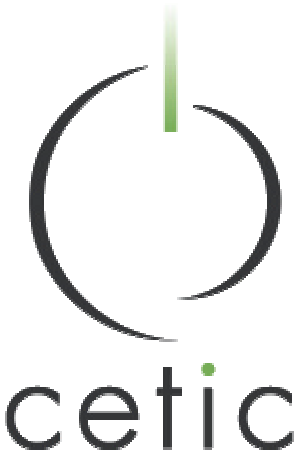
LIBD



Database Reengineering

June 25, 2003

CETIC - Centre d'excellence en TIC

Facultés universitaires de Namur
Institut d'informatique
LIBD - Laboratoire d'ingénierie des applications de bases de données
www.info.fundp.ac.be/libd

LIBD

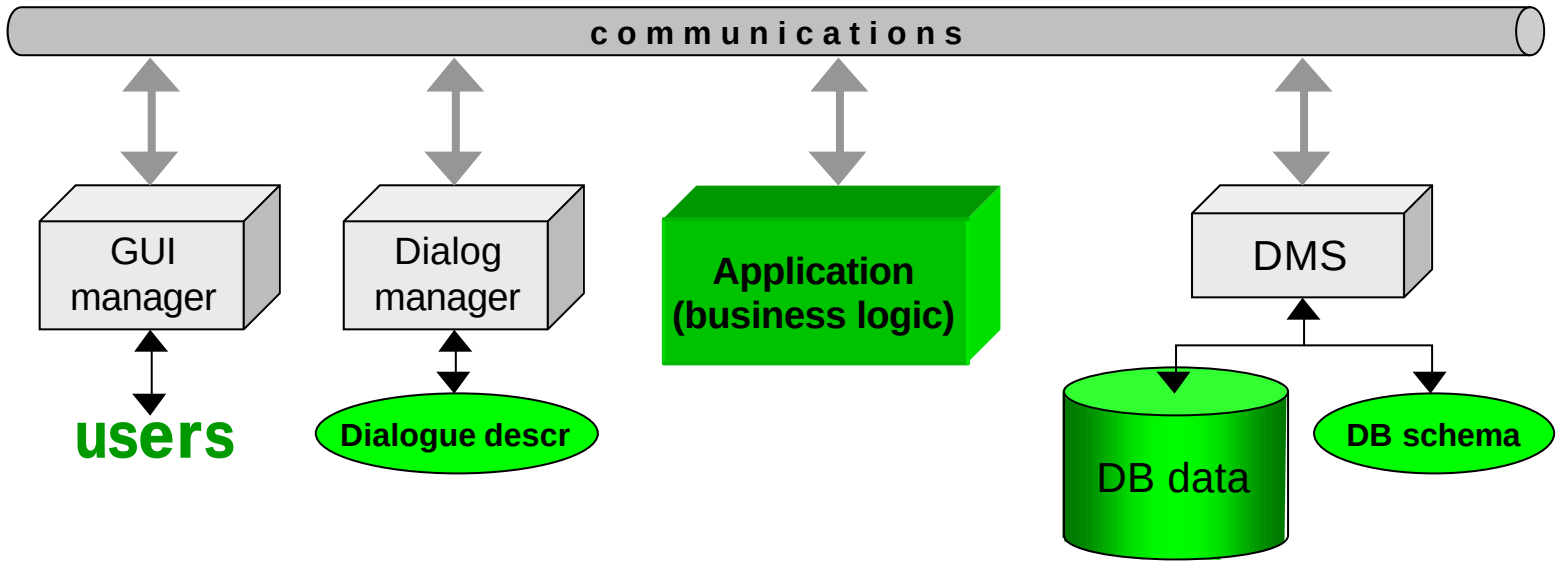
CETIC

Contents

- **Databases: facts and challenges**
- **Database engineering**
- **Database reverse engineering - Introduction**
- **Database reverse engineering - Methodology**
- **Database reverse engineering - Case studies**
- **Database reverse engineering - Toward the web**
- **Database reverse engineering - Conclusions**

Databases: facts and challenges

The role of the *database* in data-centered applications



System components

Application components

LIBD

CETIC

Databases: facts and challenges

Some facts (1)

- a company may use more than 10 DMSs to implement its information system;
- a new version of a DMS appears every 4 years, most often involving changes in programs;
- a database may be used by several thousands programs;
- the schema of a database may include more than 1,000 entity types and 30,000 attributes (technically, 1,250 files/tables and 40,000 fields/columns); SAP = 15,000-30,000 tables and 200,000 columns;
- some database schemas have got so large ands complex that no single developer can master them any longer;
- the precise description of an entity type and its attributes may span from 1 to 100 pages (what does a banking company mean by *product*?)
- the functional documentation of a large database may (*should*) comprise more than 5,000 pages; what about **your** database?
- the SQL-DDL code of a database (tables, constraints, indexes, triggers, checks, etc.) may comprise 200,000 LOC (5,000 pages);

LIBD

CETIC

Databases: facts and challenges

Some facts (2)

- database schemas share some interesting properties with programs:
  - ✓ bugs
  - ✓ awkward design
  - ✓ dead parts
  - ✓ obscure sections
  - ✓ (*nearly*) duplicated sections
  - ✓ developed on obsolete platforms
- corrective, preventive and adaptive maintenance (*no added value*) of an IS may require more than 50% of development effort;
- maintenance and evolution are **much more costly** when **no correct documentation is available**.

LIBD

CETIC

Database engineering

Database engineering is a subdomain of software engineering that addresses such processes as analysis, design, implementation, reengineering, migration, integration of databases and of their applications.

Based on

- ***models*** *(to describe)*
- ***techniques*** *(to manipulate descriptions)*
- ***methods*** *(to organise way of working)*
- ***tools*** *(to support models, techniques and methods)*

LIBD

CETIC

Database engineering

Some engineering processes (1)

- **Information analysis** / **conceptual design**: building a technology-independent specification of the information to be stored in the database = the *conceptual schema* of the DB, expressed into any conceptual model (*models*: ERA, UML classes, NIAM, ORM, etc.). ***Yields a complete, up-to-date functional documentation.***

- **Logical design**: expressing (implementing) the conceptual schema according to the model of a specific data management system or DMS (*models*: standard files, CODASYL, IMS, relational, OO, object-relational, XML-DTD, XML schema, etc.) = the *logical schema* of the DB. ***Yields a complete, up-to-date technical documentation.***

- **Physical design**: defining the technical constructs and parameters that make the DB efficient and robust (indexes, storage spaces, clusters, buffer management, locking strategies, physical storage schemes and parameters, etc.). Produces the *physical schema*. ***Should be carefully documented.***

Database engineering

Some engineering processes (2)

- **Schema coding**: producing the DMS-DDL program to be compiled by the DMS. Includes declarative code, but can also include complex procedural fragments: *triggers* (database daemons) and *stored procedures* (shared database methods). Generally a **lossy process** (some constructs and constraints are left untranslated). ***Should be carefully documented.***

- **Reverse engineering**: recovering the most correct description of the data structures of a poorly documented database, that is, its logical and conceptual schemas. ***Needed for all the following processes***.

- **Maintenance**: fixing bugs (*corrective*) in schemas, improving data structures (*preventive*), adapting to changing infrastructure (*adaptive*). May imply data conversion and program modification. ***Requires a complete, up-to-date documentation of the DB.***

Database engineering

Some engineering processes (3)

- **Evolution**: changing the schemas of the database to follow new requirements. May imply data conversion and program modification. ***Requires a complete, up-to-date documentation of the DB.***

- **Database reengineering:** In general, any process which produces an improved version of a legacy database according to definite criteria (correctness, freeing from obsolete constructs, normalisation, optimisation, distribution, using modern technologies, etc.). Sometimes called *revalidation* or *renovation*. ***Requires a complete, up-to-date documentation of the source DB.***

- **Database/application migration**: Porting a complete legacy application, or some of their components, on another, generally up-to-date, platform. For a database: changing its DMS. (Un)popular example: *migrating the files of a COBOL application to a RDBMS*. ***Requires a complete, up-to-date documentation of the source DB.***
  Beware: the *one-to-one migration* is the cheapest but also the worse approach since it deeply degrades the final structure. As a consequence, it is the most popular!

Database engineering

Some engineering processes (4)

- **Data conversion**: Extracting data from a database to load them into another repository or to transform them according to a definite format (ETL, XML extraction, feeding datawarehouse, etc.). ***Requires a complete, up-to-date documentation of the source DB.***

- **Interfacing**: Building a converter between two incompatible processors. Example: Object ↔ relational, relational ↔ XML. ***Requires a complete, up-to-date documentation of the source DB.***

- **Integration/federation**: (1) Building, loading and managing a database that includes data from several independent, heteregeneous, distributed databases. (2) Building a virtual database that allows transparent access to several independent, heteregeneous, distributed databases. ***Requires a complete, up-to-date documentation of the source DBs.***

LIBD

CETIC

Database reverse engineering - Introduction

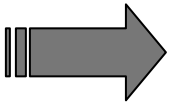
Objective: recovering an up to date documentation of a legacy database or of a set of files.

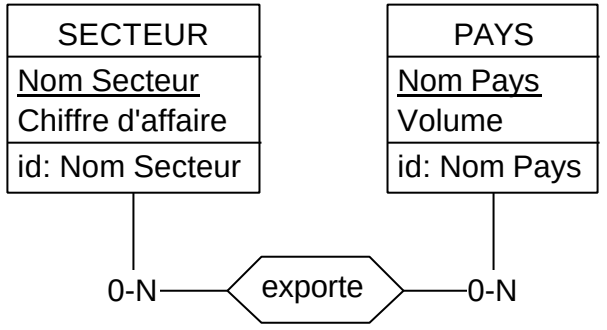
- Should be needless (*up to date documentation should be available*).
- When needed, should be easy (*clear and readable data structures*)
- Actually is always needed and not easy at all!

Database reverse engineering - Introduction

DBRE: the easy view

```
create table PAYS (
   NOMP char(24) not null,
   VOLUME numeric(12) not null,
   primary key (NOMP));

create table EXPORT (
   NOMP char(24) not null,
   NOMS char(18) not null,
   primary key (NOMP, NOMS),
   foreign key (NOMP)
       references PAYS,
   foreign key (NOMS)
       references SECTEUR);

create table SECTEUR (
   NOMS char(18) not null,
   CA numeric(14) not null,
   primary key (NOMS));
```



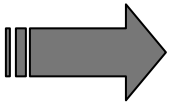


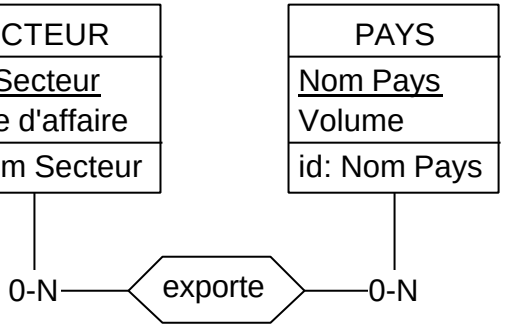
Database reverse engineering - Introduction

DBRE: the reality view (1)

```
create table TBL_P (
   IDP char(24) not null,
   COL_2 numeric(12) not null);

create table TBL_X (
   IDX char(42) not null);

create table TBL_S (
   IDS char(18) not null,
   COL_2 numeric(14) not null);

create unique index ID_P on TBL_P(IDP);
create unique index ID_X on TBL_X(IDX);
```

?

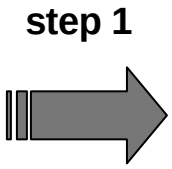


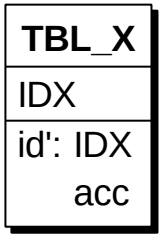


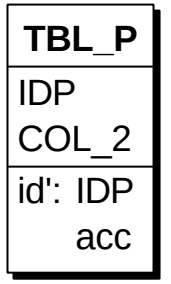
Database reverse engineering - Introduction

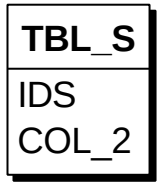
DBRE: the reality view (2)

```
create table TBL_P (
   IDP char(24) not null,
   COL_2 numeric(12) not null);

create table TBL_X (
   IDX char(42) not null);

create table TBL_S (
   IDS char(18) not null,
   COL_2 numeric(14) not null);

create unique index ID_P on TBL_P(IDP);
create unique index ID_X on TBL_X(IDX);
```



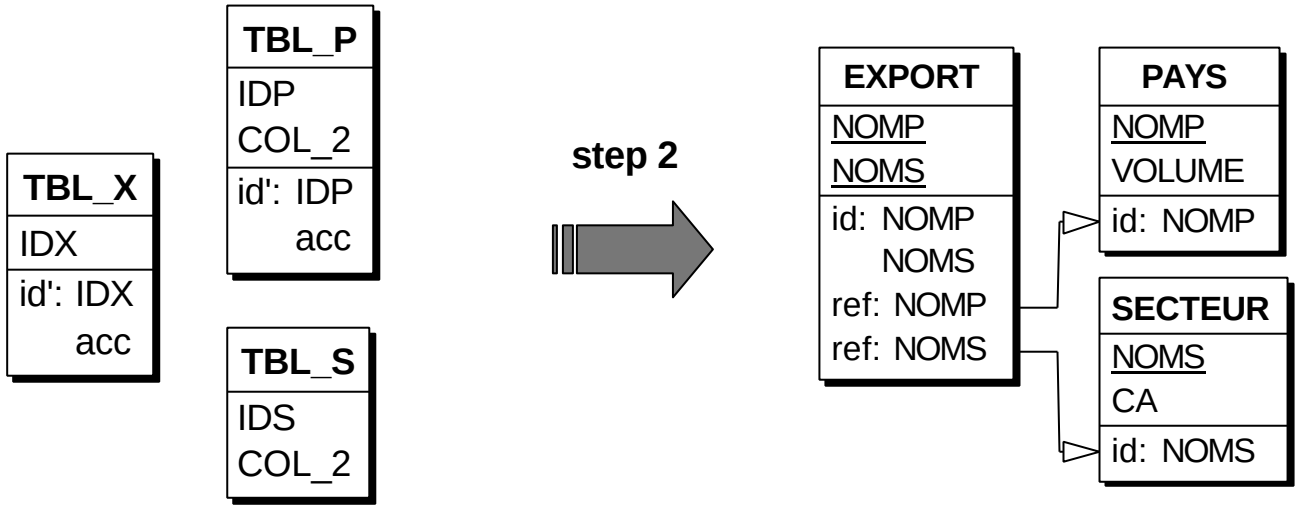






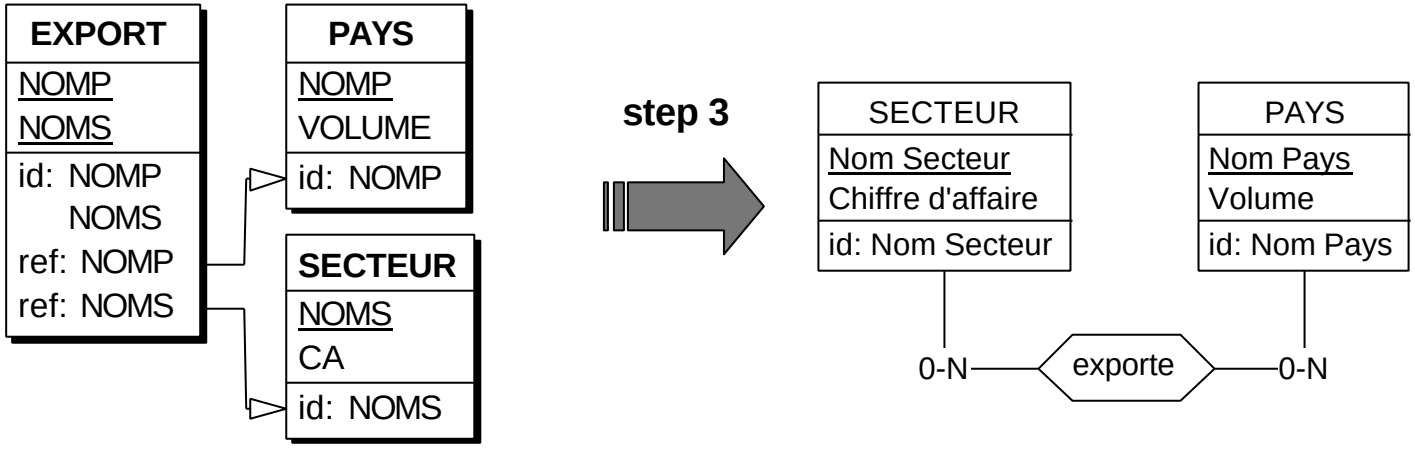
Database reverse engineering - Introduction

DBRE: the reality view (3)



Database reverse engineering - Introduction

DBRE: the reality view (4)



LIBD

CETIC

Database reverse engineering - Methodology

DBRE comprises two main phases, namely *Data structure extraction* and *Data structure conceptualization*.

- ***Data structure extraction***: recovering the actual data structures and constraints. **Result**: a complete, up to data logical schema.

  Complete = explicit constructs + implicit constructs.

  Challenge: how to identify the implicit constructs

- ***Data structure conceptualization***: rebuilding the functional description of the database. **Result**: a complete, up to date, readable conceptual schema.

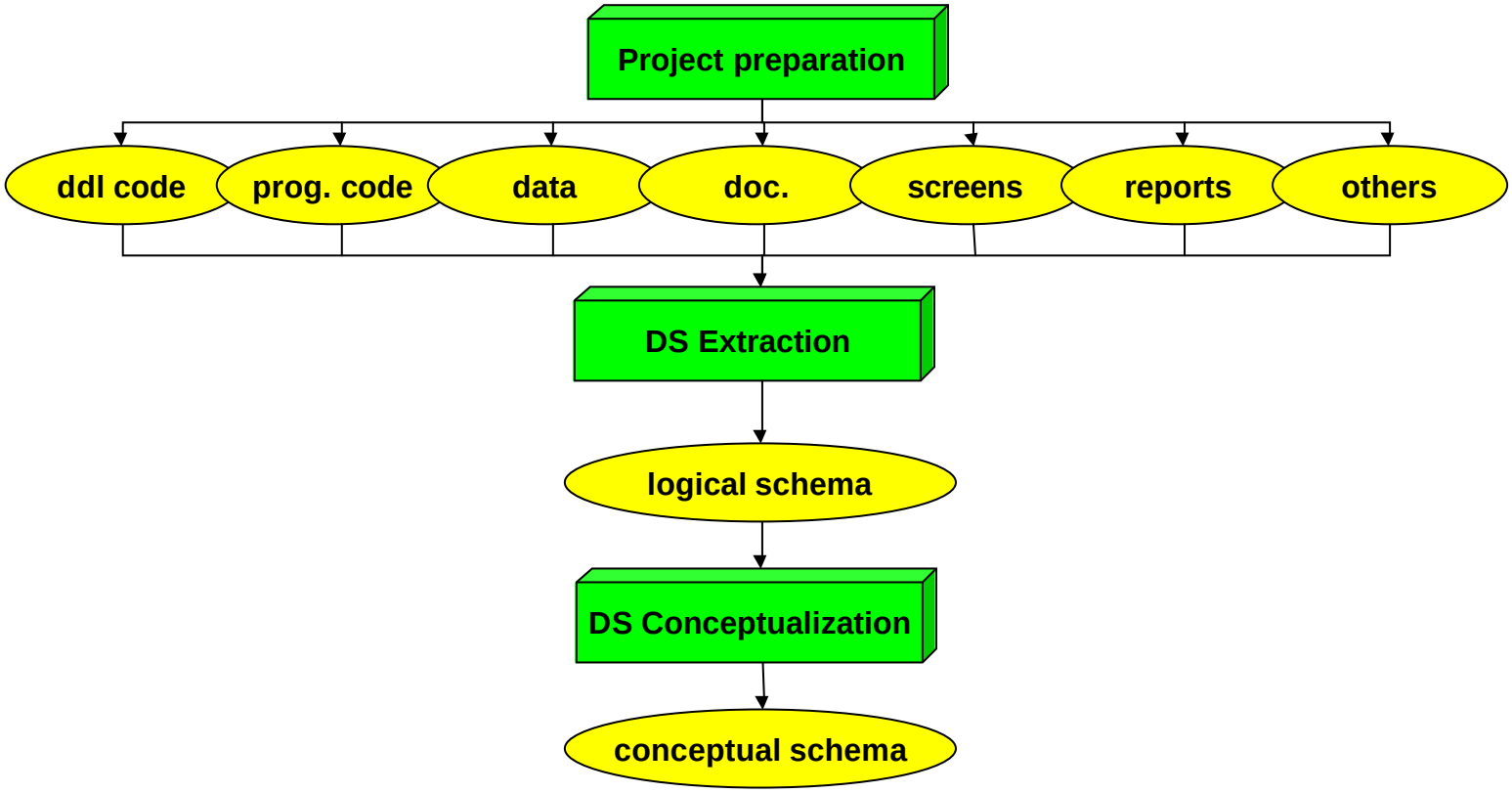
  Challenge: understanding (i.e., knowing the very meaning of) the data structures.

LIBD

CETIC

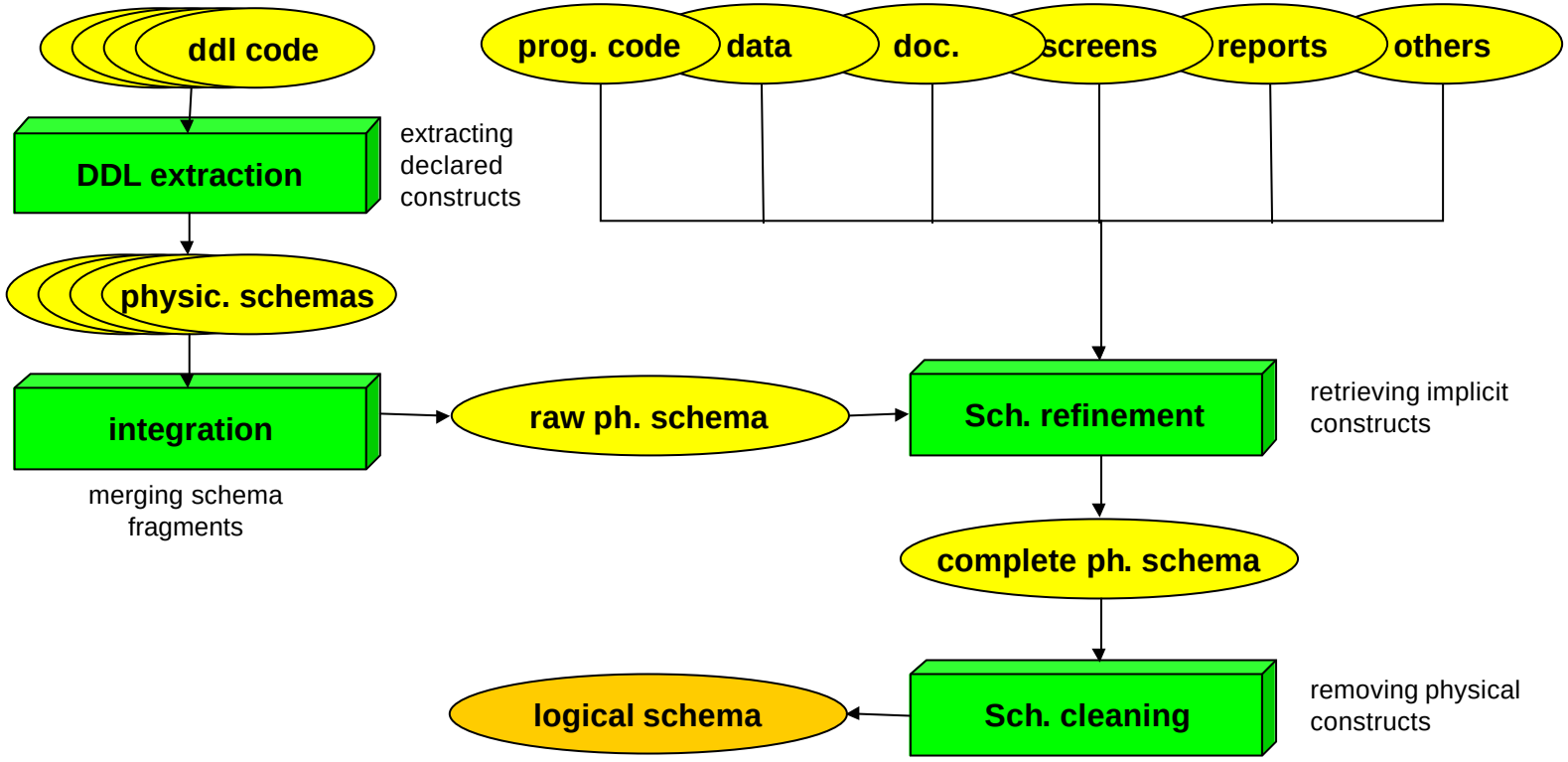
Database reverse engineering - Methodology

General methodology



Database reverse engineering - Methodology

Data structure extraction



LIBD

CETIC

Database reverse engineering - Methodology

Schema refinement: the goals

What must we look for
or *what has been lost when the database was designed?*

- the fine-grained structure of record types and compound fields;
- optional (nullable) fields;
- multivalued fields;
- record type and multivalued field identifiers;
- foreign keys (standard and non-standard);
- redundancies (duplicated or computed fields);
- value domains;
- meaningful names;
- *etc*.

LIBD

CETIC

Database reverse engineering - Methodology

Schema refinement: sources and techniques

Where and how can we discover implicit constructs?

- **documentation**, if any; including comments in DDL sources and dictionaries
- **program code**; shows how the data are read, processed, checked and written (the *usage patterns*);
  huge and complex; better to analyze synthetic excerpts: *representative patterns* (looking for join-based queries), *data flow* and *dependency diagrams* (showing relationships between apparently independent fields), *program slicing* (that includes validation code);
- **data**; are used either to **discover constructs** (find field decomposition), or to **validate hypotheses** on possible constructs (proving/disproving that field NCLI is a foreign key);
- **screen**; a screen is a view on the data; it can tell a lot on field names, field association/grouping, field meaning and constraints;
- **reports**; similar to screen; in addition, this view is populated by sample data;
- etc.

LIBD

CETIC

Database reverse engineering - Case studies

The DaimlerChrysler database (see demo)

- The raw physical schema
- The logical schema
- The conceptual schema

D'Ieteren database

Retrieving about 200 implicit foreign keys in an IBM IMS database: 6 weeks (by two last year students).

Baxter

Rebuilding the conceptual schema of an RPG3 database: 2 months by a last year student.

LIBD

CETIC

Database reverse engineering - Case studies

Ville de Namur

Federating two tax recovery databases:

3 Suisses (St Brice)

Rebuilding the conceptual schema of a Datacom/DB database (about 850 tables) and comparing it to a reference conceptual schema: 30 days.

Bus company

Retrieving the logical schema of a Business Basic set of files: *failure*.

Foreign municipality

Conciliating independent Access databases: first phase 3 weeks; second phase in preparation.

LIBD

CETIC

Database reverse engineering - Case studies

Extraction company

Rebuilding the logical schema of a database comprising 460 COBOL record types. Project pending, local developer too busy.

Pre-analysis for rural accounting system

Main sources: the encoding forms. Validation by interviews. 3 months.

Pre-analysis for an integrated student management system

Buidling the conceptual schema of the future database. Main sources: student registration forms, various reports, the current COBOL application, interviews of academic and administrative secretaries. 4 months.

LIBD

CETIC

Database reverse engineering - Toward the Web

The project WebReverse - Observations

- A web site is a collection of semi-structured and interrelated pages that comprise essential information on a company or an administration.
- Many websites are static, in that, their contents do not come from other electronic sources, such as databases, WP documents or XML documents.
- In many cases, the web site **is** an important part of the corporate information system.
- Web site reengineering consists in transforming it into two distinct parts, namely explicitely structured data (database ou XML documents) and presentation information.

LIBD

CETIC

Database reverse engineering - Toward the Web

The project WebReverse - Objectives

The project aims at building techniques and tools:

- to identify the semantically pertinent components in representative HTML documents
- to assign semantics to these components
- to derive conceptual schemas, XML schemas (or DTDs) from this knowledge
- to use this knowledge to scan large collections of similar documents to extract the page contents in a structured way, and to store them in a database or in XML documents
- additionally (if possible) to extract the presentation information to generate XSL directive to rebuild the web site;
- last, but not least, to build tools to (semi-)automate these processes.

LIBD

CETIC

Database reverse engineering - Conclusions

Sentences, aphorismes et philosophie de trottoir (1)

- La rétro-ingénierie compense des lacunes graves dans le comportement des équipes de développement en matière de documentation. Liées à la fois aux contraintes organisationnelles (dépenses/gains à court ou à long terme) et à la psychologie humaine (principe de plaisir), ces lacunes sont universelles, ont toujours existé et existeront probablement pendant longtemps.

- La rétro-ingénierie est incontournable (bien que souvent contournée !) dans tous les processus impliquant la transformation de bases de données, mais aussi dans leur simple utilisation par des informaticiens extérieurs. Elle est aussi à la base de la fonction d'administration de données (maîtrise de l'existant).

LIBD

CETIC

Database reverse engineering - Conclusions

Sentences, aphorismes et philosophie de trottoir (2)

- La rétro-ingénierie exige des compétences difficiles à rencontrer chez un même individu : théorie des BD (normalisation), pratique des SGBD (y compris anciens), connaissance des langages de programmation classiques (y compris anciens), connaissance des machines et des systèmes d'exploitation (y compris anciens), pratique du développement de programmes, psychologie (connaissance des tréfonds de l'âme humaine, patience), sens politique (la direction générale a toujours raison), sens du dévouement (la rétro-ingénierie est à l'intersection des services de voirie, de la médecine interne et de la gériatrie), modestie (le résultat semble toujours évident). En outre cet oiseau rare aurait pour plus vif désir de passer ses journées à pratiquer la rétro-ingénierie !

- Le succès d'un projet de rétro-ingénierie (ou de migration) dépend de l'adhésion de la direction. Importance d'une stratégie claire, d'un plan directeur plausible et argumenté et d'une planification stricte. En particulier, risques techniques et organisationnels souvent sous-estimés. Première source d'échecs.

LIBD

CETIC

Database reverse engineering - Conclusions

Sentences, aphorismes et philosophie de trottoir (3)

- Le succès d'un projet de rétro-ingénierie dépend de l'adhésion des informaticiens. Fortes réticentes fréquentes car le processus correspond à un audit des pratiques personnelles. Crainte d'une dépossession de la maîtrise du développement et de la maintenance : perte de pouvoir. Deuxième source d'échecs.

- La rétro-ingénierie n'est pas qu'un processus technique. Il induit de fortes tensions organisationnelles et psychologiques. L'externalisation (outsourcing) est délicat s'il n'y a pas d'implication de l'organisation. En particulier, le processus n'étant jamais terminé (comme pour la sécurité, 100% de précision est un objectif économiquement irréaliste) les techniciens de l'organisation doivent s'approprier les résultats pour les améliorer progressivement.

- Le taux de satisfaction d'un projet de rétro-ingénierie est inférieur à celui d'un développement classique. L'oubli d'un bout de programme parmi 3 MLOC peut empêcher de détecter une contrainte dont la violation risque de détériorer progressivement les données.

LIBD

CETIC

Database reverse engineering - Conclusions

Sentences, aphorismes et philosophie de trottoir (4)

- Les techniciens impliqués doivent recevoir une formation liée au processus afin d'évaluer correctement les enjeux et les risques.
- L'utilisation d'outils d'analyse est une condition essentielle à tout projet de rétro-ingénierie.