A Cognitive Approach to the Reconstruction of ER Schema from Database Applications

Oreste Signore

Mario Loffredo - Mauro Gregori - Marco Cima

ISCIS IX

CNUCE

36 VIA S. MARIA – 56126 – PISA
Tel. (050) 593111 Telex 500371 – CNUCE
Fax (050) 904052 – Telegrammi CNUCE – PISA

Using Procedural Patterns in Abstracting Relational Schemata

Oreste Signore
Mario Loffredo - Mauro Gregori - Marco Cima

SEAL (Software Engineering and Applications Laboratory)
CNUCE - Institute of CNR - via S. Maria, 36 - 56126 Pisa (Italy)
Phone: +39 (50) 593201 - FAX: +39 (50) 904052
E.mail: oreste@vm.cnuce.cnr.it

WPC'94

3rd Workshop on Program Comprehension
November 14-15, 1994
Washington D.C.

CNUCE
36 VIA S. MARIA – 56126 – PISA
Tel. (050) 593111 Telex 500371 – CNUCE
Fax (050) 904052 – Telegrammi CNUCE – PISA

CNUCE

36 VIA S. MARIA - 56126 - PISA
Tel. (050) 593111 Telex 500371 - CNUCE
Fax (050) 904052 - Telegrammi CNUCE - PISA

Reconstruction of ER Schema from Database Applications: a Cognitive Approach

Oreste Signore
Mario Loffredo - Mauro Gregori - Marco Cima

SEAL (Software Engineering and Applications Laboratory)
CNUCE - Institute of CNR - via S. Maria, 36 - 56126 Pisa (Italy)
Phone: +39 (50) 593201 - FAX: +39 (50) 904052
E.mail: oreste@vm.cnuce.cnr.it

ER '94
The Thirteenth International Conference
on

CNUCE
36 VIA S. MARIA – 56126 – PISA
Tel. (050) 593111 Telex 500371 – CNUCE
Fax (050) 904052 – Telegrammi CNUCE – PISA

The Entity-Relationship Approach

Business Modelling and Re-Engineering

December 13-16 1994

Manchester, UK

CNUCE

36 VIA S. MARIA - 56126 - PISA
Tel. (050) 593111 Telex 500371 - CNUCE
Fax (050) 904052 - Telegrammi CNUCE - PISA

Contents

p **Data Base Reverse Engineering**

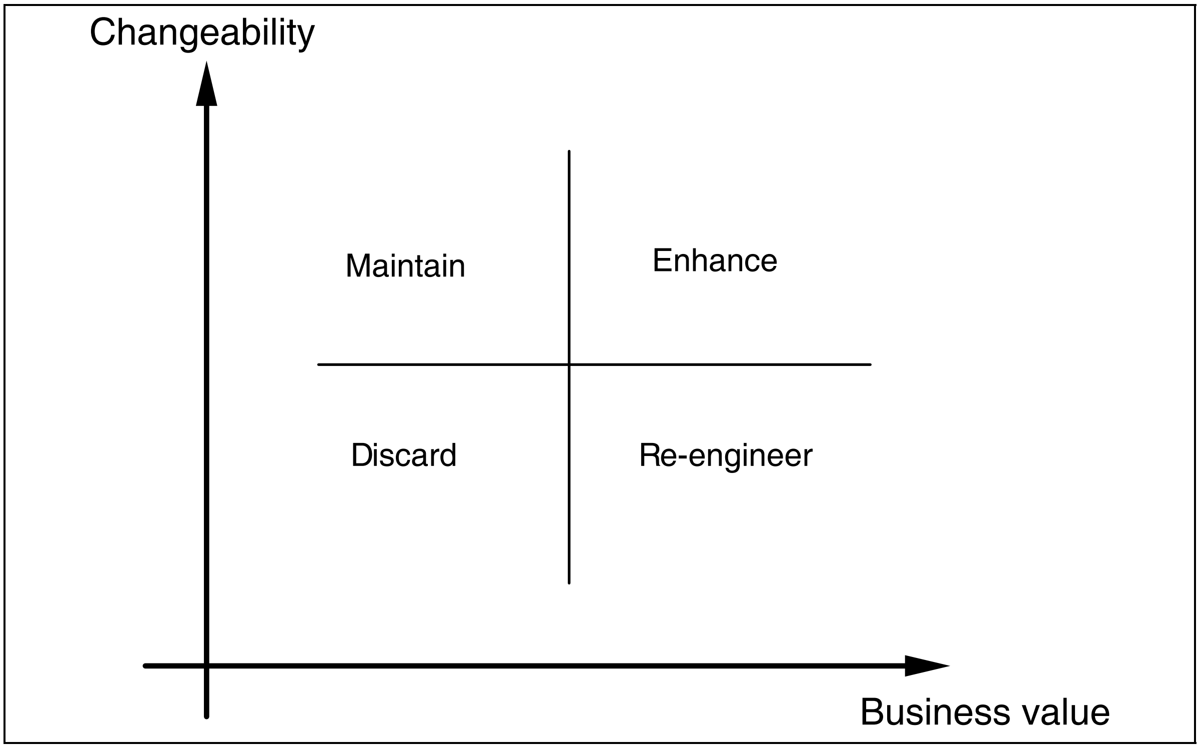
- **motivations**
- **related work**

p **The proposed methodology**

- **overall architecture**
- **three phases**
- **the “clued” approach**
- **the indicators’ matrix**

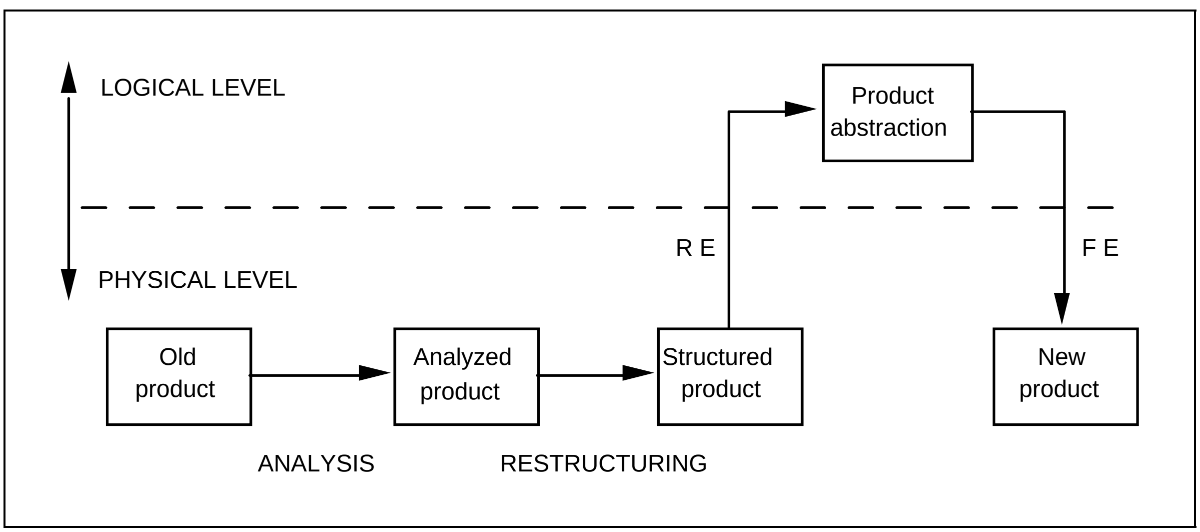
p **Conclusions**

Maintenance and Re-Engineering



2 **Maintenance (corrective, adaptive or perfective):**

- **up to 95% of EDP department activity**
- **we must understand the program semantics and the basic design issues**



Database Reverse Engineering: WHY?

- p **More recent DBMS have new features**
  - constraints can be defined at schema level

- p **Reverse Engineering towards Object-Oriented**

- p **Database applications are very often of crucial importance**

- p **Recovering design issues**

- p **The DBRE is faced with the following problem:**
  - ***given*** **the DDL/host language expression of existing data structures (global, schema and/or views)**
  - ***given*** **known operational requirements (e.g. the DMS performance requirements, etc.)**
  - ***find*** **a possible conceptual schema that could lead to these data structures**

Database Forward Engineering

p **We must well understand the FE to perform an effective reverse engineering process**

p **In FE we have several phases:**

- **mapping conceptual-logical**
- **optimisation of the logical schema**
- **mapping logical-physical**
- **translation of not directly supported specifications**

p **The sequence of transformations induces a progressive degradation of the schema, that becomes:**

- **less complete**
- **less simple**
- **less readable**
- **less expressive**

Database Reverse Engineering: related work

- ## Restrictive hypotheses:
  - **requirements completely mapped onto data structures and constraints**
  - **strict application of the mapping rules**
  - **user needs or environment constraints didn't force any further restructuring of the schema**
  - **existence of a "naming policy"**

- ## Batini, Ceri, Navathe

  (Batini C., Ceri S., Navathe S.B.: *Conceptual Database Design: An Entity-Relationship Approach*,The Benjamin/Cummings Publishing Company,Inc., 1992)
  - **simple and limited process**
  - **a suitable initial model**
  - **clear and linear description of the steps to follow to analyse relations and identify the concepts**
  - **a good semantic knowledge of the initial relational schema is supposed**

- ## Premerlani, Blaha

  (Premerlani W.J., Blaha M.R.: *An Approach for Reverse Engineering of Relational Databases*, Proceedings IEEE Working Conference on Reverse Engineering, Baltimore 1993) + CACM
  - **"experimental" point of view**
  - **set of methods, techniques and practical examples**
  - **large set of real cases**

Database Reverse Engineering: related work (cont'd)

p **Chiang, Barron, Storey**

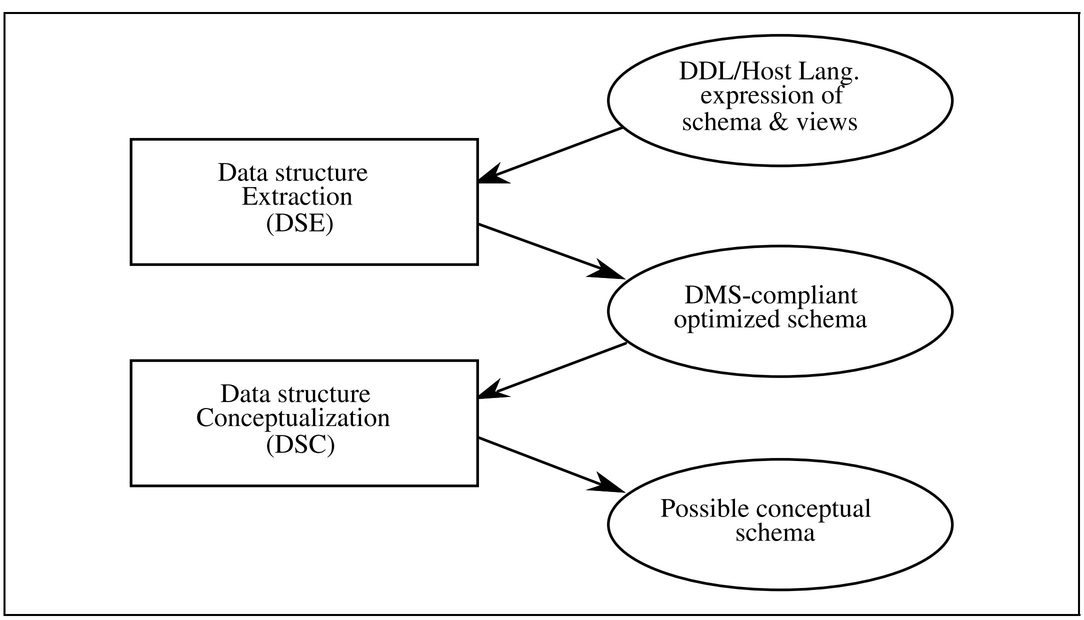
(Chiang R.H.L., Barron T.M., Storey V.C.: *Reverse engineering of relational databases: Extraction of an EER model from a relational database*, Data & Knowledge Engineeering, Vol. 12, N. 2 (March 1994))

- **takes information from the catalog and from the data stored in the relations**

p **Hainaut, Chandelon, Tonneau, Joris**

Hainaut J-L., Chandelon M., Tonneau C., Joris M.: *Contribution to a Theory of Database Reverse Engineering*,, Proceedings IEEE Working Conference on Reverse Engineering, Baltimore 1993

- **we can split the solving process in two main subsequent phases:**
  - ***Data Structure Extraction (DSE)***
    (the reverse of the physical phase)
  - ***Data Structure Conceptualisation (DSC)***
    (the reverse of the logical phase)



The DBRE process

Some problems in Data Base Reverse Engineering

p **In the following databases, try to identify:**

- **domains' identity**
- **IS-A hierarchies**
- **associative relations**
- **attributive relations**

```
BOOKS (ID, TITLE, MAIN_AUT, PUBLISHER,…)
AUTHORS (ID, NAME,…)
SEC_AUTH (BOOK_ID, AUTH_ID)
STUDENT (ID, FSTNAME, LSTNAME, COURSE,…)
LOAN (ID, STUD_ID, BOOK_ID, DATE,...)

EMPLOYEE (EID, D1,…,Dn)
MANAGERS (EID, M1,…,Mp)
TECHNICIANS (EID, T1,…,Tq)
SECRETARIES (EID, S1,…,Sr)
SKILL(EID, SKILL, LEVEL)
ENGAGED (EID, PROJECT, PERCENTAGE)
PROJECT (P#, TITLE)
```

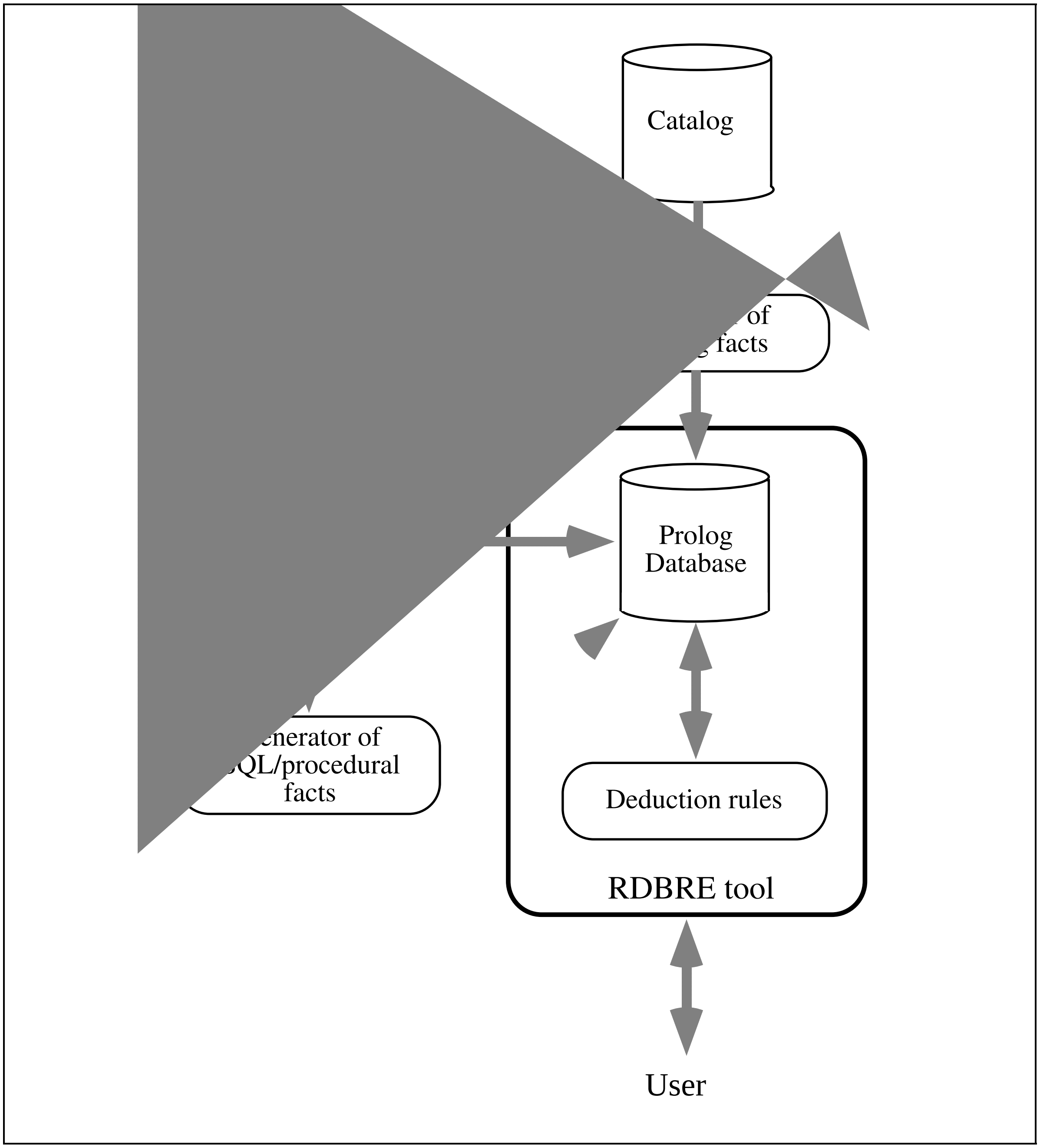
p We cannot simply rely on column names.

p To capture the semantics, we must consider how the applications make use of the data

Architecture

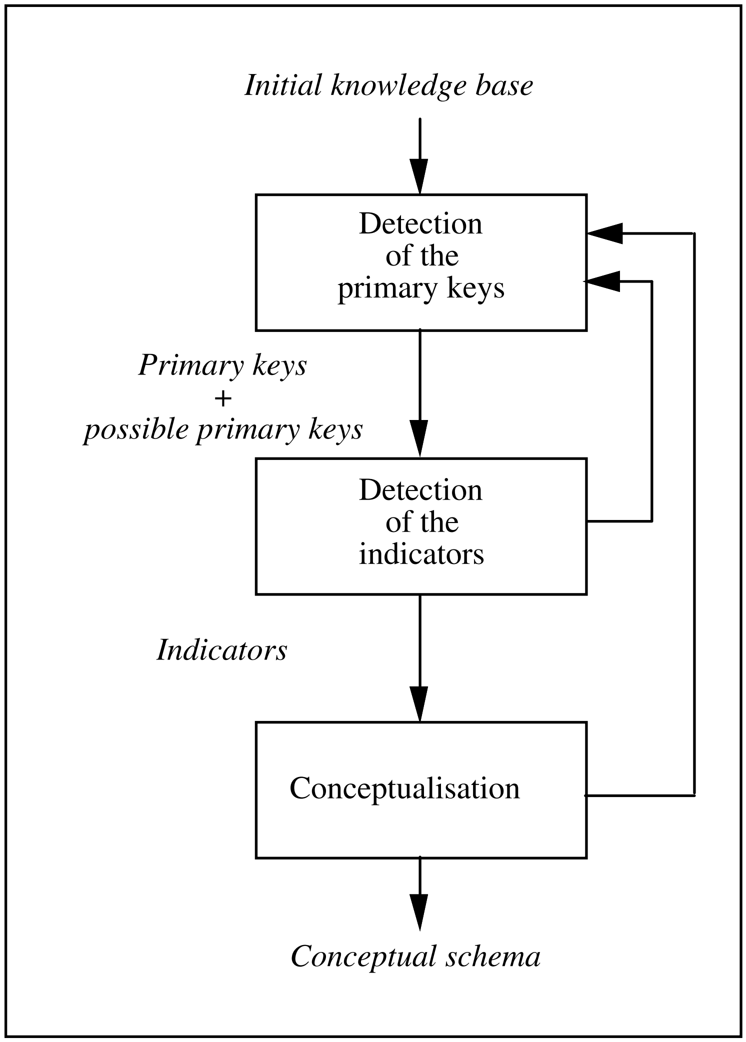
p **Implicit assumptions:**

- **a first phase of generation of SQL/procedural facts**
- **a second phase of generation of catalog facts**



Architecture of the RDBRE tool

The RDBRE process



The DBRE process phases

- Phase 1: Identification of the primary keys
- Phase 2: Detection of the indicators
- Phase 3: Conceptualisation

The indicators

p **Definition of an indicator**

a set of information detectable from one or more available sources (catalog, SQL code, output of a previous analysis phase), that could characterise, in the conceptual model, one or more relational schema items

p **Classes of indicators:**

- schema indicators

  taken from the catalog and the information deduced in the key identification phase

- key indicators

  taken from the analysis of the primary keys

  help in defining the properties of the PK of a given relation

- SQL indicators

  taken from the analysis of the SQL commands

  give information about the kind of usage the DML statements make of the table elements

- procedural indicators

  taken from the analysis of the host language code

  integrate the information supplied by the SQL indicators: made of some typical (standard) patterns for conditional manipulation of the database data

  Examples:

  - *fetch loops*
  - *referential integrity constraints' checking*
  - *actions on tables implementing class hierarchies*

Identification of the primary keys: generalities

p **A trivial case if explicitly defined**

p **If we have** only one index **with the** `UNIQUE` **option**

PK can be identified as the attribute (or attribute set) the index is defined upon

p **If we have** more than one index **with the** `UNIQUE` **option**

- **we consider every set as a candidate key**
- **we calculate the frequencies of usage**
- **we ask the user to make a choice**

p **If we do not succeed in identifying a primary or candidate key:**
we can identify some *indicators* by analysing procedural patterns:

- **at least one** `WHERE` **clause must mention all the columns composing the potential key** (a)
- **no DML statement making use of these columns and returning a set of tuples should exist** (b-g)

Identification of the primary keys: the SQL patterns

<table>
<tr><th></th><th>Pattern</th></tr>
<tr><td>a</td><td>WHERE a1=<scalar_exp1> AND…AND as=<scalar_exps></td></tr>
<tr><td>b</td><td>No declaration of a cursor like:
DECLARE <cursor_id> FOR
SELECT <selection>
FROM T
WHERE a1=<scalar_exp1> AND…AND as=<scalar_exps>
followed by
OPEN <cursor_id>
and a loop containing:
FETCH <cursor_id> INTO <list_of_host_var>
or:
No assignment of the selected tuples to an array.</td></tr>
<tr><td>c</td><td>No statement contains:
SELECT ALL|DISTINCT <selection>
FROM T
WHERE a_1=<scalar_exp$_1$> AND…AND a_s=<scalar_exp$_s$></td></tr>
<tr><td>d</td><td>No statement contains:
SELECT <function-ref>
FROM T
WHERE a_1=<scalar_exp$_1$> AND…AND a_s=<scalar_exp$_s$>
where
function-ref::= COUNT(*) | distinct-function-ref | all-function-ref
distinct-function-ref::={AVG|MAX|MIN|SUM|COUNT}(DISTINCT column-ref)
all-function-ref::= {AVG|MAX|MIN|SUM|COUNT}([ALL]scalar-exp)</td></tr>
<tr><td>e</td><td>No statement contains:
SELECT <selection>
FROM T
WHERE a_1=<scalar_exp$_1$> AND…AND a_s=<scalar_exp$_s$>
GROUP BY <column-ref-commalist>
or
SELECT <selection>
FROM T
WHERE a_1=<scalar_exp$_1$> AND…AND a_s=<scalar_exp$_s$>
ORDER BY <ordering-ref-commalist></td></tr>
<tr><td>f</td><td>No statement contains:
SELECT <selection>
FROM T
GROUP BY a_1, a_2,…, a_s</td></tr>
<tr><td>g</td><td>No statement contains:
WHERE <scalar-exp> [NOT] IN <subquery>
or
WHERE <scalar-exp><comparison> ALL|ANY|SOME <subquery>
where <subquery> is like
SELECT <selection>
FROM T
WHERE a_1=<scalar_exp$_1$> AND…AND a_s=<scalar_exp$_s$></td></tr>
</table>

Second phase:
indicators' detection

p **First phase:**

- **PK identified**
- **hypotheses about PK formulated**
- **existence of candidate keys indicated**

p **Second phase:**

- **face the difficulties arising from the different semantic richness of ER and relational model**
- **we must consider:**
  - *mapping from an ER to a relational model is not unique*
  - *optimisation choices*
  - *poorness of the DDL*
  - *unusual implementation techniques*
- **we must adopt a "clued" approach**
  (a conceptualisation phase will follow)
- **the steps:**
  - *domains' identification*
  - *FK's identification*
  - *detection of integrity constraints*
  - *analysis of integrity constraints*

p **Third phase (conceptualisation)**

suitable combinations of indicators can lead to the identification of "probable concepts"

Domains' identification

p **No ambiguities, thanks to the usage of the extended name:**

```
tablename.attributename
```

p **Identification of the attributes defined on the same domain (synonyms):**

- **we can't rely on identical types as defined in the catalog:**
  (SQL type checking is weak!)

  Es.
  Given the relations:

  ```
  BOOKS (ID, TITLE, MAIN_AUT, PUBLISHER,…)
  AUTHORS (ID, NAME,…)
  SEC_AUTH (BOOK_ID, AUTH_ID)
  STUDENT (ID, FSTNAME, LSTNAME, COURSE,…)
  LOAN (ID, STUD_ID, BOOK_ID, DATE,...)
  ```

  only a query like:

  ```
  SELECT       NAME
  FROM  AUTHORS, BOOKS
  WHERE AUTHORS.ID=BOOKS.MAIN_AUT AND BOOKS.PUBLISHER='X'
  ```

  or:

  ```
  SELECT       NAME
  FROM  AUTHORS
  WHERE ID      IN
        (SELECT MAIN_AUT
                FROM   BOOKS
                WHERE  PUBLISHER = 'X')
  ```

  can show that:

AUTHORS.ID, BOOKS.AUT	synonyms
BOOKS.ID, STUDENT.ID, LOAN.ID, AUTHORS.ID	**not synonyms**

Domains' identification (cont'd)

- **some typical patterns**

<table>
<tr><th>Type</th><th>Pattern</th></tr>
<tr><td>equijoin</td><td>SELECT …
FROM T_1, T_2
WHERE …T_1.ATTR = T_2.ATTR'…</td></tr>
<tr><td>multiple join</td><td>SELECT …
FROM T_1, T_2, T_3,…
WHERE …T_1.ATTR$^{(1)}$=T_2.ATTR$_1^{(2)}$ AND
T_2.ATTR$_2^{(2)}$=T_3.ATTR$^{(3)}$…</td></tr>
<tr><td>nested queries</td><td>SELECT …
FROM T_1, …
WHERE …T_1.ATTR [NOT] IN (SELECT T_2.ATTR'
FROM T_2,…
WHERE …)
or:
WHERE …T_1.ATTR{=|•} (SELECT T_2.ATTR'
FROM T_2,…
WHERE …)</td></tr>
<tr><td>auto-join</td><td>SELECT A.STAFF_ID
FROM STAFF A, STAFF B
WHERE A.SALARY > B.SALARY AND A.SUPERVISOR =
B.STAFF.ID</td></tr>
</table>

- **some other cases of semantic equivalence:**

```
INSERT INTO <table> (<column-commalist>)
SELECT <selection-commalist>
<table-exp>.
```

(semantic equivalence of the corresponding attributes in `<column-commalist>` *and* `<selection-commalist>`*)*

- **the usage of host variables induces some additional complexity** *(data dependences must be detected)*

```
SELECT NAME
FROM AUTHORS A, BOOKS B
WHERE A.ID = B.AUT AND B.TITLE = :book
```

is equivalent to:

```
SELECT AUT
INTO :aut_code
FROM BOOKS
WHERE BOOKS.TITLE = :book
        . . .
SELECT NAME
INTO :aut_name
FROM AUTHORS
WHERE ID = :aut_code
```

Foreign Keys

p **Three steps**

a) **Annotate explicitly defined FK**
A trivial case

b) **Identification of not explicitly defined FKs**
Given a relation T, having a primary key PK, we select the synonyms of PK that all belongs to a relation T'.
They are the components of a FK, defined in T', that references T.

c) **Identification of the FKs that refer an uncertain PK**
For all the relations that only have a Possible Primary Key (PPK) we apply the same procedure.
The result is affected by the same uncertainty that affects the PPK.

Referential integrity constraints

p **Identifying the referential integrity constraints checking embedded in the code can help in validating the ambiguous cases.**

p **Referential integrity constraints checking:**
- **in less recent DBMSs was a programmers' task**
- **in more recent DBMSs can be defined at the schema level (triggers)**

p **Identifying the procedural patterns that implement the constraints' checking can be of valuable help in re-engineering phase**
(clean up of the code, homogeneity, etc.)

p **An example**
(procedural pattern to assure the referential integrity when inserting a tuple in the referencing table)

```
PROFESSORS (LSTNAME, FRSTNAME, BIRTHDATE,
ADDRESS,…)
COURSES (COURSE_ID, CLASSROOM,PROF_LSTNAME,
         PROF_FRSTNAME,PROF_BIRTHDATE,…)

EXEC SQL BEGIN TRANSACTION;
EXEC SQL
  SELECT *
  FROM PROFESSORS
  WHERE LSTNAME = :prof_lstname AND
       FRSTNAME = :prof_frstname AND
       BIRTHDATE = :date;
if (SQLCODE == 0)
  {
    EXEC SQL
      INSERT INTO COURSES (COURSE_ID,
       PROF_LSTNAME,PROF_FRSTNAME,
       PROF_BIRTHDATE)
      VALUES (:course, :prof_lstname,:prof_frstname
              :date);
    EXEC SQL COMMIT WORK;
```

```
  }
else <call of the error_handling routine>
```

Referential integrity constraints: analysis

- Identifying FKs and referential integrity constraints can help in recognising the relationships.

- The procedural indicators can be used to:
  - identify the FKs
  - confirm or reject the hypotheses

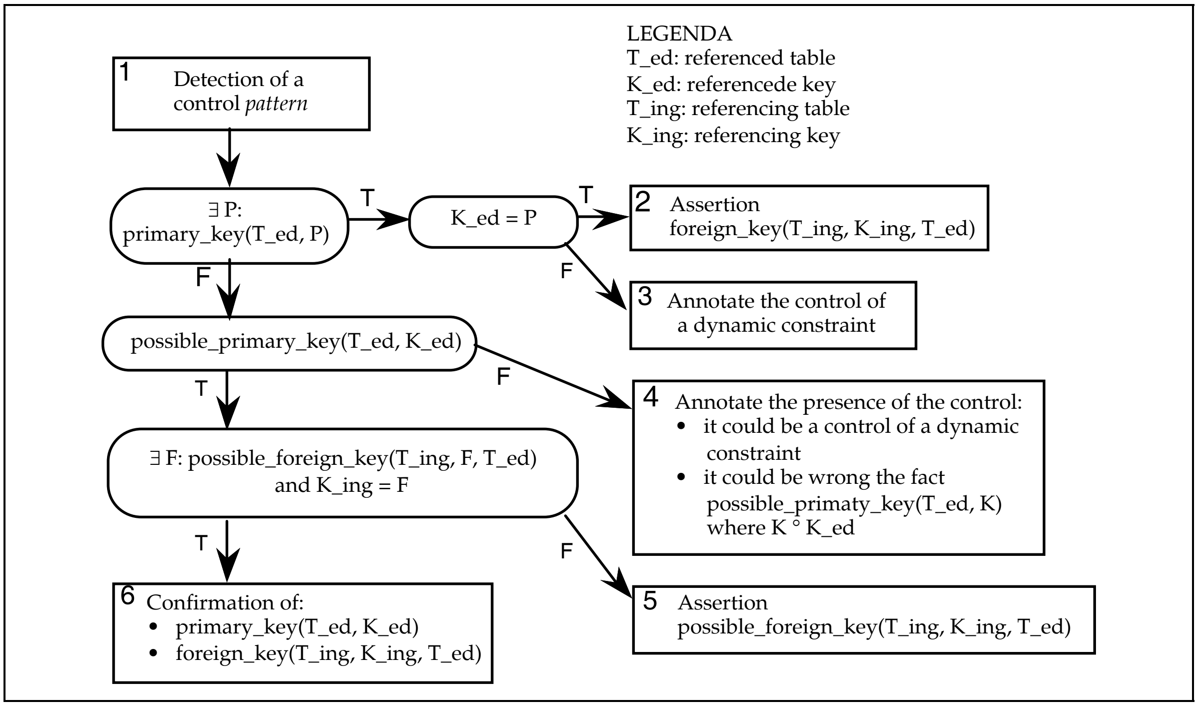
- Some patterns are very similar:

```
CUSTOMERS (CUSTOMER_ID, COMPANY, COUNTRY,…)
AGENTS (AGENT_ID, …, ZONE).

EXEC SQL BEGIN TRANSACTION;
EXEC SQL
  SELECT *
  FROM CUSTOMERS
  WHERE COUNTRY = :zone;
if (SQLCODE == 0)
  {
    EXEC SQL
      INSERT INTO AGENTS (AGENT_ID,…, ZONE)
      VALUES (:agent,…, :zone);
    EXEC SQL COMMIT WORK;
  }
else <call of the error_handling routine>
```

*this pattern implements a **constraint,** but not a **referential integrity constraint**, as the existence check is performed on a non-key field*

Referential integrity constraints: the checking algorithm



Third phase: conceptualisation

p **A simple and extensible paradigm:**

the indicators' matrix

- **rows correspond to ER concepts**
  (with or without direct mapping)

- **columns corresponds to indicators' categories**

- **every cell C_{ij} contains the indicator of $Class_j$, that can be used for the identification of the $Concept_i$**

p **Preceding phases populate the cells making use of:**

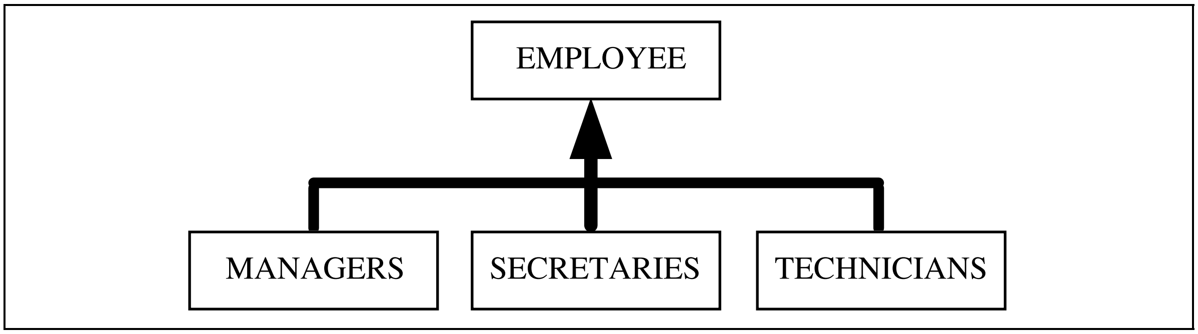
- **Models and mapping rules knowledge**
- **practical knowledge deduced from the implementation experience**

p Quality and quantity of the indicators affect the concepts' identification.

The indicators' matrix

The indicators' matrix

The IS-A hierarchies



Conceptual schema

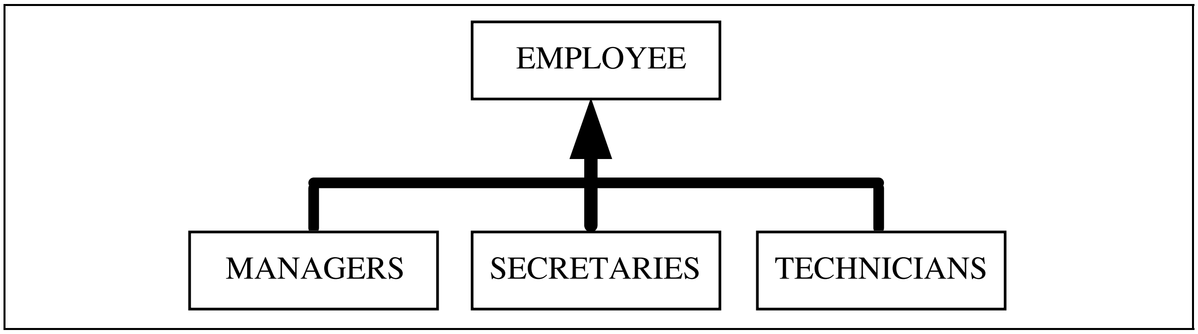
```
EMPLOYEE (EID, D1,…,Dn)
MANAGERS (EID, M1,…,Mp)
TECHNICIANS (EID, T1,…,Tq)
SECRETARIES (EID, S1,…,Sr)
```

The relations' schemas

```
EXEC SQL
 INSERT INTO EMPLOYEE (EID, D1,…,Dn)
 VALUES (:id, :d1,…,:dn);
switch (role)
{
 case '01':
   EXEC SQL
        INSERT INTO MANAGERS (EID, M1,…,Mp)
        VALUES (:id, :m1,…,:mp);
   break;
 case '02':
   EXEC SQL
        INSERT INTO TECHNICIANS (EID, T1,…,Tq)
        VALUES (:id, :t1,…,:tq);
   break;
 case '03':
   EXEC SQL
        INSERT INTO SECRETARIES (EID, S1,…,Sr)
        VALUES (:id, :s1,…,:sr);
   break;
}
```

A typical insertion pattern for a disaggregate hierarchy

The IS-A hierarchies (cont.)



Conceptual schema

EMPLOYEE (<u>EID</u>, D1,…, Dn, M1,…, Mp, T1,…, Tp,S1,…, Sr)

The relation's schema

```
switch (role)
{
case '01':
   EXEC SQL
   INSERT INTO EMPLOYEE (EID, D1,…,Dn, M1,…,Mp)
        VALUES (:id, :a1,…,:an, :m1,…,:mp);
   break;
case '02':
   EXEC SQL
   INSERT INTO EMPLOYEE (EID, D1,…,Dn, T1,…,Tq)
   VALUES (:id, :a1,…,:an, :t1,…,:tq);
   break;
case '03':
   EXEC SQL
   INSERT INTO EMPLOYEE (EID, D1,…,Dn, S1,…,Sr)
   VALUES (:id, :a1,…,:an, :s1,…,:sr);
   break;
}
```

A typical insertion pattern for an aggregate hierarchy

Associations' detection

Type	Pattern	Feature
Schema	NULL <foreign_key> NOT ALLOWED IN <table>	*total association*
Schema	NULL <foreign_key> ALLOWED IN <table>	*partial association*
SQL	SELECT … FROM …, T,… WHERE …T.FK IS [NOT] NULL	*partial association*
SQL	Joins FK-PK have clauses: FROM T WHERE FK1=:host_var1 AND…AND FKn=:host_varn or FROM T, T' WHERE T.FK1 = T'.PK1 AND…AND T.FKn=T'.PKn	*multiple association*

Typical patterns for the detection of the associations

Conclusion

- As the DB Conceptual Schema is semantically much richer than the Physical DB Schema, when reconstructing an ER schema we must look at the constraints that are maintained at the procedural level, too.

- More recent DBMSs offer enhanced possibilities for defining and maintaining the constraints.

- We described a RDBRE methodology that makes use of information taken from:
  - catalog
  - source code

- Innovative aspects:
  - interpreting how applications make use of the data
  - using procedural patterns

- Pros:
  - the "cognitive" approach
  - easy recognition of new *patterns*

- Limitations:
  - the methodology must be refined
  - no user friendly interface at present

- A prototype has been implemented

- Future developments:

- **integration in TROOP: a *reverse engineering* tool under development**