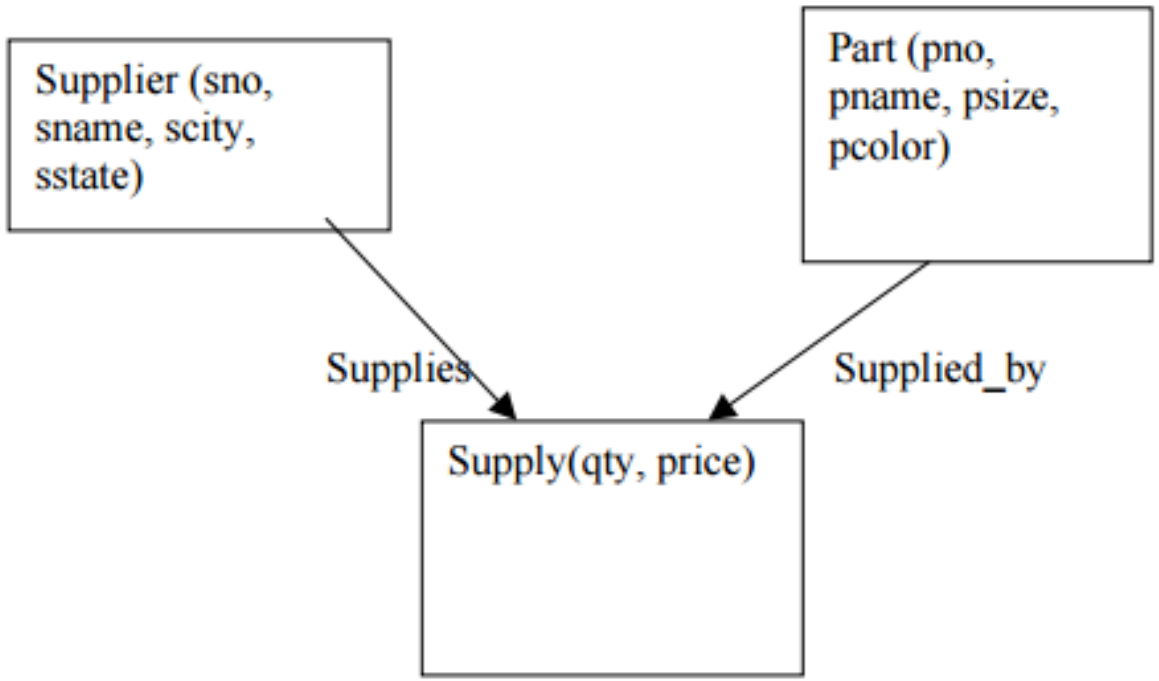
Lesson 2: Tree structured data models are very restrictive

Lesson 3: It is a challenge to provide sophisticated logical reorganizations of tree structured data

Lesson 4: A record-at-a-time user interface forces the programmer to do manual query optimization, and this is often hard.


CODASYL

1 Directed graph data model



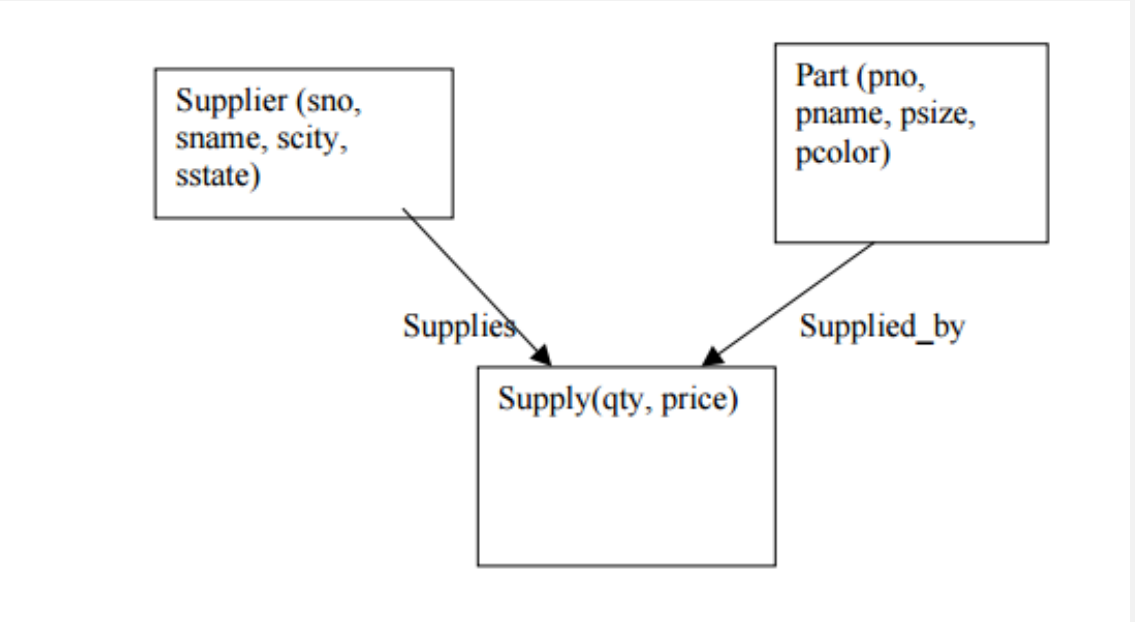
Record & Set

Redundancy Solved

Part not supplied by anyone Solved

2 Still record-at-a-time

```
Find Supplier (SNO = 16)
Until no-more {
  Find next Supply record in Supplies
  Find owner Part record in Supplied_by
  Get current record
 -check for red—
}
```



How hard is this?

2 Still record-at-a-time

The last record touched by the application
The last record of each record type touched
The last record of each set type touched

Navigating in hyperspace

Still record-at-a-time

Physical independence?

No, example: hash key change

```
Find Supplier (SNO = 16)
Until no-more {
  Find next Supply record in Supplies
  Find owner Part record in Supplied_by
  Get current record
 -check for red—
}
```

Still record-at-a-time

Logical independence?

Still partial with logical database.

Still record-at-a-time

New Problem

Data Combined too tight.

Bulk load takes too long

Tiny corrupted data cost huge recovery

Lesson Learned

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

Motivation

IMS programmers spend too much time maintaining their applications.

Let’s focus on better data independence!

1 Threefold

Store the data in a simple data structure (tables)

Access it through a high level set-at-a-time DML

No need for a physical storage proposal

CODASYL

a) Nothing as complex as CODASYL can possibly be a good idea
b) CODASYL does not provide acceptable data independence
c) Record-at-a-time programming is too hard to optimize
d) CODASYL and IMS are not flexible enough to easily represent common situations

Relational

a) COBOL programmers cannot possibly understand the new-fangled relational languages
b) It is impossible to implement the relational model efficiently
c) CODASYL can represent tables, so what's the big deal?

CODASYL's Response

a) It is possible to specify set-at-a-time network languages, such as LSL [TSIC76], that provide complete physical data independence and the possibility of better logical data independence.
b) It is possible to clean up the network model [CODA78], so it is not so arcane.

Relational

a) Codd is a mathematician, and his languages are not the right ones. SQL and [STON76] are much more user friendly.
b) System R and INGRES prove that efficient implementations of
Codd's ideas are possible. Moreover, query optimizers can be built that are competitive
with all but the best programmers at constructing query plans.
c) These systems prove that physical data independence is achievable. Moreover, relational views offer vastly enhanced logical data independence, relative to CODASYL.
d) Set-at-a-time languages offer substantial programmer productivity improvements, relative to record-at-a-time languages.

Relational won, but how?

Commercial Reasons

Lesson Learned

Lesson 7: Set-a-time languages are good, regardless of the data model, since they offer much improved physical data independence.

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

Lesson Learned

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

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

Q&A