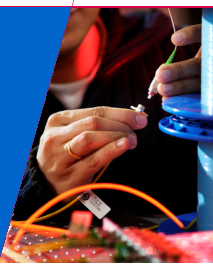


2IS55 Software Evolution

# Implementing evolution: Database migration

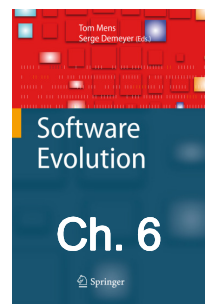
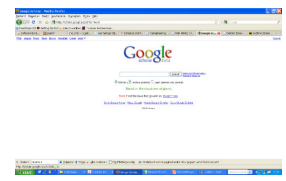
Alexander Serebrenik



**TU/e** Technische Universiteit Eindhoven University of Technology

Where innovation starts

## Sources

/ SET / W&I 31-5-2010 PAGE 1

**TU/e** Technische Universiteit Eindhoven University of Technology

## Last week

- Assignment 8
  - How is it going?

/ SET / W&I 31-5-2010 PAGE 2

**TU/e** Technische Universiteit Eindhoven University of Technology

## Databases

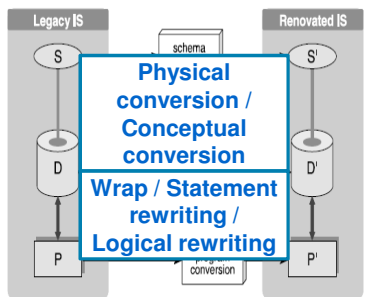
- Central for information systems
- Contain major company assets: data
- Often developed using outdated technology
- Migration should
  - Preserve the data
  - Improve the technology used
    - Flexibility
    - Availability of skills

/ SET / W&I 31-5-2010 PAGE 3

**TU/e** Technische Universiteit Eindhoven University of Technology

## Database migration

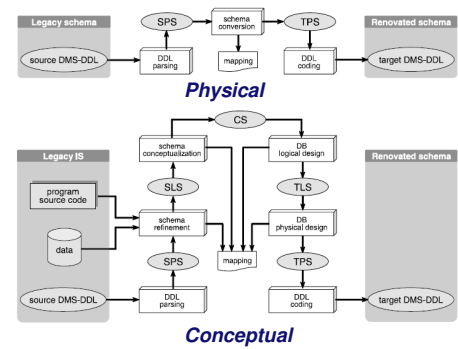
- S – DB schema
- D – DB data
- P – data manipulation programs



/ SET / W&I 31-5-2010 PAGE 4

**TU/e** Technische Universiteit Eindhoven University of Technology

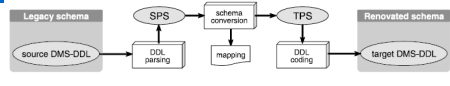
## Schema conversion: Physical vs Conceptual



/ SET / W&I 31-5-2010 PAGE 5

**TU/e** Technische Universiteit Eindhoven University of Technology

## Schema conversion: Physical



**SQL**

**COBOL**

```

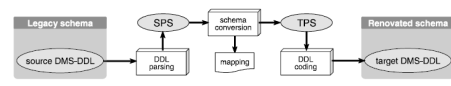
DATA DIVISION.
FILE SECTION.
FD PERSON-FILE
  DATA RECORD IS PERSON-ITEM.
01 PERSON-ITEM.
  02 PERSON-KEY.
    03 PERSON-ID PICTURE X(4).
    02 PERSON-NAME PICTURE X(20).
    02 PERSON-ADDRESS PICTURE X(20).
    02 PERSON-CITY PICTURE X(18).
  
```

**Advantages and disadvantages of physical conversion?**

**TU/e** Technische Universiteit Eindhoven University of Technology

/ SET / W&I 31-5-2010 PAGE 6

## Schema conversion: Physical

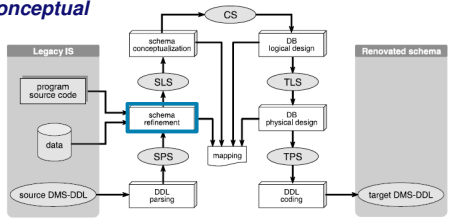


- Easy to automate
  - Existing work: COBOL ⇒ relational, hierarchical ⇒ relational, relational ⇒ OO
- “Migration as translation” vs “migration as improvement”
- Semantics is ignored
  - Limitations of COBOL ⇒ Design decisions in the legacy system ⇒ Automatic conversion ⇒ the same design decisions in the new system
    - Risk: compromised flexibility

**TU/e** Technische Universiteit Eindhoven University of Technology

/ SET / W&I 31-5-2010 PAGE 7

## Schema conversion: Physical vs Conceptual



**Conceptual**

- Refinement: Data and code may contain implicit constraints on the schema
- Conceptualization: Remove implementation details

**TU/e** Technische Universiteit Eindhoven University of Technology

/ SET / W&I 31-5-2010 PAGE 8

## Implicit constraints [Cleve, Hainaut 2008]

- DB schema as defined by DDL commands
- Query

```

select substring(Address from 61 for 30) into :CITY
from CUSTOMER C, ORDERS O
where C.Num = O.Sender and O.Num = :ORDID
  
```

CUSTOMER	
Num:	num (8)
Name:	varchar (30)
Address:	char (120)
Id:	Num

ORDERS	
Num:	num (10)
Date:	date (1)
Reference:	char (12)
Sender:	num (8)
Id:	Num

- What are the implicit constraints implied?

**Field refinement**

CUSTOMER	
Num:	num (8)
Name:	varchar (30)
Address:	compound (120)
Data1:	char (60)
City:	char (30)
Data2:	char (30)
Id:	Num

**Foreign key elicitation**

ORDERS	
Num:	num (10)
Date:	date (1)
Reference:	char (12)
Sender:	num (8)
Id:	Num
ef:	Sender

**TU/e** Technische Universiteit Eindhoven University of Technology

/ SET / W&I 31-5-2010 PAGE 9

## Field refinement

- Explicit
- select substring(Address from 61 for 30) into :CITY
- Implicit: 4 code fragments

a) Local variable (“working storage”)

```

01 DESCRIPTION.
  02 NAME PIC X(20).
  02 ADDRESS PIC X(40).
  02 FUNCTION PIC X(10).
  02 REC-DATE PIC X(10).
  
```

b) DB table (“file section”)

```

FD CUSTOMER.
01 CUS.
  02 CUS-CODE PIC X(12).
  02 CUS-DESCR PIC X(80).
  02 CUS-HIST PIC X(1000).
  
```

c) MOVE DESCRIPTION TO CUS-DESCR.

d) MOVE CUS-DESCR TO DESCRIPTION.

- CUS-DESCR and DESCRIPTION refer to the same data
- They should have the same structure

**TU/e** Technische Universiteit Eindhoven University of Technology

/ SET / W&I 31-5-2010 PAGE 10

## How can we elicit foreign keys?

- Statically and dynamically
  - Do you remember the difference?
- Statically:
  - Parsing (easy for COBOL, difficult for Java)
  - M.Sc. thesis of Martin van der Vlist: “Quality Assessment of Embedded Language Modules”
- Dynamically:
  - Instrument the code
  - Capture traces
  - “Guess constraints”

**TU/e** Technische Universiteit Eindhoven University of Technology

/ SET / W&I 31-5-2010 PAGE 11

## Static approaches

- Look for programming clichés:
  - Data validation, modification, access
- Look at the attribute comparisons used
- Look at the variables produced by queries or imported to them

```
EXEC SQL
SELECT XAP626.HIA_HEFFINGSGRONDSLAG
      INTO :A4.HIA-HEFFINGSGRONDSLAG
FROM XAP626
WHERE (TO_CHAR(XAP626.HIA_DATUM_INGANG, 'YYYYMMDD')
      = :Q4.HIA-DATUM-INGANG
      AND
      XAP626.HIA_VOLGNUMMER = :Q4.HIA-VOLGNUMMER )
END-EXEC
```

/SET /W&I

31-5-2010 PAGE 12

## COBOL vs Java

[vd Vlist, Roubtsov, Serebrenik, vd Brand 2009]

- COBOL: EXEC SQL END-EXEC quotes
- Java: Strings
- Can be freely manipulated:

```
String sql = "SELECT cd.credit ";
sql += "FROM CustomerDetails cd" +
        "WHERE cd.category = " +
        this.getCategory();
if (this.restrict) {
    sql += " AND cd.restriction = 1";
}
sql += " AND cd.id = ?";
PreparedStatement s = this.con.prepareStatement(sql);
s.setInt(1, id);
ResultSet rs = s.executeQuery();
```

/SET /W&I

31-5-2010 PAGE 13

## Dynamic approaches

- Look at the trace:

```
select count(*) from CUSTOMER where Num = 'C400'
getint(1) = 1
insert into ORDERS(Num, Date, Reference, Sender) values (456,'2008-06-20','09603','C400')
select count(*) from CUSTOMER where Num = 'C352'
getint(1) = 0
select count(*) from CUSTOMER where Num = 'C351'
getint(1) = 1
insert into ORDERS(Num, Date, Reference, Sender) values (457,'2008-06-20','07014','C351')
```

- getInt(1) returns the number of tuples satisfying the preceding query
- What foreign key constraint can you guess here?
  - ORDERS.Sender = CUSTOMER.Num

/SET /W&I

31-5-2010 PAGE 14

## Cardinality constraints: As defined

- Local variable
- Array of 20 elements
- DB attribute

```
01 LIST-DETAIL.
02 DETAILS OCCURS 20 TIMES
   INDEXED BY IND-DET.
03 REF-DET-STK PIC 9(5).
03 ORD-QTY PIC 9(5).
```

Hainaut, Hick, Henrard, Roland, Englebert

- ```
FD ORDER.
01 ORD.
02 ORD-CODE PIC 9(10).
02 ORD-CUSTOMER PIC X(12).
02 ORD-DETAIL PIC X(200).
```
- represent the same info MOVE LIST-DETAIL TO ORD-DETAIL.
  - Hence, ORD can be associated to not more than 20 details (and not less than 0 details – trivial)
    - As defined
    - What about the use?

/SET /W&I

31-5-2010 PAGE 15

## Cardinality constraints: As used

- Look for list traversals: e.g., reading data

```
SET IND-DET TO 1.
MOVE 1 TO END-FILE.
PERFORM READ-DETAIL
UNTIL END-FILE = 0 OR IND-DET = 21.
MOVE LIST-DETAIL TO ORD-DETAIL.
```

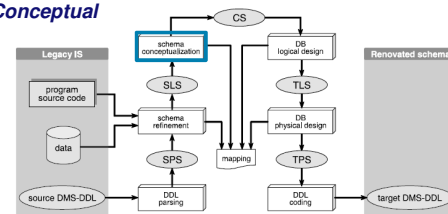
- Here: cardinality as used = cardinality as defined
- Not always the case

/SET /W&I

31-5-2010 PAGE 16

## Schema conceptualization

### Conceptual



- So far we only added complexity to the schema

/SET /W&I

31-5-2010 PAGE 17