Chapter XXI
Database Reverse Engineering

Jean-Luc Hainaut
University of Namur, Belgium

Jean Henrard
REVER s.a., Belgium

Didier Roland
REVER s.a., Belgium

Jean-Marc Hick
REVER s.a., Belgium

Vincent Englebert
University of Namur, Belgium

INTRODUCTION

Database reverse engineering consists of recovering the abstract descriptions of files and databases of legacy information systems. A legacy information system can be defined as a "data-intensive application, such as [a] business system based on hundreds or thousands of data files (or tables), that significantly resists modifications and changes" (Brodie & Stonebraker, 1995). The objective of database reverse engineering is to recover the logical and conceptual descriptions, or schemas, of the permanent data of a legacy information system, that is, its database, be it implemented as a set of files or through an actual database management system.

The logical schema is the technology-dependent (e.g., relational) description of the database structures while the conceptual schema is the abstract, technology-independent description of their semantics.

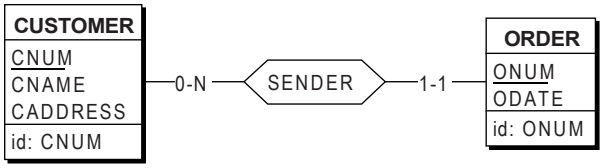
Database reverse engineering often is the first steps of broader engineering projects. Indeed, rebuilding the precise documentation of a legacy database is an absolute prerequisite before migrating, reengineering, maintaining or extending it, or merging it with other databases.

The current commercial offering in CASE tools poorly supports database reverse engineering. Generally, it reduces to the straightforward derivation of a conceptual schema such as that of Figure 1 from the following DDL code.

Figure 1. A naive view of data reverse engineering

```
create table CUSTOMER (
  CNUM decimal(10) not null,
  CNAME varchar(60) not null,
  CADDRESS varchar(100) not null,
primary key (CNUM))
```

```
create table ORDER (
  ONUM decimal(12) not null,
  SENDER decimal(10) not null,
  ODATE date not null,
primary key (ONUM),
foreign key (CNUM) references CUSTOMER))
```



Unfortunately, actual database reverse engineering often is closer to deriving the conceptual schema of Figure 2 from the following sections of COBOL code, using meaningless names that do not declare compound fields or foreign keys.

Getting such a result obviously requires additional sources of information, which may prove more difficult to analyze than mere DDL statements. Untranslated (implicit) data structures and constraints, empirical implementation approaches and techniques, optimization constructs, ill-designed schemas, and, above all, the lack of up-to-date documentation are some of the difficulties that the analysts will face when trying to understand existing databases.

The goal of this article is to describe the problems that arise when one tries to rebuild the documentation of a legacy database and the methods, techniques, and tools through which these problems can be solved. A more in-depth analysis can be found in Hainaut (2002).

BACKGROUND: STATE OF THE ART AND KEY PROBLEMS

Database reverse engineering has been recognized to be a specific problem in the ’80s, notably in Casanova and Amaral De Sa (1984), Davis and Arora (1985), and Navathe (1988). These pioneer-

Figure 2. A more realistic view of data reverse engineering

```
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-PF0S-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).
```