The Nature of Data Reverse Engineering

Hainaut, Jean-Luc; Henrard, Jean

Publication date:
2003

The Nature of Data Reverse Engineering

Jean-Luc Hainaut

Facultés universitaires de Namur • Institut d'informatique
Laboratoire d'ingénierie des applications de bases de données

CETIC a.s.b.l.

FNRS Contact day

LLN, May 20, 2003

Organization

1. What is Data Reverse Engineering?
2. The *Implicit construct* problem
3. The main processes of Data Reverse Engineering
4. Data Structure Extraction
5. Data Structure Conceptualization
6. Data Reverse Engineering Tools
7. Effort Quantification
8. Conclusions

1. What is Data Reverse Engineering?

Domain

Legacy Information Systems, [i.e., data-intensive applications, such as business systems based on hundreds or thousands of data files (or tables)], *that significantly resist modifications and changes [Brodie, 1995]*.

Objective of DBRE

To recover the technical and conceptual descriptions of the permanent data of the application, i.e., its database.

- **Technical description**: what are the files, the record types, the fields and their data types, the relationships and the constraints. Expressed in a *Logical schema*.
- **Conceptual description**: what do these data structures mean? Expressed in a *Conceptual schema*.

1. What is Data Reverse Engineering? (2)

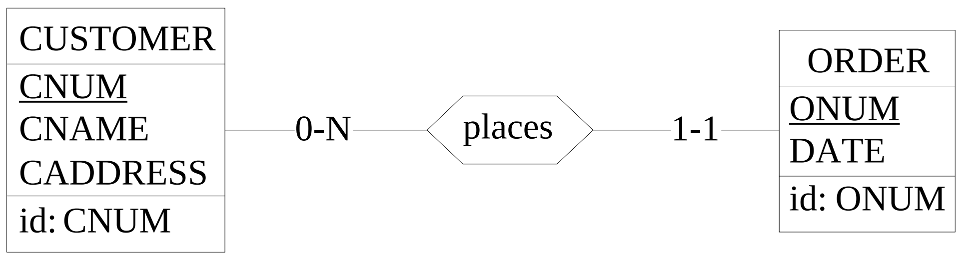
Is Data Reverse Engineering really that difficult?

It’s fairly easy ... in some cases

```
create table CUSTOMER (
        CNUM .. not null,
        CNAME .. not null,
        CADDRESS .. not null,
        primary key (CNUM))
```

```
create table ORDER (
        ONUM .. not null,
        CNUM .. not null,
        DATE .. not null,
        primary key (ONUM),
        foreign key (CNUM)
        references CUSTOMER))
```

⇓



. . . but quite difficult in others

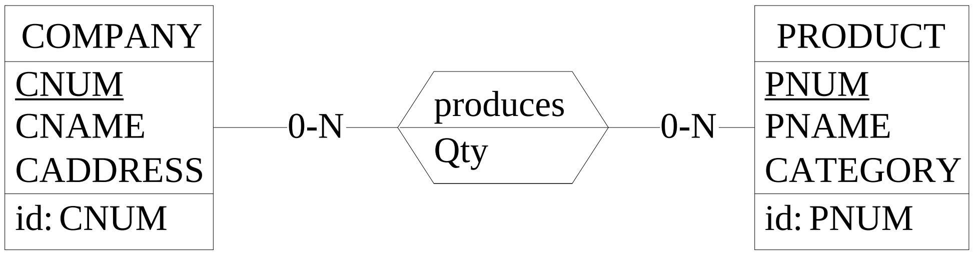
```
select CF008 assign to DSK02:P12,
 organization is indexed,
 record key is K1 of REC-CF008-1.

 select PF0S assign to DSK02:P27,
 organization is indexed,
 record key is K1 of REC-PFOS-1.

 fd CF008;
 record is REC-CF008-1.
 01 REC-CF008-1.
    02 K1 pic 9(6).
    02 filler pic X(125).
```

```
fd PF0S;
records are REC-PF0S-1,REC-PF0S-2.
01 REC-PF0S-1.
   02 K1
      03 K11 pic X(9).
      03 filler pic 9(6)
   02 filler pic X(180).
01 REC-PF0S-2.
   02 filler pic X(35).
```

⇓



1. What is Data Reverse Engineering? (3)

Why Data Reverse Engineering?

Doesn’t seem to be the most exciting engineering activity, but it is a prerequisite for:

- Knowledge acquisition in system development
- System maintenance
- System reengineering
- System extension
- System migration
- System integration
- Quality assessment
- Data extraction/conversion/migration (e.g., to data warehouses)
- Data Administration
- Component reuse

1. What is Data Reverse Engineering? (4)

Data reverse engineering *vs* Program reverse engineering

Two observations

- It is impossible to understand a (business) program until the main data structures have been fully understood.
- It is impossible to fully understand data structures without a clear understanding of the programs that manipulate them.

Objective of *Program* Reverse Engineering

To extract abstractions from the programs in order to understand some of its aspects (= *program understanding*). Recovering full functional specifications still unreachable.

Objective of *Data* reverse Engineering

To recover the (hopefully) complete technical and functional specifications of the data structures.

1. What is Data Reverse Engineering? (5)

Specific DBRE problems

- **Weakness of the DBMS models**: The technical model provided by the DMS can express only a small subset of the structures and constraints of the intended conceptual schema.
- **Implicit structures**: Some constructs have intentionally not been explicitly declared in the DDL specification of the database
- **Optimized structures**: For technical reasons, such as time and/or space optimization, many database structures include non semantic constructs
- **Awkward design**: Not all databases were built by experienced designers. Novice and untrained developers, generally unaware of database theory and database methodology, often produce poor or even wrong structures.
- **Obsolete constructs**: Some parts of a database have been abandoned, and ignored by the current programs.
- **Cross-model influence**: Some relational databases actually are straightforward translations of IMS or CODASYL databases, or of COBOL files.
- **. . . and, of course, no documentation!**

2. The *Implicit construct* problem (1)

Explicit construct (intended structure)

```
create table CUSTOMER (C_ID   integer not null primary key,
                       C_DATA char 80 not null);
create table ORDER    (O_ID   integer not null primary key,
                       OWNER  integer not null
                       foreign key (OWNER) references CUSTOMER);
```

Implicit construct (coded structure)

```
create table CUSTOMER (C-ID   integer not null primary key,
                       C-DATA char(80) not null);
create table ORDER    (O-ID   integer not null primary key,
                       OWNER  integer not null);
...
exec SQL select count(*) in :ERR-NBR from ORDER
         where OWNER not in (select C-ID from CUSTOMER)
end SQL
...
if ERR-NBR > 0 then display ERR-NBR,'referential constraint violation';
```

2. The *Implicit construct* problem (2)

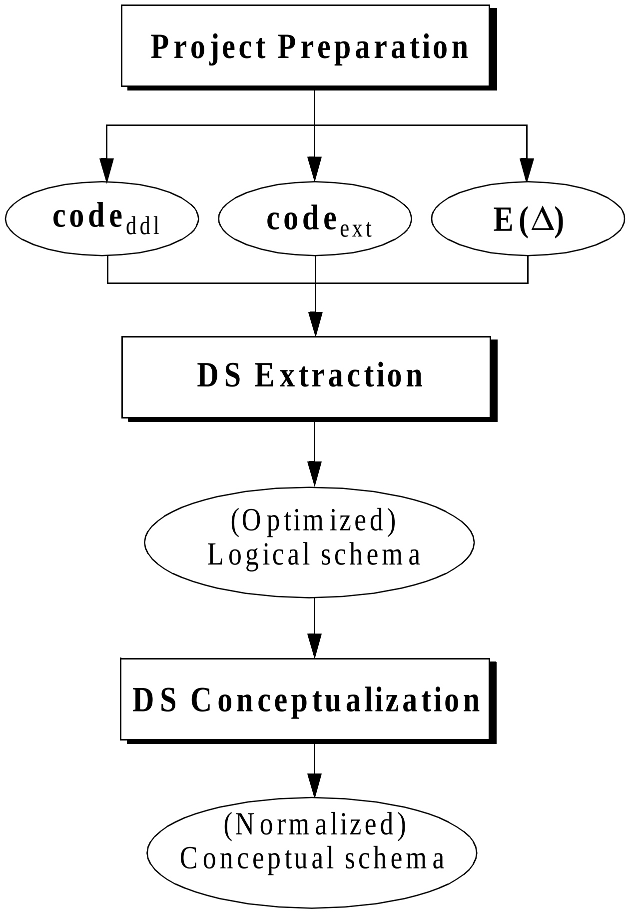
Explicit construct (intended structure)

```
01 CUSTOMER.
    02 C-KEY.
       03 ZIP-CODE pic X(8).
       03 SER-NUM  pic 9(6).
    02 NAME     pic X(15).
    02 ADDRESS  pic X(30).
    02 ACCOUNT  pic 9(12).
```

Implicit construct (coded structure)

```
01 CUSTOMER.
 02 C-KEY  pic X(14).
 02 filler pic X(57).
```

3. The main processes of Data Reverse Engineering



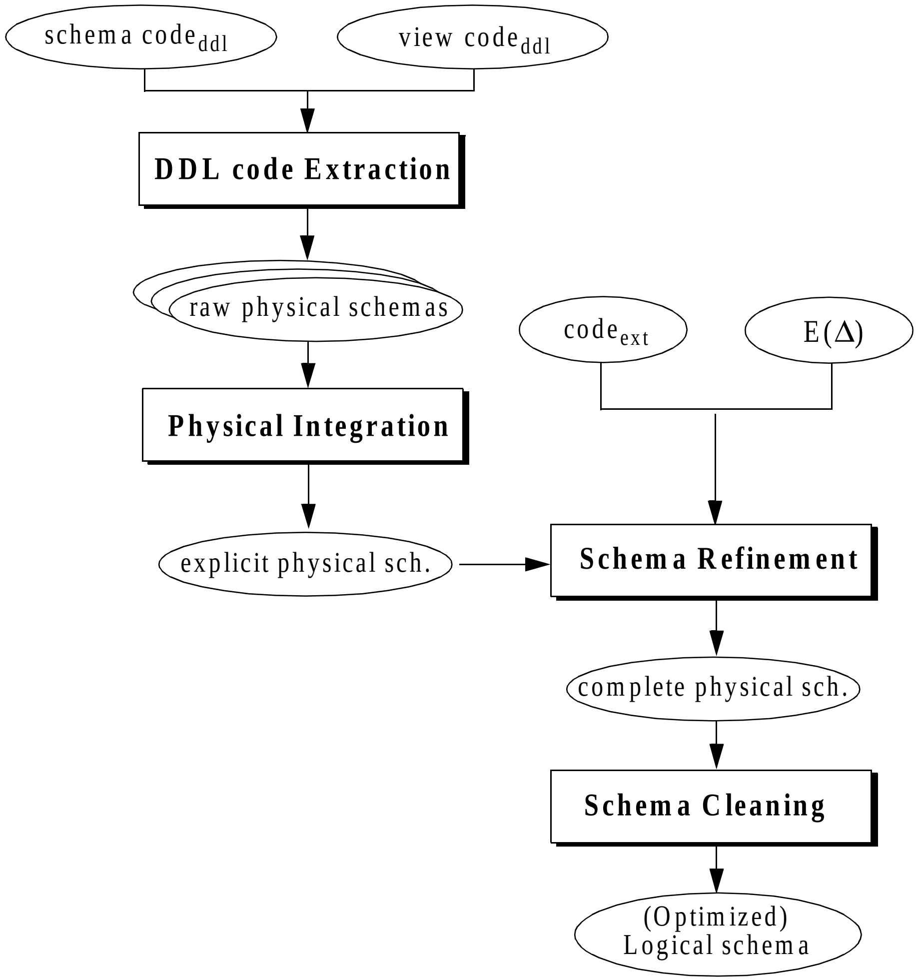
Project Preparation (mainly source inventory):

- *explicit* code (**code**$_{ddl}$)
- code for *implicit* constructs (**code**$_{ext}$)
- other, environmental, sources (**E(Δ)**)

Data Structure Extraction
Recovering the description of the data structures (the *Logical schema*) as seen and used by the programmer (relational, files, IMS, CODASYL, etc.).

Data Structure Conceptualization
Interpreting the data structures in abstract terms pertaining to the application domain (the *Conceptual schema*).

4. Data Structure Extraction



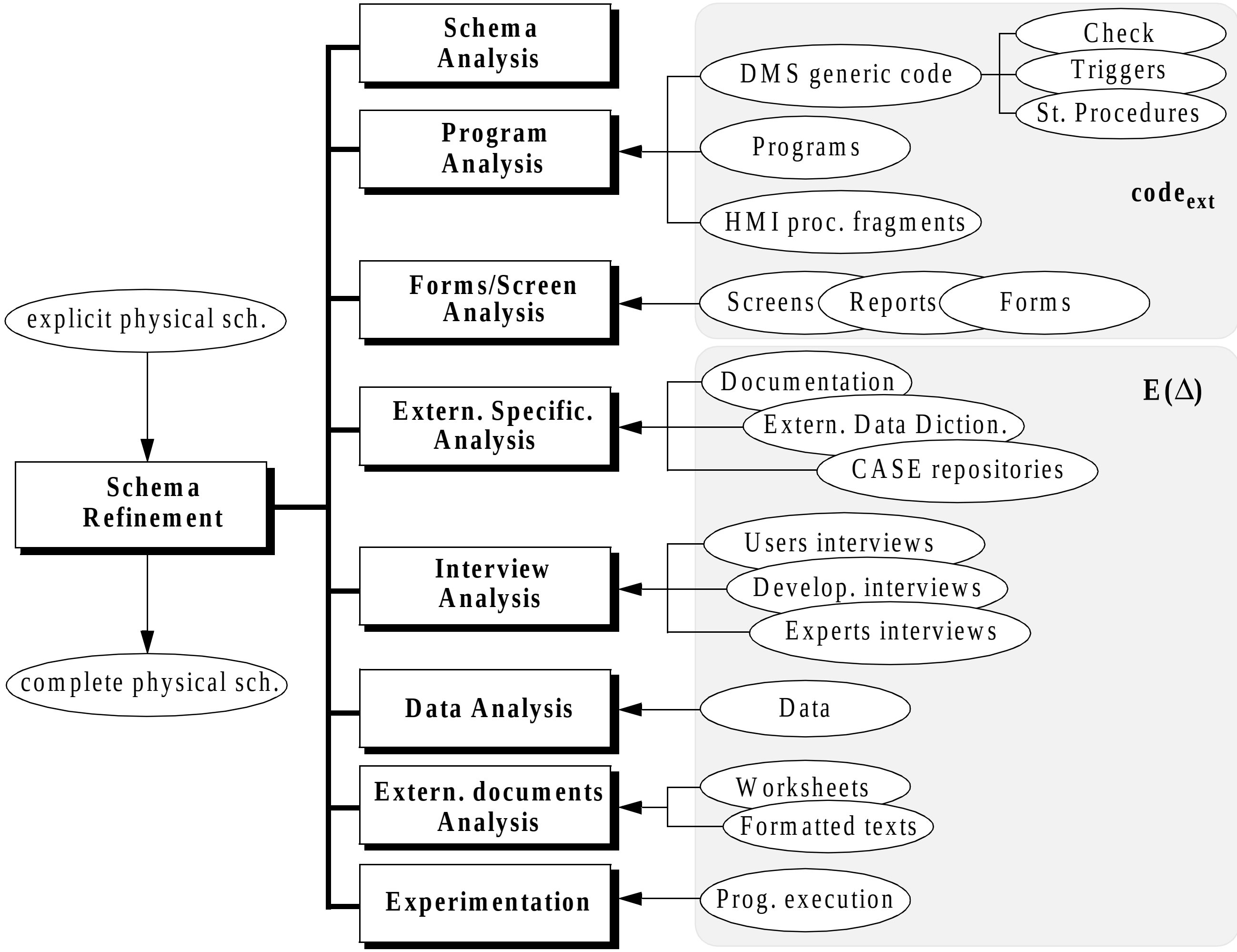
DDL code Extraction:
Automatic parsing of the code to extract **explicit** data structures.

Physical Integration
Merging multiple views of the same data sets.

Schema Refinement
Recovering the **implicit** data structures and constraints.

Schema Cleaning
Removing physical constructs (bearing no semantics).

4. Data Structure Extraction - Schema Refinement



4. Data Structure Extraction - Elicitation techniques

Schema Analysis

- Constructs and constraints can be inferred from existing structural patterns.

Program Analysis

- **Pattern matching**: finding programming *clichés* (they suggest implicit constraint management).
- **Dataflow Analysis**: finding variables that share common values at run time (they could be structurally simalar or semantically related).
- **Program Slicing**: computing the sequence of statements that contribute to the state of an object at a program point, therefore reducing the search space of a programming *cliché*.

Data Analysis

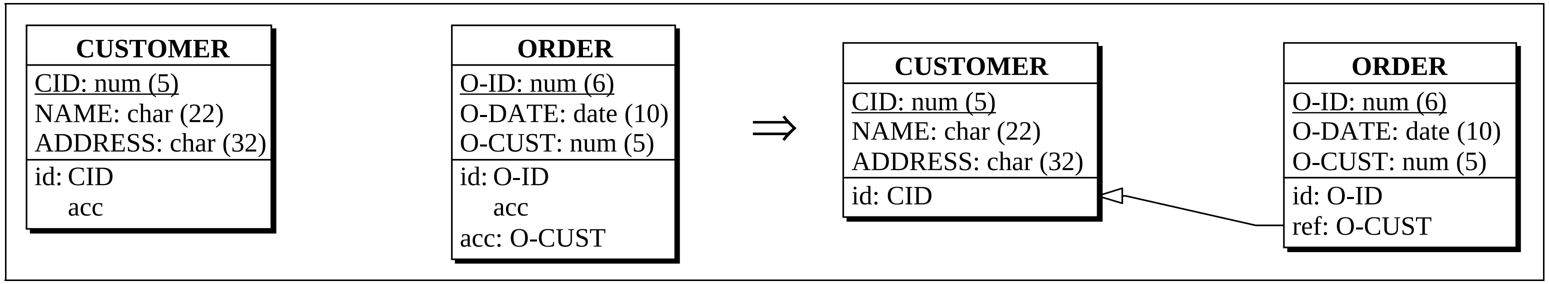
- finding relationships and patterns in a data set; nice to find potential correlations (such as FD).
- evaluating hypotheses: *is this field a foreign key?*

Name Analysis

- Names can suggest roles, data types and relationships between data.

4. Data Structure Extraction - Finding implicit foreign keys (1)

- Implicit FK can be found in all systems, even in SQL, IMS and CODASYL databases.
- **Standard** (RDB-like) *vs* **non standard** (multivalued, alternate, computed, fuzzy, multi-target, conditional, overlapping, embedded, etc.).



Questions

- is ORDER.O-CUST a foreign key to CUSTOMER.CID?
- what are the possible target record types of ORDER.O-CUST?
- what are the possible source record types that target CUSTOMER.CID?
- what are the possible source record types that target CUSTOMER?
- what are the possible target record types of ORDER?

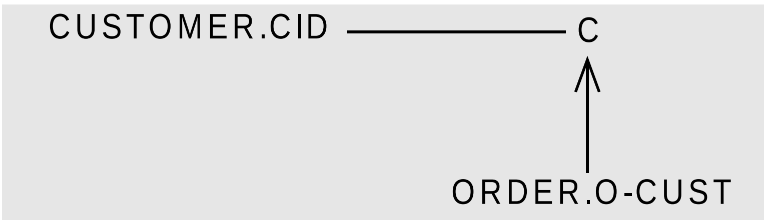
4. Data Structure Extraction - Finding implicit foreign keys (2)

Program Analysis: Dataflow analysis

```
DATA DIVISION.
FILE SECTION.
FD F-CUSTOMER.
  01 CUSTOMER.
    02 CID pic 9(5).
    02 NAME pic X(22).
    02 ADDRESS pic X(32).
FD F-ORDER.
  01 ORDER.
    02 O-ID pic 9(6).
    02 O-DATE pic 9(8).
    02 O-CUST pic 9(5).

WORKING-STORAGE SECTION.
  01 C  pic 9(5).
  01 OI pic 9(6).
```

```
PROCEDURE DIVISION.
  ...
  display "Enter order number "
     with no advancing.
  accept OI.
  move 0 to IND.
  call "SET-FILE" using OI, IND.
  read F-ORDER
     invalid key go to ERROR-1.
  ...
  if IND > 0 then
     move O-CUST of ORDER to C.
  ...
  if C = CID of CUSTOMER then
     read F-CUSTOMER
        invalid key go to ERROR-2.
  ...
```



4. Data Structure Extraction - Finding implicit foreign keys (3)

Program Analysis: *cliché* analysis

```
read-first ORDER(O-CUST=CUSTOMER.CID);
while found do
  process ORDER;
  read-next ORDER(O-CUST=CUSTOMER.CID)
end-while;
```

Schema Analysis

- The **name** O-CUST suggests that of CUSTOMER
- O-CUST and the identifier of CUSTOMER (CID) share the same **type** and the same **length**.
- O-CUST is supported by an **index** (acc).

Data Analysis

```
select count(*)
from   ORDER
where  O-CUST not in (select CID from CUSTOMER)
```

5. Data Structure Conceptualization

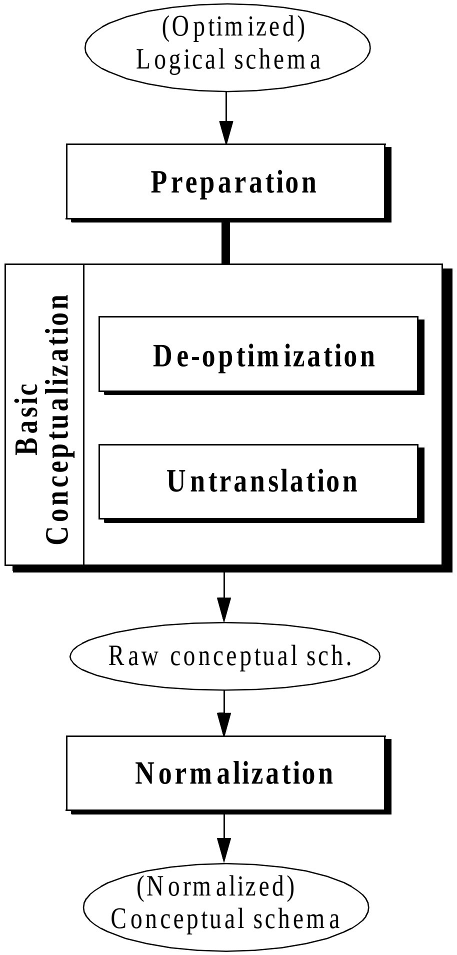
Goal

extracting a conceptual schema from the complete logical schema

challenge: the logical schema is the result of *translation* and *optimization* processes

Also called **data structure interpretation**.

5. Data Structure Conceptualization



Preparation:
Removing dead and technical constructs; renaming.

Basic Conceptualization
Extracting the relevant semantic concepts.

De-optimization
Identifying and transforming optimization constructs.

Untranslation
Retrieving the source conceptual structure of each implementation construct.

Normalization
Reshaping the schema for readability, expressiveness, etc.

5. Data Structure Conceptualization - schema transformations

Transformational view of software engineering

(almost) every software engineering process can be modelled as a chain of specification transformations

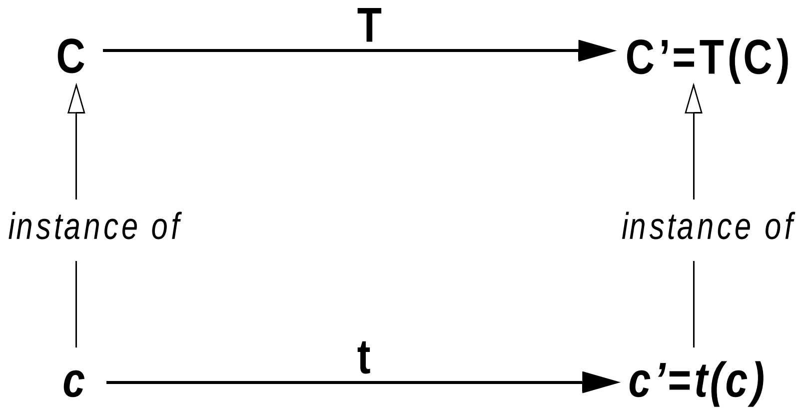
Transformational view of database engineering

(almost) every database engineering process can be modelled as a chain of schema transformations

Application: DS conceptualization ≈ $(\text{DB logical design})^{-1}$

5. Data Structure Conceptualization - schema transformations

A schema transformation Σ is a couple of mapping $<T,t>$, where T is the structural maping (the *syntax* of Σ) and t the instance mapping (the *semantics* of Σ).



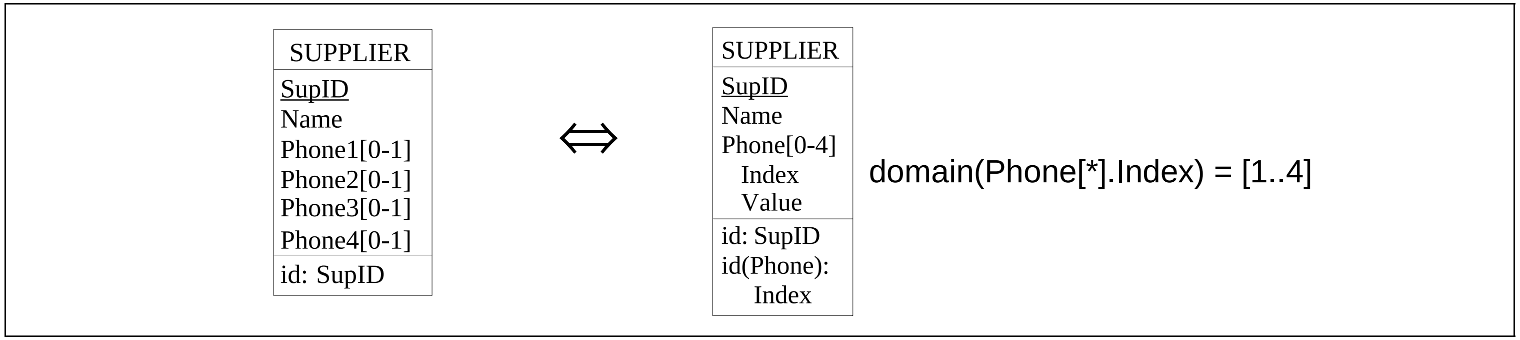
$\Sigma_1 = <T_1,1>$ is **reversible**, or **semantics-preserving**, *iff* there exists a transformation $\Sigma_2 = <T_2,t_2>$ such that, for any construct C and any instance c of C,

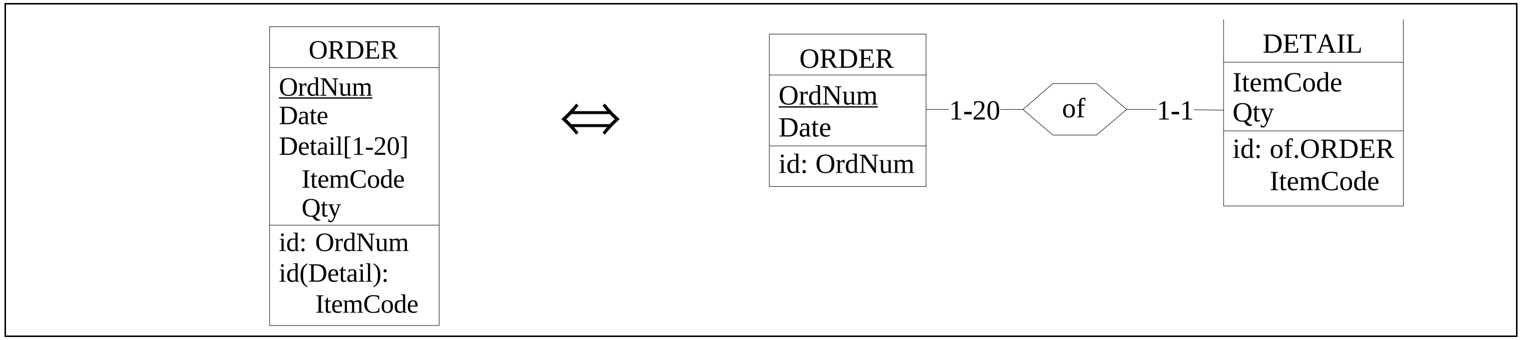
$$C = T_2(T_1(C)) \wedge c = t_2(t_1(c))$$

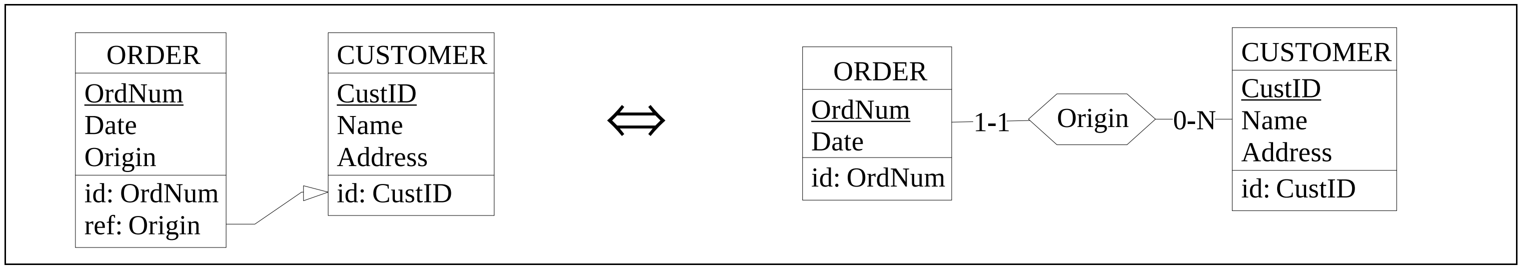
$$C = T_1(T_2(C)) \wedge c = t_1(t_2(c))$$

Reversible transformations are first-class operators, but weaker operators sometimes are necessary

5. Data Structure Conceptualization - schema transformations



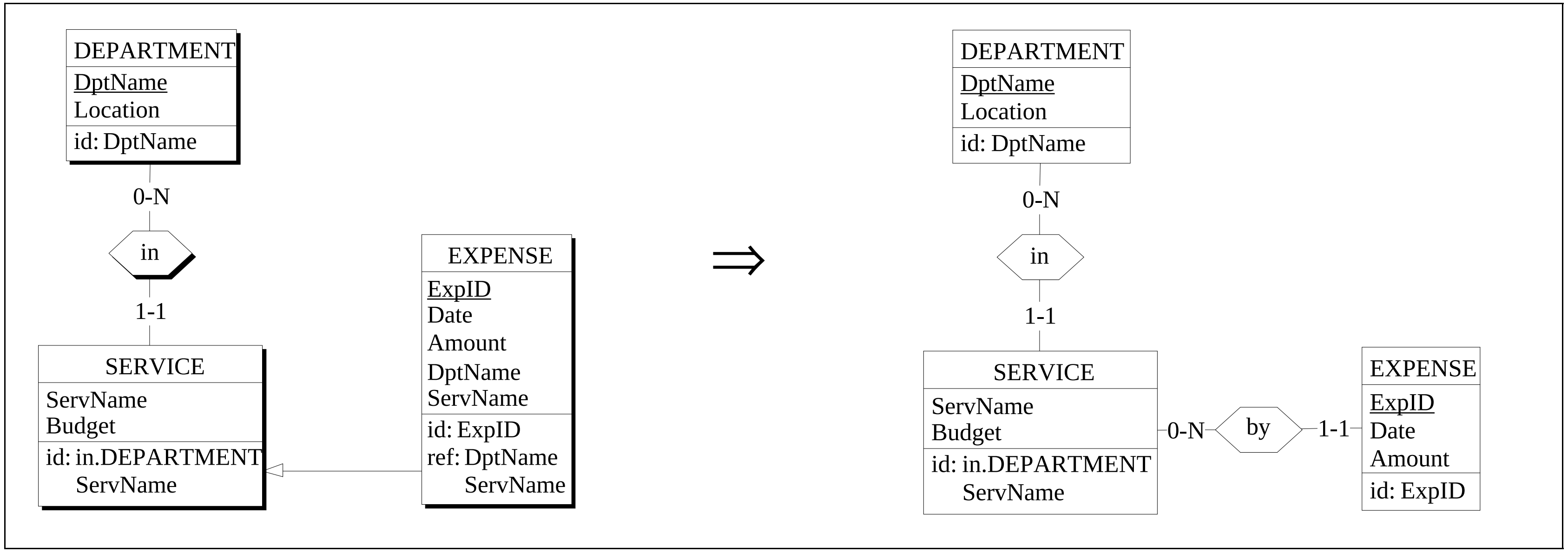




5. Data Structure Conceptualization - interpreting FK (1)

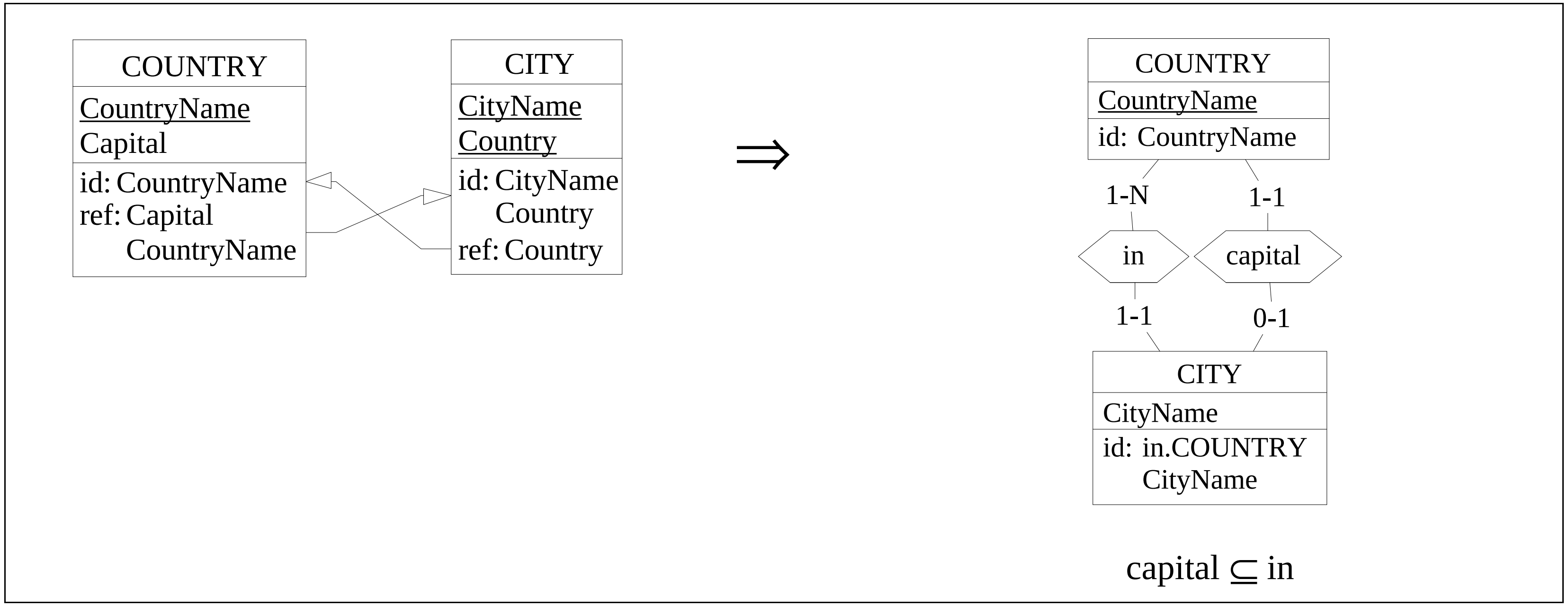
Three classes of non standard foreign keys

A. Hierarchical FK (IMS databases)



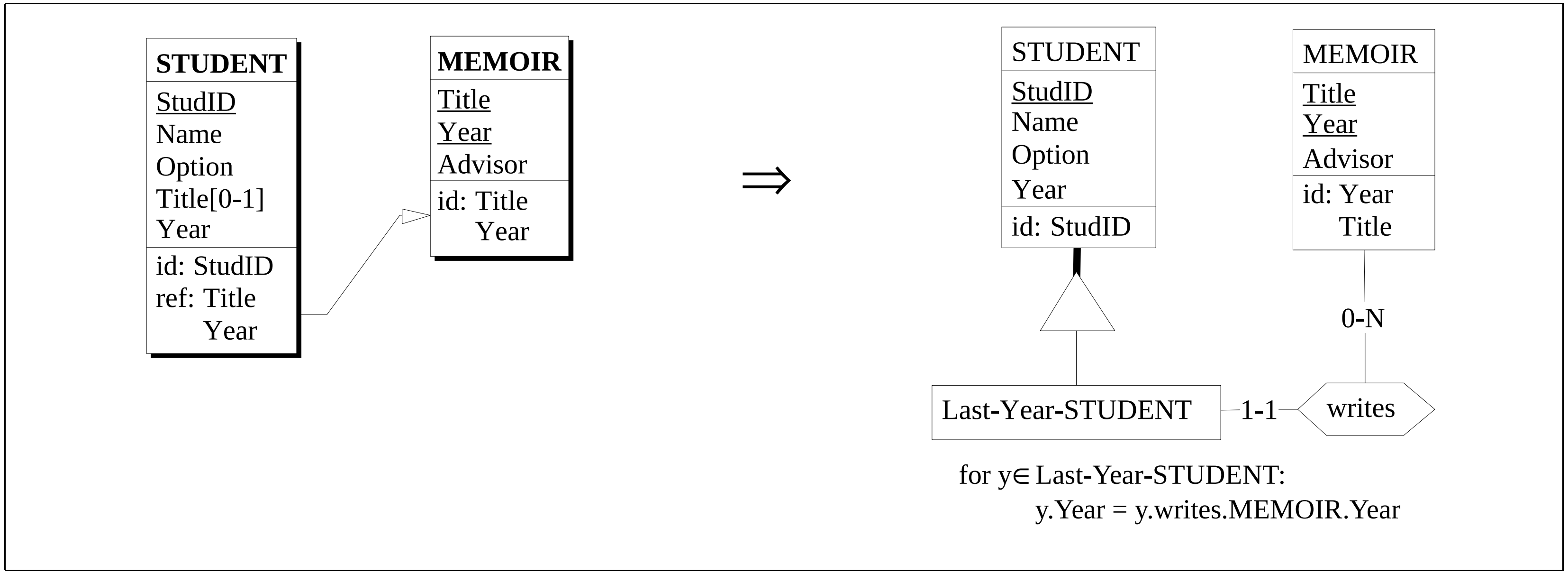
5. Data Structure Conceptualization - interpreting FK (2)

B. Partially reciproqual FK



5. Data Structure Conceptualization - interpreting FK (1)

C. Partially optional FK



6. Data Reverse Engineering Tools (1)

- No specific CARE tools so far (not a drawback anyway).
- Only *limited* DBRE functions in current CASE tools (Power-Designer, AMC-Designor, Rose, Designer 2000, etc):
    - parsers for SQL DB,
    - foreign key elicitation under very strong assumptions (PK and FK have same names and types)
    - standard foreign key transformation.

6. Data Reverse Engineering Tools (2)

The DB-MAIN CASE environment

Project and document representation and management

- **specifications management**: access, browsing, creation, update, copy, analysis, memorizing;
- **representation of the project history**: processes, schemas, views, source texts, reports, generated programs and their relationships;
- a **generic, wide-spectrum, representation model** for conceptual, logical and physical objects; accept both entity-based and object-oriented specifications; schema objects and text lines can be selected, marked, aligned and colored;
- **semantic and technical annotations** can be attached to each specification object;
- **multiple views of the specifications** (4 hypertexts and 2 graphical views); some views are particularly intended for very large schemas; both entity-based and object-oriented schemas can be represented;

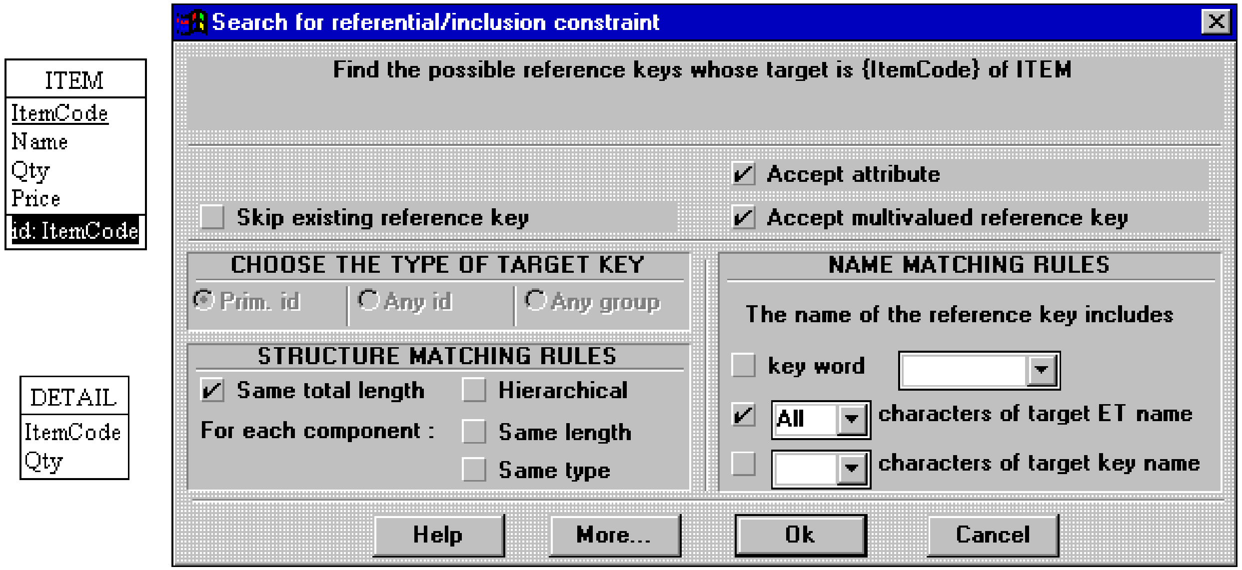
6. Data Reverse Engineering Tools (3)

Support for the data structure extraction process

- **code parsers** for SQL, COBOL, CODASYL, RPG and IMS source programs; other parsers can be developed and plugged into the tool;
- interactive and programmable **text analyzer**;
- **dataflow and dependency diagrams** builder and analyzer;
- **program slicer**;
- **name processor** to search a schema for name patterns;
- programmable **schema analyzer**;
- programmable **foreign key discovery assistant**;

6. Data Reverse Engineering Tools (4)

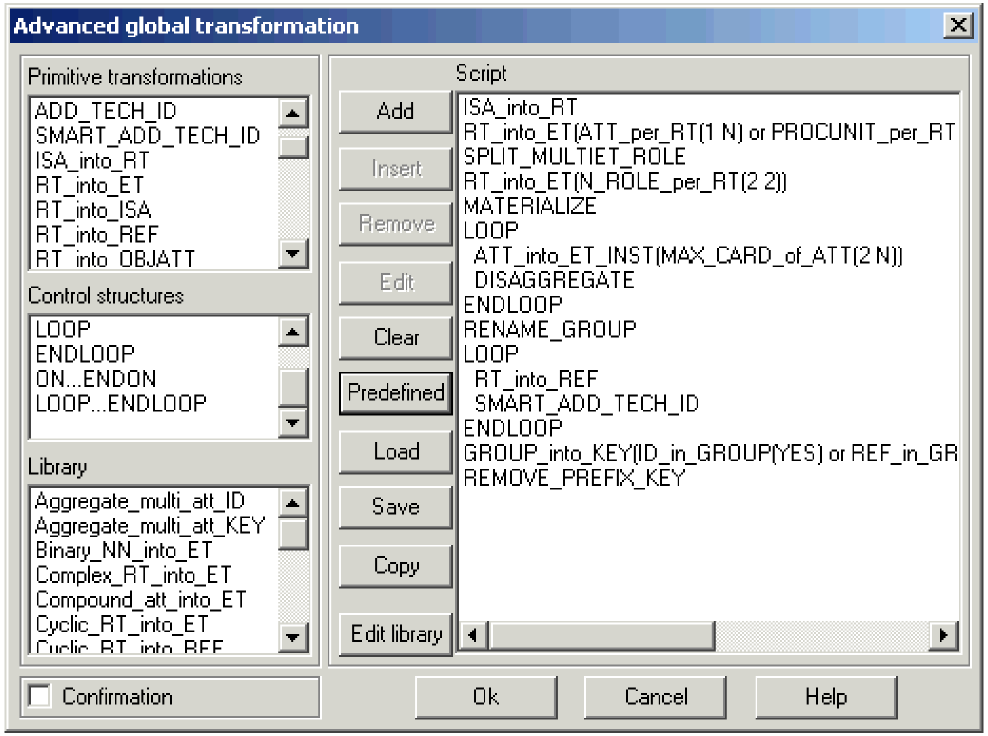
The foreign key discovery assistant (view of the Search engine)



6. Data Reverse Engineering Tools (5)

Support for the data structure conceptualization process

- a toolbox of about 30 semantics-preserving schema transformation;
- name processor to transform names;
- schema integrator;
- programmable schema transformation assistant.



7. Effort Quantification (*tentative*)

Typical database

800 files/tables; 20,000 fields/columns;
(current champion: SAP internal database, with 16,000-30,000 tables; 200,000 columns).

Depends on the objective

Quality assessment:	1 week *
Data extraction:	2 month
Reengineering:	6 months.

Depends on the quality of the source

Well documented, normalized relational database :	C
Undocumented, poorly designed legacy IMS database :	5 x C
Undocumented, poorly designed COBOL files :	10 x C

Example: recovering 200 implicit foreign keys in a *Part inventory* IMS DB = 60 work. days.

* Blaha, M., The Case for Reverse Engineering, *IEEE IT Professional*, March-April, 1999

8. Conclusions (*tentative too*)

What is available

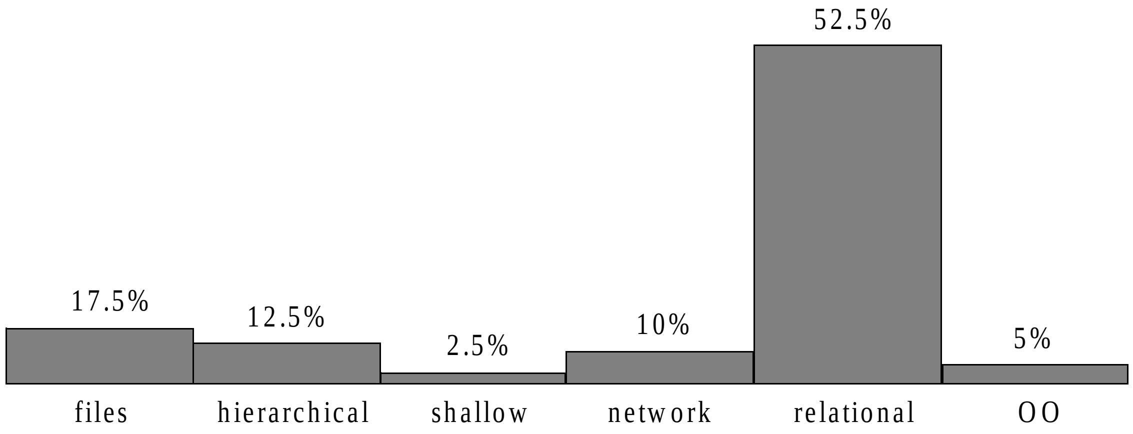
- Most problems are identified
- Many elicitation techniques (nice for micro-problems, inadequate for large scale projects)
- Heuristics
- Popular but limited CARE functions (in standard CASE tools)
- Proprietary and unpublished powerful analysis tools

What remains to be done

- Sensitizing practioners: Data RE is useful and is practicable
- Sensitizing practioners: Data RE can be expensive
- Training
- Developing popular **and** powerful CARE tools
- Improving tool and method scalability
- Refining heuristics (less noise, fewer missing construicts)
- Generalizing to system level problem: how to reverse engineer the whole IS?
- Developing techniques for reengineering legacy systems into distributed components architectures (so far, DB $\rightarrow$ OO techniques disappointing).

8. Conclusions (*2*)

- Addressing less *sexy* but much more critical problems: COBOL applications, IMS, CODASYL, RPG, Business Basic.



Distribution of 40 recent research publications according to the DMS model (2000)

Introductory reference

Hainaut, J.-L., *Database Reverse Engineering*, 5th edition, LIBD research report, Namur, 2002, 150 p.; available at `http://www.info.fundp.ac.be/libd` > `Documents` > `Publications` > `Books`