Un approccio cognitivo per il reverse engineering di basi di dati relazionali

Oreste Signore
Mario Loffredo - Mauro Gregori - Marco Cima

AICA 94 - Congresso Annuale
Palermo, 21-23 Settembre 1994

CONSIGLIO
NAZIONALE
delle RICERCHE

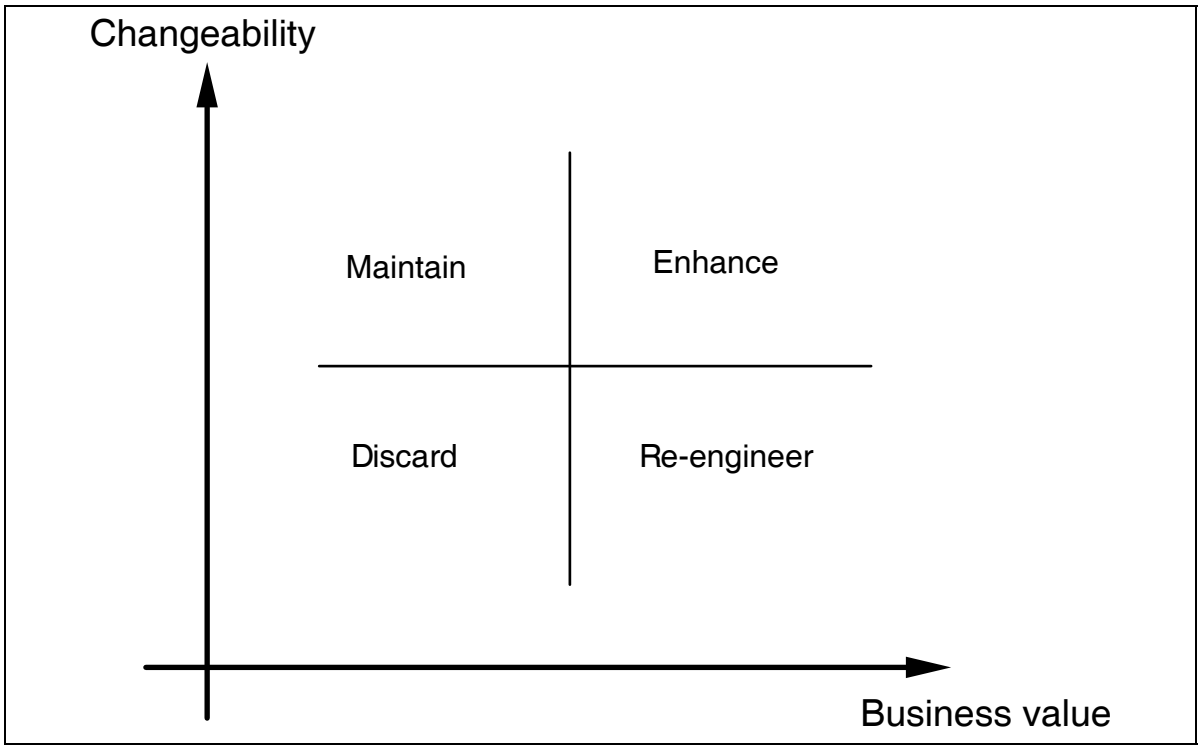
Istituto *CNUCE*
via S. Maria, 36
56126 Pisa (Italy)

tel. +39 (50) 593201
FAX: +39 (50) 904052
e.mail: oreste@vm.cnuce.cnr.it

Contenuto

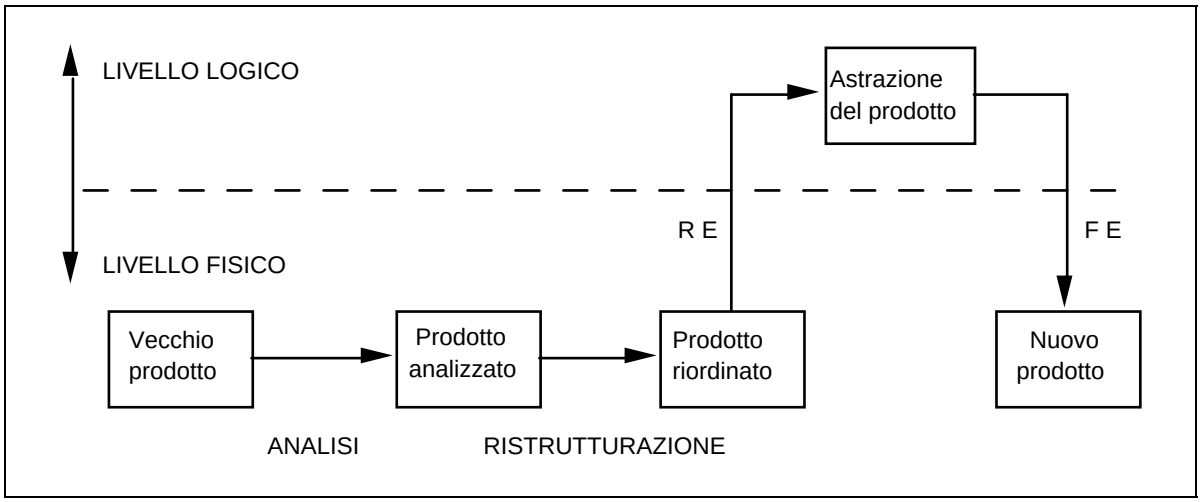
- **Il Data Base Reverse Engineering**
  - **motivazioni**
  - **altre proposte**
- **La metodologia proposta**
  - **architettura generale**
  - **le tre fasi**
- **Conclusioni**

Manutenzione e Re-Engineering



2 **Manutenzione (*correttiva, adattativa* o *perfettiva*):**

- **fino al 95% dell' attività del dipartimento EDP**
- **è necessario comprendere la semantica ed estrarre gli elementi essenziali della progettazione**



Perché
Database Reverse Engineering?

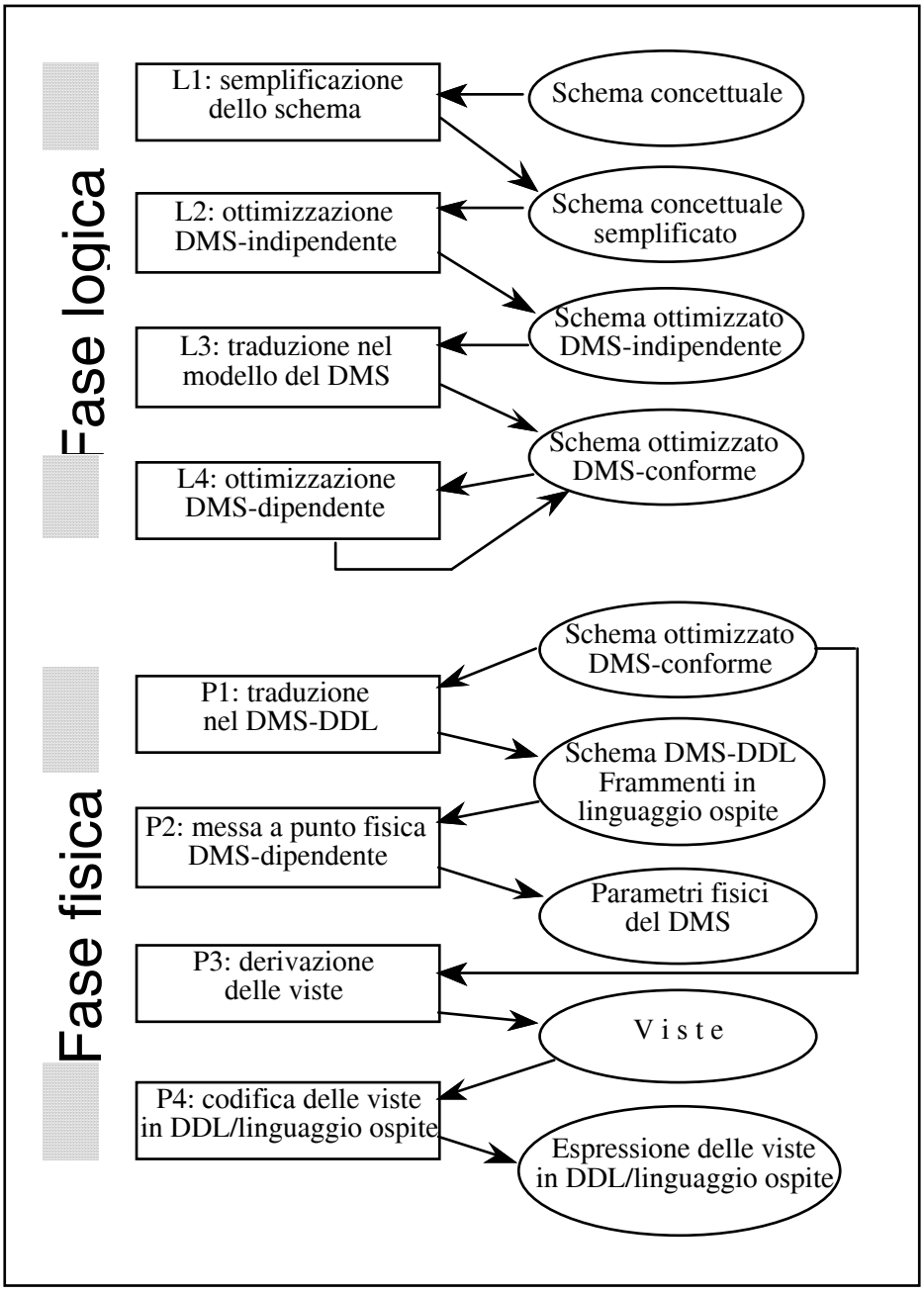
- **Nuovi DBMS più sofisticati**
  - vincoli definibili a livello di schema

- **Reverse Engineering verso Object-Oriented**

- **Importanza delle applicazioni database**

- **Possibilità di estrarre elementi fondamentali della progettazione**
  (recovering design issues)

Database Forward Engineering



Principali processi e prodotti di progetto

- **Si ha un impoverimento semantico dello schema**
  (adattamento schema concettuale al modello logico - traduzione - ottimizzazione schema logico - specifiche non direttamente traducibili)

Database Reverse Engineering: alcune proposte

Ipotesi restrittive:

- **traduzione completa dei requisiti in strutture dati e vincoli**
- **applicazione rigorosa delle regole di mapping**
- **schema non ulteriormente ristrutturato in base a esigenze dell' utente o vincoli di ambiente**
- **esistenza di una "naming policy"**

Batini, Ceri e Navathe

(Batini C., Ceri S., Navathe S.B.: *Conceptual Database Design: An Entity-Relationship Approach*,The Benjamin/Cummings Publishing Company,Inc., 1992)

- **processo semplice e limitato**
- **buon modello iniziale**
- **descrizione chiara e lineare dei passi di analisi delle relazioni e di riconoscimento dei concetti**
- **si prevede una buona conoscenza semantica dello schema relazionale di partenza**

Premerlani e Blaha

(Premerlani W.J., Blaha M.R.: *An Approach for Reverse Engineering of Relational Databases*, Proceedings IEEE Working Conference on Reverse Engineering, Baltimora 1993)

- **impostazione essenzialmente sperimentale**
- **insieme di metodi, tecniche ed esempi pratici**
- **ampia casistica di situazioni reali di partenza**

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))

- **utilizza informazioni desumibili dal catalogo e dai dati memorizzati nelle relazioni**

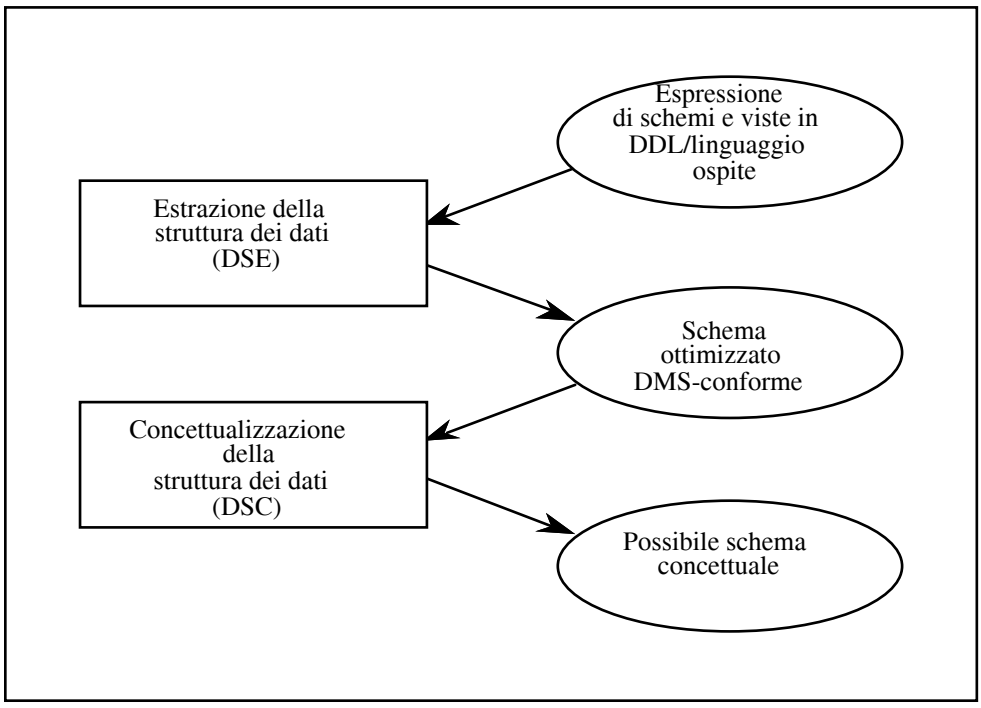
Database Reverse Engineering: alcune proposte (cont.)

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, Baltimora 1993

- **il DBRE consiste nella ricerca di un possibile schema concettuale in grado di condurre ad una organizzazione fisica data dall' insieme delle strutture dei dati nel DDL e nel linguaggio ospite e dai requisiti operazionali**
- **processo risolutivo può essere suddiviso in due fasi generalmente sequenziali:**
  - ***estrazione della struttura dei dati (DSE)***
    (inversa della fase fisica)
  - ***concettualizzazione della struttura dei dati (DSC)***
    (inversa della fase logica)

p Il processo di DBRE:



Database Reverse Engineering (cont.)

p **Estrazione della struttura dei dati (DSE)**
Obiettivi:

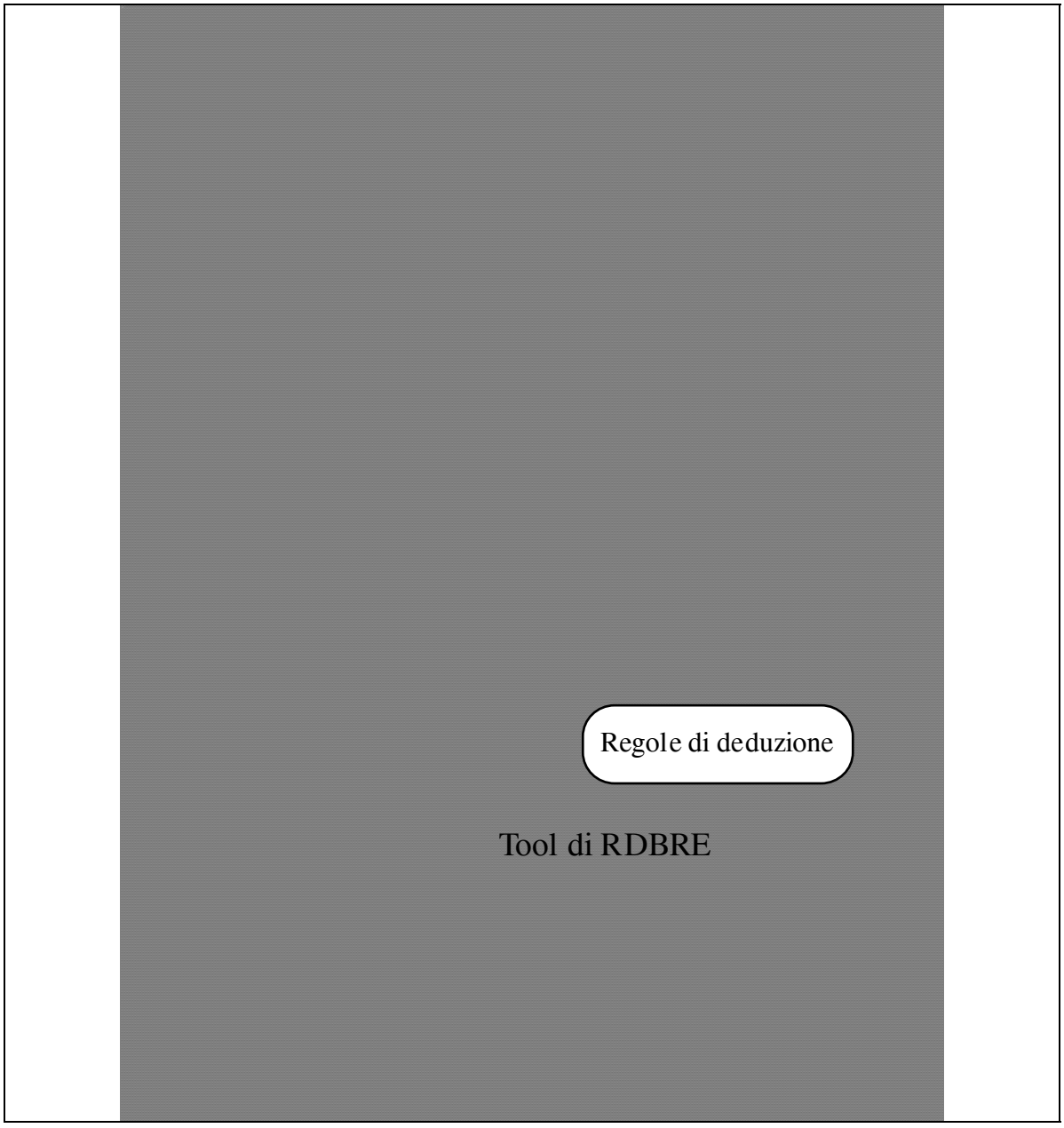
- **ritrovare le strutture dati esplicite ed implicite**
- **ritrovare le specifiche "scartate"**
- **analisi delle parti procedurali delle applicazioni**
  (moduli di programma e trigger: controllo vincoli di integrità, vincoli referenziali e calcolo di dati derivati - struttura standard, semplice e facilmente riconoscibile)
- **analisi dei dati stessi per individuare le proprietà di unicità (identificatori), l'integrità referenziale o il dominio dei valori**

p **Concettualizzazione della struttura dei dati (CSE)**
Obiettivi:

- **interpretare i costrutti dipendenti dal DBMS adottato**
- **eliminare le ridondanze dei dati introdotte per motivi di performance**
- **rendere espliciti gli elementi concettuali nascosti**
- **produrre uno schema concettuale chiaro, naturale e normalizzato**

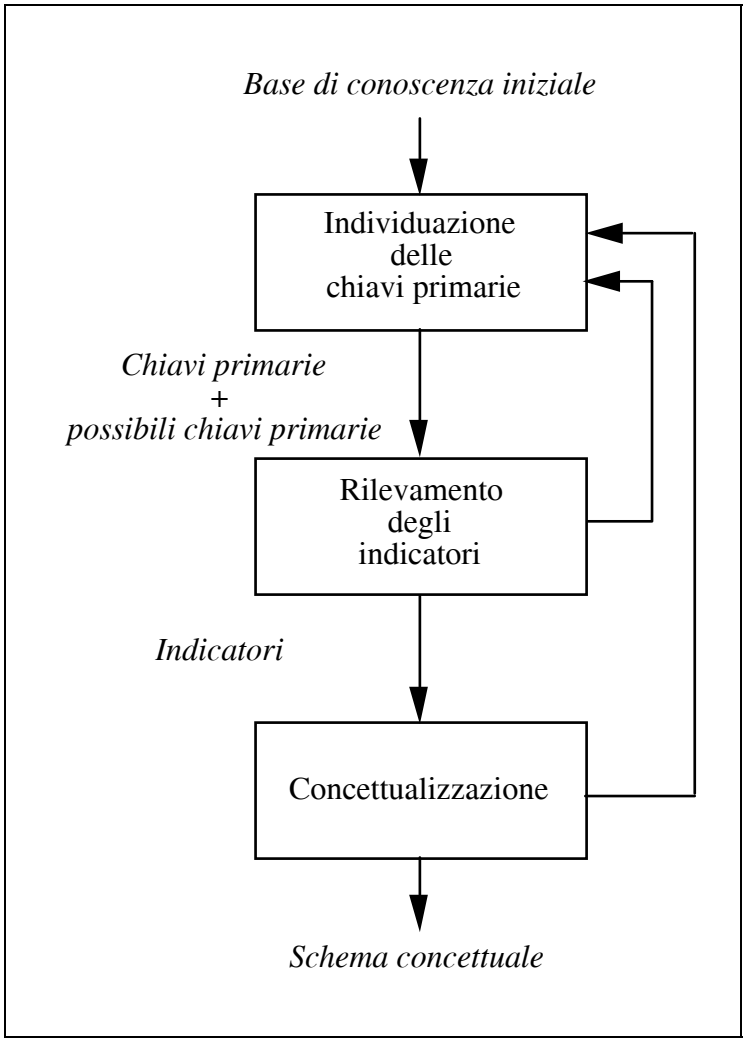
Architettura

p Assunzioni implicite
- **una fase di generazione dei fatti SQL/procedurali**
- **una fase di generazione dei fatti di catalogo**



Ambiente architetturale del tool di RDBRE

Il processo di RDBRE



Le fasi del processo di DBRE

- **Fase 1: Individuazione delle chiavi primarie**
- **Fase 2: Rilevamento degli indicatori**
- **Fase 3: Concettualizzazione**

Gli indicatori

Definizione di indicatore

insieme di informazioni rilevabili da una o più fonti disponibili (catalogo, codice SQL/ospite, risultati di una fase di analisi precedente), che possa costituire un indizio ritenuto significativo per la caratterizzazione, nel modello concettuale, di uno o più elementi dello schema relazionale

Classi di indicatori:

- ***indicatori di schema***
  estratti dal catalogo e dalle informazioni dedotte nella fase di identificazione delle chiavi

- ***indicatori di chiave***
  ricavati dall'analisi delle chiavi primarie
  servono a definire proprietà della chiave primaria di una data relazione

- ***indicatori SQL***
  estratti dall'analisi dei comandi SQL
  forniscono indicazioni sulle modalità con cui gli elementi degli schemi di relazione sono utilizzati nelle interrogazioni e nella manipolazione dei dati

- ***indicatori procedurali***
  estratti dall'analisi del codice del linguaggio ospite
  integrano l'informazione fornita dagli indicatori SQL; sono costituiti da procedure standard (patterns) per manipolazioni condizionate dei dati della base

  Esempi:
  cicli di prelievo delle tuple selezionate, controllo dei vincoli di integrità referenziale, operazioni su tabelle che implementano gerarchie, etc.

Individuazione delle chiavi primarie: concetti generali

- **Banale se ne è prevista la dichiarazione esplicita**

- **Se sulla tabella è definito un solo indice con l' opzione** `UNIQUE`

  la chiave primaria può essere identificata con l' attributo (o l' insieme di attributi) sui quali è definito l' indice

- **Se sulla tabella sono definiti più indici con l' opzione** `UNIQUE`
  - **consideriamo ogni insieme come candidate key**
  - **calcoliamo le frequenze d' uso**
  - **chiediamo all' utente di operare una scelta**

- **Se non si riesce ad identificare una chiave primaria o candidata:**
  possiamo identificare alcuni *indicatori* analizzando pattern procedurali:

  - **deve esistere almeno una clausola** `WHERE` **che menziona tutte le colonne che compongono la potenziale chiave (a)**
  - **la struttura del codice deve escludere che un' operazione di selezione che utilizza tali colonne restituisca un insieme di tuple (b-g)**

Individuazione delle chiavi primarie: i pattern SQL

	Pattern
a	WHERE a_1=<scalar_exp$_1$> AND…AND a_S=<scalar_exp$_S$>
b	No declaration of a cursor like: DECLARE <cursor_id> FOR SELECT <selection> FROM T WHERE a_1=<scalar_exp$_1$> AND…AND a_S=<scalar_exp$_S$> 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.
c	No statement contains: SELECT ALL|DISTINCT <selection> FROM T WHERE a_1=<scalar_exp$_1$> AND…AND a_S=<scalar_exp$_S$>
d	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)
e	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>
f	No statement contains: SELECT <selection> FROM T GROUP BY a_1, a_2,…, a_S
g	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$_S$>

Seconda fase: rilevamento degli indicatori

Prima fase:

- **sono state identificate le chiavi primarie**
- **sono state formulate ipotesi per la PK**
- **è stata indicata l' esistenza di candidate key**

Seconda fase:

- **vengono affrontate le difficoltà derivanti dalla differente ricchezza semantica tra modello ER e relazionale**

- **bisogna tener conto di:**
  - *mapping non univoco*
  - *scelte di ottimizzazione*
  - *povertà del DDL*
  - *tecniche di implementazione non usuali*

- **occorre ragionare sulla base di "indizi"**

- **è articolata in:**
  - *identificazione dei domini*
  - *individuazione delle chiavi esterne (FK)*
  - *individuazione dei vincoli di integrità*
  - *analisi dei vincoli di integrità*

Terza fase (concettualizzazione)

identifica i concetti probabili in base alla opportuna combinazione di indicatori

Identificazione dei domini

p **Soluzione dei casi di omonimia mediante l' uso del nome esteso:**

`tablename.attributename`

p **Identificazione degli attributi definiti sullo stesso dominio:**

- **non è possibile basarsi sull' identità del tipo come definito nel catalogo**
  (SQL ha un controllo debole dei tipi)

  Es.
  Date le relazioni:

```
BOOKS (ID, TITLE, AUT, PUBLISHER,…)
AUTHORS (ID, NAME,…)
```

  e la query:

```
SELECT        NAME
FROM  AUTHORS, BOOKS
WHERE AUTHORS.ID = BOOKS.AUT AND BOOKS.PUBLISHER =
'XYZ'
```

  o:

```
SELECT        NAME
FROM  AUTHORS
WHERE ID      IN
      (SELECT AUT
              FROM   BOOKS
              WHERE  PUBLISHER = 'XYZ')
```

  evidentemente `AUTHORS.ID` **e** `BOOKS.AUT` **sono definiti sullo stesso dominio, mentre** `AUTHORS.ID` **e** `BOOKS.ID` **no, anche se potrebbero avere lo stesso tipo SQL**

Identificazione dei domini (cont.)

- **è possibile dedurre l' equivalenza semantica di due domini dalle procedure che manipolano i dati**
  Es.

Type	Pattern
equijoin	SELECT … FROM T_1, T_2 WHERE …T_1.ATTR = T_2.ATTR'…
multiple join	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$^{(3)}$…
nested queries	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 …)
auto-join	SELECT A.STAFF_ID FROM STAFF A, STAFF B WHERE A.SALARY > B.SALARY AND A.SUPERVISOR = B.STAFF.ID

- **altri esempi di rilevamento di equivalenza semantica:**

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

(equivalenza semantica degli attributi corrispondenti in `<column-commalist>` *e* `<selection-commalist>`**)**

- **il caso in cui vengono utilizzate le variabili ospite è più complesso** *(richiede l' analisi delle dipendenze dei dati)*

Es.

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

è equivalente a:

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

Chiavi esterne

p Tre passi

a) Annotazione delle chiavi esterne definite esplicitamente
Caso elementare se il DDL possiede i costrutti opportuni

b) Identificazione delle chiavi primarie non definite esplicitamente
Data una relazione T con chiave primaria PK, estraiamo i sinonimi della chiave primaria PK che appartengono tutti alla relazione T'.
Essi sono i componenti di una chiave esterna, definita in T', che riferisce T.

c) Identificazione delle chiavi esterne che fanno riferimento a chiavi primarie non certe.
Per tutte le relazioni che hanno solo una indicazione di possibile chiave primaria (PPK) si applica il medesimo processo, trasferendo al risultato l' indeterminazione sulla chiave primaria.

Vincoli di integrità referenziale

- **Nei casi ambigui è necessario confermare le indicazioni ottenute nelle fasi precedenti mediante la verifica dei vincoli di integrità referenziale contenuti nel codice.**

- **Verifica dei vincoli di integrità referenziale:**
  - **nei DBMS meno recenti era affidata ai programmatori**
  - **nei DBMS più recenti è possibile definirli a livello di schema (triggers)**

- **L' identificazione dei pattern procedurali che implementano i vincoli ne permettono una gestione più omogenea**

- **Un esempio**

  (pattern procedurale che assicura l' integrità referenziale all' inserimento nella tabella referenziante)

```
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>
```

Analisi
dei vincoli di integrità referenziale

- **Riconoscere le chiavi esterne e i vincoli di integrità referenziale può consentire l' identificazione delle associazioni.**

- **Gli** *indicatori procedurali* **possono consentire di:**
  - **identificare le chiavi esterne**
  - **confermare o respingere le ipotesi formulate**

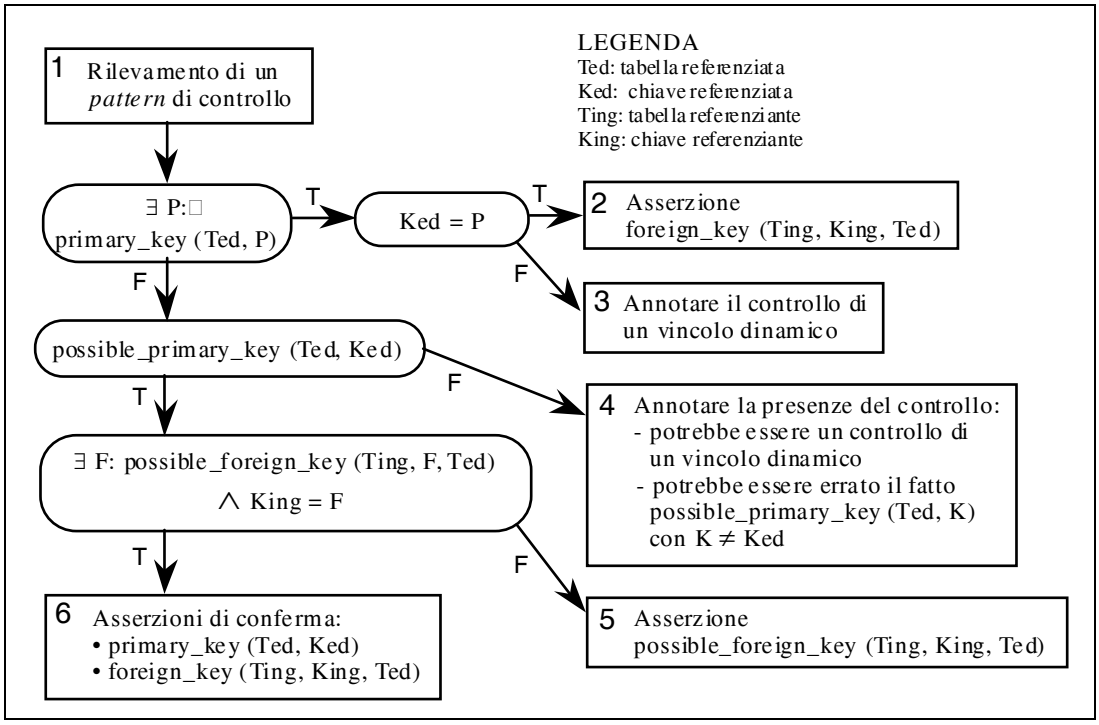
- **Alcuni pattern sono molto simili:**

```
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>
```

esempio di pattern che implementa un vincolo, ma non l' integrità referenziale (il check di esistenza avviene su un campo non chiave)

L’ algoritmo per la verifica dei vincoli di integrità referenziale



Terza fase: la concettualizzazione

p **Un paradigma semplice ed estensibile:**
la matrice degli indicatori

- **le righe corrispondono ai concetti ER**
  (con o senza mapping diretto)
- **le colonne corrispondono alle categorie di indicatori**
- **ogni cella contiene l' indicatore della relativa classe, da cui partire per dedurre il concetto della riga corrispondente**

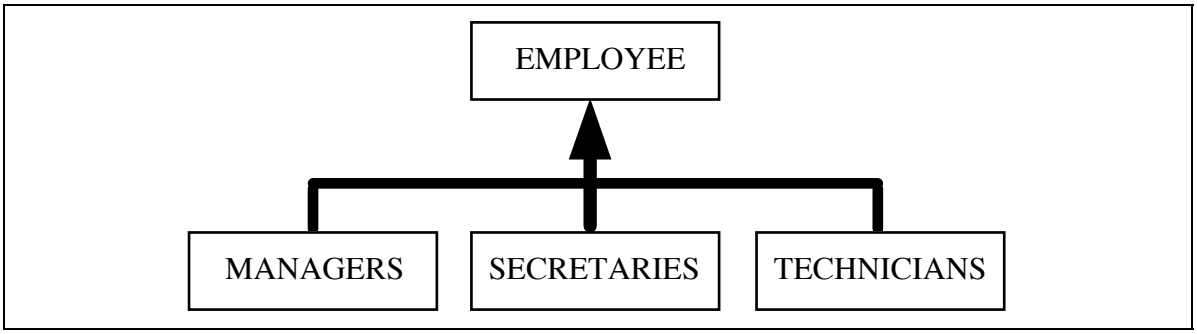
p **Le celle vengono popolate nelle fasi precedenti in base a:**
- **conoscenza dei modelli e delle regole di mapping**
- **conoscenza pratica dedotta dall' esperienza implementativa**

p **L' identificazione del concetto dipende dalla qualità e quantità degli indicatori presenti**

La matrice degli indicatori

La matrice degli indicatori

Le gerarchie IS-A



Schema concettuale

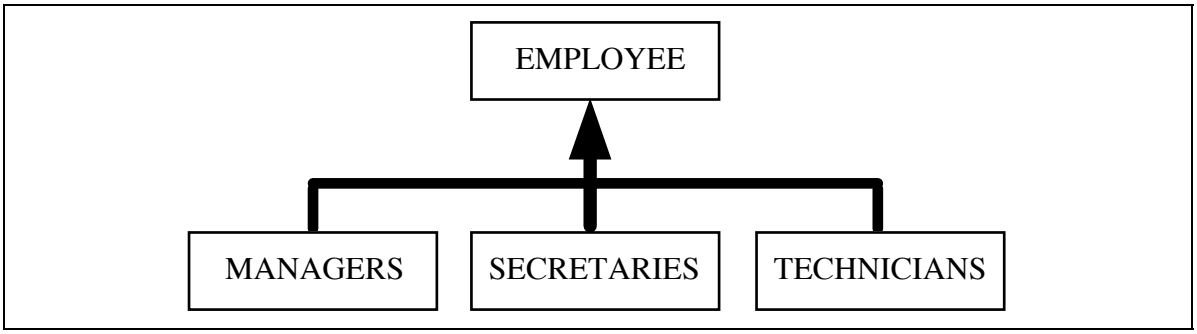
```
EMPLOYEE (EID, D1,…,Dn)
MANAGERS (EID, M1,…,Mp)
TECHNICIANS (EID, T1,…,Tq)
SECRETARIES (EID, S1,…,Sr)
```

Schemi delle relazioni corrispondenti

```
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;
}
```

Tipico pattern di inserzione per una gerarchia disaggregata

Le gerarchie IS-A (cont.)



Schema concettuale

```
EMPLOYEE (EID, D1,…, Dn, M1,…, Mp, T1,…, Tp,S1,…, Sr)
```

Schema della relazione corrispondente

```
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;
}
```

Tipico pattern di inserzione per una gerarchia aggregata

Rilevamento delle associazioni

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*

Pattern tipici per il rilevamento di associazioni

Conclusioni

- **Nella ricostruzione dello schema ER si può verificare che alcuni vincoli vengono mantenuti anche a livello procedurale**

- **È quindi possibile sfruttare le opportunità offerte dai DBMS più recenti**

- **È stata descritta una metodologia di RDBRE che utilizza informazioni desunte da:**
  - **catalogo**
  - **codice sorgente**

- **Aspetti innovativi:**
  - **interpretazione dell' uso dei dati da parte delle applicazioni**
  - **utilizzo di pattern procedurali**

- **Pregi**
  - **Particolare approccio "cognitivo"**
  - **Duttilità dell'approccio nel riconoscimento di nuovi *patterns***

- **Limitazioni**
  - **Metodologia ancora in una fase di definizione iniziale, necessita di ulteriori raffinamenti**
  - **Il prototipo è privo di un'interfaccia utente evoluta**

- **Sviluppi futuri**
  - **Ingegnerizzazione dello strumento realizzato e sua integrazione in un tool di *reverse engineering* in corso di realizzazione (TROOP)**