

Methodology of Schema Integration for New Database Applications: A Practitioner's Approach

Joseph Fong¹

Kamalakar Karlapalem²

Qing Li²

Irene Kwan³

¹ Associate Professor, Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Hong Kong, email: csjfung@cityu.edu.hk, fax: (852)27888614, Tel: (852)27888498

² Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong

³ Department of Computing, Hong Kong Polytechnic University, Hong Kong

Methodology of Schema Integration for New Database Applications: A Practitioner's Approach

Abstract

A practitioner's approach to integrate databases and evolve them so as to support new database applications is presented. The approach consists of a joint bottom-up and top-down methodology; the bottom-up approach is taken to integrate existing database using standard schema integration techniques(B-Schema), the top-down approach is used to develop a database schema for the new applications(T-Schema). The T-Schema uses a joint functional-data analysis. The B-schema is evolved by comparing it with the generated T-schema. This facilitates an evolutionary approach to integrate existing databases to support new applications as and when needed. The mutual completeness check of the T-Schema against B-Schema derive the schema modification steps to be performed on B-Schema to meet the requirements of the new database applications. A case study is presented to illustrate the methodology.

keywords: schema integration, joint functional-data analysis, data transformation, extended entity relationship model, data schema, functional schema

1 Introduction

There has been a proliferation of databases in most organizations. These databases are created and managed by the various units of the organization for their own localized applications. Thus the global view of all the data that is being stored and managed by the organization is missing. Schema integration is a technique to present such a global view of an organization's databases. There has been a lot of work done on schema integration. Batini et al. (1986) and Özsu and Valduriez (1991) present surveys of work in this area. But all these techniques concentrate on integrating database schemas without taking into consideration of new database applications. This paper presents a practical approach to schema integration to support new database applications by comparing the existing databases against data requirements of the new applications. If the existing databases are inadequate to support new applications then they are evolved to support them.

In any schema integration methodology all the database schemas have to be specified using the same data model. The proposed approach uses an extended entity relationship(EER) data model. Therefore, the first step in the schema integration methodology is to translate a non-EER database schema to an EER database schema. A joint bottom-up and top-down approach for schema integration to support new database applications is proposed. The bottom-up approach is taken to integrate existing databases using standard schema integration techniques whereas the top-down approach is used to come up with the database schema for the new applications. The top-down approach uses the joint functional-data analysis. The B-schema generated by bottom-up approach is evolved by comparing it with the T-schema generated by the top-down approach. This facilitates a stream lined approach to evolve integrated databases to support new applications.

Conventional approaches that have been widely used in database community for database design can be classified as top-down, and are therefore suitable for designing databases from scratch to support new applications. On the other hand, research in the area of heterogeneous distributed databases over the last decade has emphasized on bottom-up approaches towards global schema derivation by integrating existing databases. These two kinds of approaches are complementary in many aspects, and thus can be combined into a unified framework for schema integration.

Fong et al. (1994) developed a hierarchical comparison scheme using three major criteria for comparing relationships in two schemas. The paper classified the relationship integration by taking

into account the degree of relationship, roles and structural constraints as the main features to guide the comparison of relationships. Fong (1992) applied information capacity equivalence as a measure of correctness for judging transformed schemas in schema integration. It presents a classification of common integration based on their operational goals and derive from them the instances level of equivalence of schemas after integration.

1.1 Top-down schema design techniques

Traditional database design has focused on data elements and their properties, and the approaches taken by database professionals were data-driven; the entire focus of the design process is placed on data and their properties(Korth and Silberschatz, 1991; Ullman, 1982; Elmarsr and Navathe, 1989). Typically, a data-driven(DD) approach first creates a conceptual schema by analyzing data requirements, which is then followed by logical and physical schema design; the applications that use the database will be developed after the database is created. An alternative kind of design that has been very popular in information systems design is termed as function-driven(Senn, 1989). In these kind of approaches, the main focus is on applications rather than data. More specifically, functional analysis starts with application requirements to generate functional schemas, which are then mapped into application specifications. These form the basis for the subsequent application program design. In functional analysis, databases are seen as isolated repositories of information, used by individual activities; the vision of data as a global resource of the enterprise is not present.

More recently, the idea of applying functional analysis techniques and concepts from traditional information systems area into conceptual database design has become increasingly popular, and has resulted in so-called Joint Data- and Function-Driven(JDFD) approach, which is more powerful than pure DD approach(Batini et al., 1986). As shown in Figure 1, JDFD produces the conceptual database structure and the function schema in parallel, so that the two design processes influence each other. More specially, the JDFD approach makes it possible to test whether data and function schemas are mutually consistent and complete. Note that both pure DD and JDFD types of approaches are used for designing new databases to support new applications.

1.2 Bottom-up schema integration techniques

Schema integration is a relatively recent problem that has appeared in the context of distributed, heterogeneous databases(Sheth and Larson, 1990; McLoed and Heimbigner, 1980). It takes place

in a bottom-up fashion, requiring that an integrated global schema be designed from the local schemas, which refer to existing databases. Figure 2 summarizes the schema integration activity which has as input the local schemas and local transactions, and has as its output the global schema as well as the specifications of data and query-mapping from global to local databases.

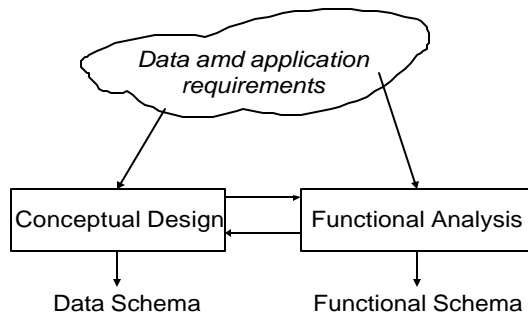


Figure 1 Joint data and function driven approach to database design

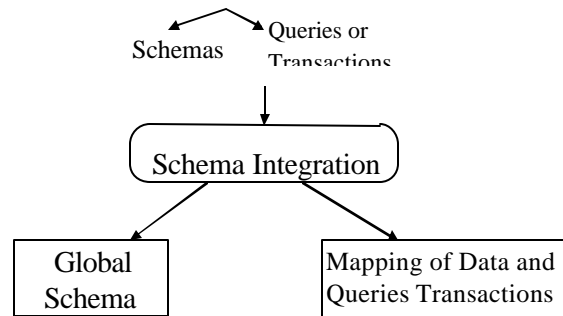


Figure 2 Bottom-up schema integration methodology

Though different researchers have proposed different solution procedures for conducting schema integration, they can be eventually considered to involve a mixture of the following activities: pre-integration, comparison, conformation, and integration (Batini et al., 1986). Hence schema integration in general involves such four steps. Note that the emphasis of such a bottom-up approach towards generating a global schema has been on identifying the correspondences and inter-schema properties, and that the subsequent integration of the local databases schemas is aimed to satisfy the criteria of completeness and correctness, minimality, and understandability. The bottom-up approach is to sum up the capacity of existing databases in one global schema.

1.3 Our approach

The above overviews of the two different kinds of approaches for database schema design and integration demonstrate that they have very different emphasis and focuses, reflecting their different problem domains. Yet they are complementary from a database integration point of view. Indeed it is the theme of this paper to combine these two sorts of approaches into a unified framework for the database integration purpose. The purpose is to develop a practical methodology to integrate, and to be able to evolve existing databases by considering the new requirements from the applications to be developed and supported. More specifically, the proposed mutual completeness checking methodology can be described as in the Figure 3 (Ozkarahan, 1990).

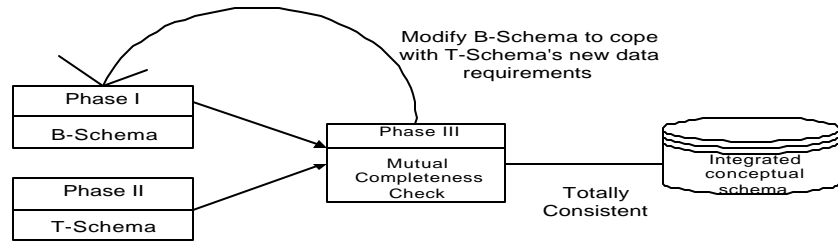


Figure 3 Context diagram for schema integration methodology

- Phase I. Integrate existing database schemas using a bottom-up schema integration methodology; the resultant global schema is called B-schema;
- Phase II. Design a conceptual schema using the Joint Data- and Function-Driven methodology for the applications to be developed and supported; the resultant schema is called T-schema.
- Phase III. Evaluate the B-schema by comparing it with the T-schema obtained in phase II, checking both consistency and completeness of B-schema with respect to the T-schema. If the two schemas are totally consistent, then the B-schema can be the integrated schema. Otherwise refine the B-schema by resolving any conflicts and incorporating any new concepts that are in T-schema but not in B-schema.

2 Phase I - B-Schema design

Algorithm:

Begin

For each existing database do /* step 1 */

If its conceptual schema does not exist

then reconstruct its conceptual schema in EER model by reverse engineer;

For each pair of existing databases' EER models of schema A and schema B do

begin

Resolve the semantics conflicts between the schema A and schema B; /* step 2 */

Merge the entities between the schema A and schema B; /* step 3 */

Merge the relationships between the schema A and schema B; /* step 4 */

end

In the first step, the conceptual schema of existing databases may or may not exist, and may also be out-of-dated. Thus, the first step of schema integration is to reconstruct the conceptual schema, using the EER models from the logical schema of the various data models (such as hierarchical, network or relational). The second step is to merge them into one EER model and

preserve all the semantics of the existing databases. This can be done iterative by merging each pair of EER models into one. However, the conflicts among the EER models must be resolved before merging. This process requires users intervention to solve the contradictions in the semantics. The third step is to associate the entities of different EER models by relationship. By doing so, the relationship of EER models can be captured in one EER model along with the semantics. The fourth step is to associate the relationships among EER models by absorbing one into another or by their related semantics such as subtype, generalization, aggregation and categorization.

Step 1. Reverse engineer logical schema to conceptual schema.

Input: the DDL(data description language) statements of the existing hierarchical, network or relational database to the reverse engineer

Output: the conceptual schema in EER model

Logical schema describes the data structure of databases. They only represent few semantics in the databases. Conceptual schema in EER model carries more semantics than logical schema. To rebuild an EER model from a logical schema is a reverse engineering process. User involvement is needed to confirm the implied semantics in the logical schema. Basically, the constraints and semantics in the logical schema will be preserved in the reconstructed EER model. Refer to (Fong, 1992; Batini et al., 1992) for detailed methodology in the process.

Step 2. Resolve conflicts among EER models.

Input: Schema A and B with entities E_s and attributes A_s in conflicts to each other on semantics

Output: Integrated schema X after data transformation

Step 2.1 Resolve conflicts on synonyms and homonyms

For each pair of (E_a, E_b) Do

 For each $x \in (\text{Attr}(E_a) \cap \text{Attr}(E_b))$, Do

 If $E_{a,x}$ and $E_{b,x}$ have different data types or sizes

 then x in E_a and E_b may be homonyms, let users clarify x in E_a and E_b ;

For each pair of (E_a, E_b) , Do

For each pair (x, y) , $x \in \text{Attr}(E_a)$ and $y \in \text{Attr}(E_b)$, Do

If $x \neq y$, and $E_a.x$ and $E_b.y$ have the same data type and size
then $((x,y)$ may be synonyms, let users clarify (x, y));

In some cases, we can consider the derived data as synonyms by matching their domain value based on conditions. For instance, two attributes A_a and A_b can be taken as synonyms by applying the following constraint rules in a stored procedure:

```
IF          Aa > 74 mark          /* student mark above 74 is a “A”
                                     grade */
      THEN  Ab = ‘A’ grade
IF          75 mark > Aa > 64 mark /* student mark above 64 is a “B” grade */
      THEN  Ab = ‘B’ grade
```

Step 2.2 Resolve conflicts on data types

Case 1: Conflict: an attribute appears as an entity in another schema

Resolution: For each pair of (E_a, E_b) , Do

```
IF  $x \in (A.A_b \cap B.E_b)$  /*attribute Ab in schema A appears as entity Eb
in schema B*/
```

```
THEN cardinality  $(E_a, E_b) \leftarrow n:1$ ;
```

Case 2: Conflict: a key appears as an entity in another schema

Resolution: For each pair of (E_a, E_b) , Do

```
IF  $x \in (\text{keys}(A.A_b) \cap B.E_b)$  /* entity key Ab in schema A appears as entity Eb in schema B
*/
```

```
THEN cardinality  $(E_a, E_b) \leftarrow 1:1$ ;
```

Case 3: Conflict: a component key appears as an entity in another schema

Resolution: For each pair of (E_a, E_b) , Do

```
IF  $(x \subset \text{keys}(A.A_b)) \cap B.E_b$  /*entity key component Ab in schema A appears as entity Eb in
schema B*/
```

```
THEN cardinality  $(E_a, E_b) \leftarrow m:n$ ;
```

Step 2.3 Resolve conflicts on key

Conflict: A key appears as a candidate key in another schema

Resolution: For each pair of (E_a, E_b) , Do

IF $x \in (\text{key}(A.A_c) \cap B.A_c)$

THEN Let users clarify x in E_a and E_b ;

Step 2.4 Resolve conflicts on cardinality

Conflict: Identical entities with different cardinality in two schemas

Resolution: For each pair of cardinality $(A.E_a, A.E_b)$ and $(B.E_a, B.E_b)$, Do

IF $(\text{cardinality}(A.E_a, A.E_b) = 1:1) \wedge (\text{cardinality}(B.E_a, B.E_b) =$

$1:n)$

THEN $\text{cardinality}(X.E_a, X.E_b) \leftarrow 1:n$;

For each pair of cardinality $(A.E_a, A.E_b)$ and $(B.E_a, B.E_b)$, Do

IF $(\text{cardinality}(A.E_a, A.E_b) = 1:1 \text{ or } 1:n) \wedge (\text{cardinality}(B.E_a,$

$B.E_b) = m:n)$

THEN $\text{cardinality}(X.E_{a1}, X.E_{a2}) \leftarrow m:n$;

Step 2.5 Resolve conflicts on weak entities

Conflict: A strong entity appears as a weak entity in another schema

Resolution: For each pair of (E_a, E_b) , and their relationships: R in schema A and B ,

Do IF $((x \in \text{key}(A.E_a)) \cap ((x \notin \text{key}(E_{a2}))) \wedge ((x \in \text{key}(B.E_a)) \cap ((x \subset \text{key}(B.E_b))))$

THEN $\exists (x \in \text{key}(E_a)) \cap ((x \subset \text{key}(E_b)))$

In other words, the key of the weak entity must concatenate with its strong entity key.

Step 2.6 Resolve conflicts on subtype entities

Conflict: A subtype entity appears as a supertype entity in another schema

Resolution: For each pair of (E_a, E_b) Do

IF $((\text{domain}(A.E_a) \subseteq \text{domain}(A.E_b)) \wedge ((\text{domain}(B.E_b) \subseteq$

$\text{domain}(B.E_a)))$

THEN $\text{cardinality}(X.E_a, X.E_b) \leftarrow 1:1$ /* $A.E_a = E_a$ in A schema */

The above step 2.1 to 2.6 can be illustrated in Figure 4.

Step	Before data transformation	After data transformation
2.1 EER model with synonyms and homonyms	<p>Schema A Schema B</p>	<p>Schema X <small>Synonyms: E_a, E_b</small></p>
2.2 EER models in data type conflicts	<p>Case 1</p> <p>Schema A Schema B</p> <p>Case 2</p> <p>Schema A Schema B</p> <p>Case 3</p> <p>Schema A Schema B</p>	<p>Case 1 Schema X</p> <p>Case 2 Schema X</p> <p>Case 3 Schema X</p>
2.3 EER models in key conflicts resolved with users confirmation	<p>Schema A Schema B</p>	<p>Schema X</p>
2.4 EER models in cardinality conflict	<p>Schema A</p> <p>Schema B</p>	<p>Schema X</p>
2.5 EER models in weak entity conflict	<p>Schema A</p> <p>Schema B</p>	<p>Schema X</p>
2.6 EER models in subtype conflicts (where → means isa relationship)	<p>Schema A</p> <p>Schema B</p>	<p>Schema X</p>

Figure 4 Examples on conflicts resolutions by data transformation

Step 3 Merge entities

Input: existing entities in schema A and B

Output: merged (connected) entities in (integrated) schema X

Step 3.1 Merge entities by union

Condition: Identical entities with different attributes in two schemas

Integration: For each pair of (E_a, E_b) , Do

```
IF  $((\text{domain}(A.E_a) \cap \text{domain}(B.E_b)) \neq 0)$ 
  THEN  $\text{domain}(X.E_a) \leftarrow \text{domain}(A.E_a) \cup \text{domain}(B.E_a);$ 
```

Step 3.2 Merge entities by generalization

Case 1: Condition: Entities with same attributes appear in two schemas, but instance of the first entity in one schema cannot appear in another schema (i.e. disjoint generalization)

Integration: For each pair of (E_a, E_b) , Do

```
IF  $((\text{domain}(E_a) \cap \text{domain}(E_b)) \neq 0) \wedge ((x \in \text{instance}(E_a)) \wedge (x \notin \text{instance}(E_b)))$ 
   $\wedge ((y \in \text{instance}(E_b)) \wedge (y \notin \text{instance}(E_a)))$ 
  THEN begin  $\text{domain}(E_x) \leftarrow \text{domain}(E_a) \cap \text{domain}(E_b)$ 
     $((x \in \text{instance}(X.E_a) \wedge (x \notin \text{instance}(X.E_b)))$ 
     $((y \in \text{instance}(X.E_b)) \wedge (y \notin \text{instance}(X.E_a)));$ 
  end
```

Case 2: Condition: Entities with same attributes appear in two schemas, but instance of the first entity in one schema can appear in another schema (i.e. overlap generalization)

Integration: For each pair of (E_a, E_b) , Do

```
IF  $((\text{domain}(E_a) \cap \text{domain}(E_b)) \neq 0)$ 
  THEN  $\text{domain}(E_x) \leftarrow \text{domain}(E_a) \cap \text{domain}(E_b);$ 
```

Step 3.3 Merge entities by subtype relationship

Condition: Two subtype related entities appear in two different schemas

Integration: For each pair of (E_a, E_b) , Do

IF $\text{domain}(E_a) \subseteq \text{domain}(E_b)$

THEN E_a isa E_b ; /* entity E_a is a subset of entity E_b */

Step 3.4 Merge entities by aggregation

Condition: A relationship in one schema is related to an entity in another schema

Integration: For each pair of entity A and relationship B, Do

IF $(\exists \text{ MVD: } R_b \rightarrow \rightarrow E_b)$ /* MVD means multivalued dependency (Fong, 1992) */

THEN begin $E_x \leftarrow (E_{b1}, R_b, E_{b2})$ /* aggregation */

cardinality $(E_x, E_a) \leftarrow$

1:n;

end

Step 3.5 Merge entities by categorization

Condition: One entity in one schema is subtype related to two entities in another schema

Integration: For each group of (E_{a1}, E_{a2}, E_b) , Do

IF $(\text{instance}(E_b) \subseteq \text{instance}(E_{a1})) \vee (\text{instance}(E_b) \subseteq \text{instance}(E_{a2}))$

THEN begin $E_c \leftarrow (E_{a1}, E_{a2})$ /* categorization */

$(\text{instance}(E_b) \text{ isa } \text{instance}(E_{x1})) \vee$

$(\text{instance}(E_b) \text{ isa } \text{instance}(E_{x2}))$; /* E_b is subset to E_{a1} or E_{a2} */

end

Step 3.6 Merge entities by implied binary relationship

Condition: An identical data item appears in different data types in two schemas

Integration: For each pair of (E_a, E_b) , Do

IF $x \in (A.A_d \cap \text{key}(B.A_d))$

THEN Cardinality $(E_a, E_b) \leftarrow n:1$;

Condition: Two identical data items appear in different data types in two schemas

Integration: For each pair of (E_a, E_b) , Do

IF $(x \in (A.A_d \cap \text{key}(B.A_d))) \wedge (y \in (\text{key}(A.A_c) \cap B.A_c))$

THEN Cardinality $(E_a, E_b) \leftarrow 1:1;$

The above step 3.1 to 3.6 can be illustrated in Figure 5.

Step	Before entities merge	After entities merge
3.1 Merge EER models by union	Schema A Schema B $\frac{A_b}{A_c}$ E_a E_a $\frac{A_b}{A_d}$	Schema X E_a $\frac{A_b}{A_c}{A_d}$
3.2 Merge EER models by generalization	Schema A Schema B $\frac{A_c}{A_d}$ E_a E_b $\frac{A_c}{A_e}$	Schema X E_x $\frac{A_c}{A_d}{A_e}$ d = disjoint generalization o = overlap generalization E_a $\frac{A_c}{A_d}$ E_b $\frac{A_c}{A_e}$
3.3 Merge EER models by subtype	Schema A Schema B E_a E_b	Schema X $E_a \rightarrow E_b$
3.4 Merge EER models by aggregation	Schema A MVD: $R_d \twoheadrightarrow E_a$ E_a Schema B $E_{b1} \xrightarrow{1} R_b \xrightarrow{n} E_{b2}$	Schema X $E_{b1} \xrightarrow{1} R_b \xrightarrow{n} E_{b2}$ E_x $E_x \xrightarrow{1} R \xrightarrow{n} E_a$
3.5 Merge EER models by categorization	Schema A Schema B $\frac{A_c}{A_d}$ E_{a1} E_{a2} $\frac{A_c}{A_e}$ Schema C $\frac{A_c}{A_d}$ E_a	Schema X $\frac{A_c}{A_d}$ E_{x1} E_c E_{x2} $\frac{A_c}{A_e}$ E_c $\frac{A_c}{A_d}$ E_a
3.6 Merge EER models by implied relationship	Case 1 Schema A A_d $\frac{A_c}{A_d}$ E_a E_b Case 2 Schema B A_d $\frac{A_c}{A_d}$ E_a E_b $\frac{A_d}{A_c}$	Case 1 Schema X $\frac{A_c}{A_d}$ $E_a \xrightarrow{n} R \xrightarrow{1} E_b \frac{A_d}{A_c}$ Case 2 Schema X $\frac{A_c}{A_d}$ $E_a \xrightarrow{1} R \xrightarrow{1} E_b \frac{A_d}{A_c}$

Figure 5 Examples on entities merge into an integrated schema

Step 4 Merge relationships

Input: existing relationships in schema A and B

Output: merged (connected) relationships in integrated schema X

Step 4.1 Merge relationships by subtype relationship

Case 1: Condition: Two relationship R_a, R_b are in the same role with different level of participation (Elmarsr and Navathe, 1989)

Integration: For each pair of (R_a, R_b) , Do

IF $((\text{Cardinality}(A.E_a, A.E_b) = 1:n) \wedge (\text{Cardinality}(B.E_a, B.E_b) = 1: (0,n)))$

THEN begin $\exists E_c$

$\text{Cardinality}(E_a, E_c) \leftarrow$

1:n

$E_c \text{ isa } E_b$

end

Case 2: Condition: Two relationships have different semantics but with intersecting relationship

Integration: For each pair of relationships (R_a, R_b) , Do

IF $((\text{domain}(E_a) \cap \text{domain}(E_b)) \neq 0)$

THEN begin $E_c \leftarrow R_a$

$E_d \leftarrow R_b$

$E_c \text{ isa } E_b$

$E_d \text{ isa } E_b$

end

Step 4.2 absorbing lower degree relationship into a higher degree relationship

Condition: The semantic of a lower degree relationship is implied in the semantics of a higher degree relationship

Integration: For each pair of (R_a, R_b) , Do

IF $((\text{domain}(R_b) \subset \text{domain}(R_a)) \wedge (\text{degree}(R_b) < \text{degree}(R_a)))$

THEN $X.R_b \leftarrow R_a$

Note: in case $\text{domain}(R_b) \subset \text{domain}(R_a)$ provided that certain constraints are met for reason of their semantics, these constraints must be preserved in the integrated schema.

The above step 4.1 to 4.2 can be illustrated in Figure 6.

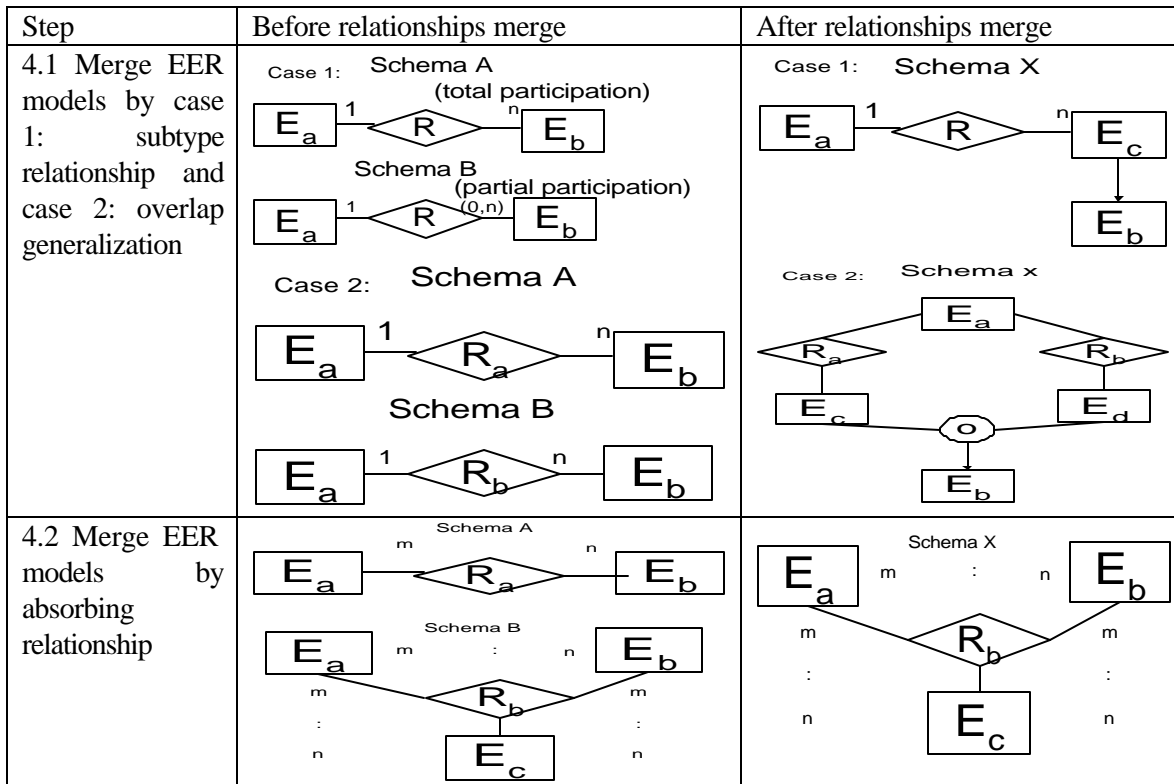


Figure 6 Merge relationships for schema integration

4 Phase II - T-Schema Design

T-Schema design methodology is used to derive with a database schema for the data and process requirements of new set of applications. This methodology is based on joint functional and data analysis. It consists of evaluating the data and process requirements of the applications and incrementally developing the database schema by applying transformations to it. There are two approaches for this joint functional and data analysis: data driven and function driven(Ozkarahan, 1990). This methodology chooses function driven for its simplicity. Its procedure is to decompose a main process into sub-processes in a DFD, i.e. functional schema. Then it refines the data analysis for each sub-process. The sum of these data analysis is a refined data analysis, i.e. data schema, for the refined function analysis. They can be described as follows:

Step 1 Identify a main process and its data requirements for a new application

Define a main process P in a DFD and its data requirements in an EER model.

Step 2 Refine the new application's functional schema using a DFD and data schema using in the EER model.

Decompose P into sub-processes P_1, P_2, \dots, P_n in a functional schema;

For each sub-process P_i , Do

IF P_i requires $E_{i1}, E_{i2}, \dots, E_{ik}$ and $R_{i1}, R_{i2}, \dots, R_{ij}$ /*Rs are relationships between entities E_s */

THEN integrate $E_{i1}, E_{i2}, \dots, E_{ik}$ and $R_{i1}, R_{i2}, \dots, R_{ij}$ into a refined EER model

Case 1: The data analysis of two sub-processes can be integrated by binary relationship

Case 2: The data analysis of two sub-processes can be integrated by generalization

Case 3: The data analysis of two sub-processes can be integrated by subtype relationship

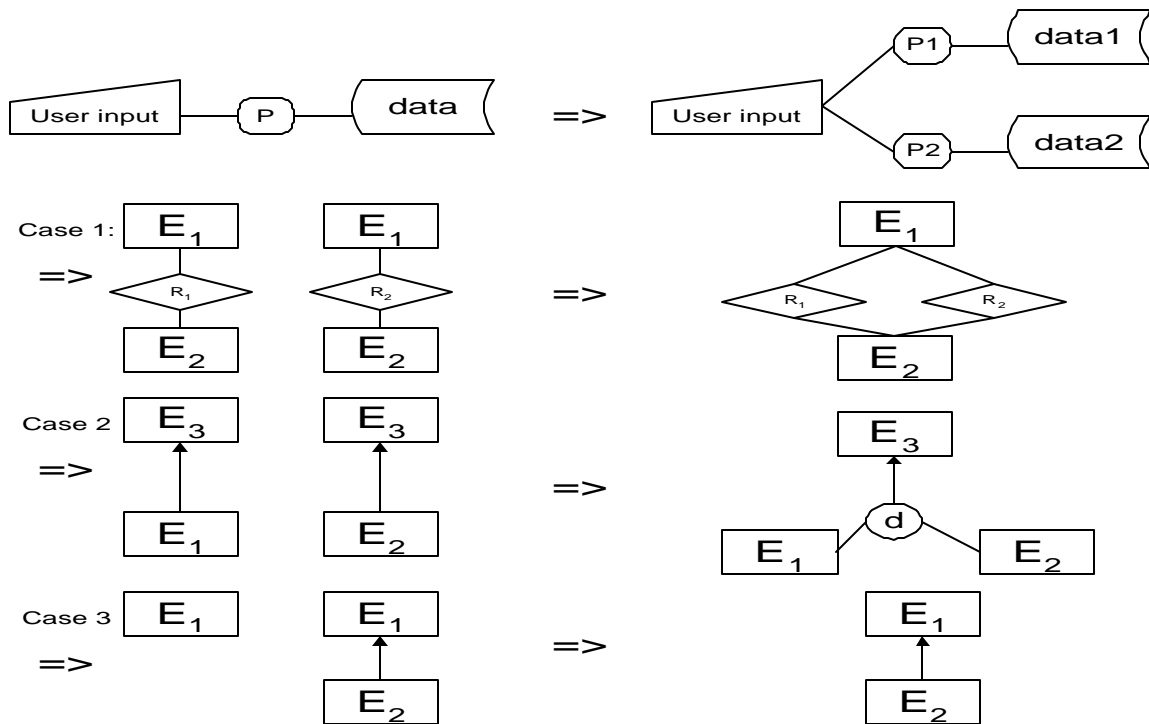


Figure 7 Refine Data Schema by process

5 Phase III - Mutual Completeness Check

This system is used to compare the B-schema with the T-schema and provides a database schema that supports both the existing and new applications. The input to this system is T-Schema and B-Schema and the output is the integrated database schema.

Notations:

T_i: an entity in T-Schema, $i = 1, 2, \dots, m$.

B_p: an entity in B-Schema, $p = 1, 2, \dots, n$.

$\text{Attr}(E)$: a set of attributes in entity E .
 $R(E_1, E_2, \dots, E_n)$: a relationship participated by E_1, E_2, \dots, E_n .
 $E_1 = E_2$: $\text{Attr}(E_1) = \text{Attr}(E_2)$, E_1 and E_2 are equivalent.
 $E_1 \subset E_2$: $\text{Attr}(E_1) \supset \text{Attr}(E_2)$, E_1 is a subclass of E_2 .
 $E_1 \supset E_2$: $\text{Attr}(E_1) \subset \text{Attr}(E_2)$, E_1 is a superclass of E_2 .
 $E_1 \cap E_2 = \emptyset$: $\text{Attr}(E_1) \cap \text{Attr}(E_2) = \emptyset$, E_1 and E_2 are disjoint.
 $E_1 \cap E_2 \neq \emptyset$: $\text{Attr}(E_1) \cap \text{Attr}(E_2) \neq \emptyset$, E_1 and E_2 are overlapping.
 \longrightarrow : denote "is-a" relationship

Algorithm:

begin

Resolve the naming conflicts; /* similar to the step 1 in B-schema construction */

Transform all relationships with degree > 2 into a set of binary relationships;

Repeat

For each relationship/entity in T-Schema

compare and modify B-Schema by Rule 1

Until there is no more evolution;

For those entities haven't been added in B-Schema

apply Rule 2 to integrate it;

end

The algorithm is complete because it checks all relationships and entities in both schema.

Rules:

Rule 1: Comparison of relationships and entities

Without loss of generality, we can always transform a higher degree of relationship into binary relationships. Thus, we just consider the comparison of binary relationships here.

Let $R_T = R(T_i, T_j)$, $i, j = 1, 2, \dots, m$, and $i \neq j$

$R_B = R(B_p, B_q)$, $p, q = 1, 2, \dots, n$, and $p \neq q$

$T, T' \in \{T_i, T_j\}$, $T \neq T'$

$B, B' \in \{B_p, B_q\}$, $B \neq B'$

The representations in ER diagram are shown in Figure 8.



Figure 8 a) A relationship in T-Schema b) A relationship in B-Schema.

While comparing the relationships, we consider two cases:

i) $R_T = R_B$ (i.e. same relationship name).

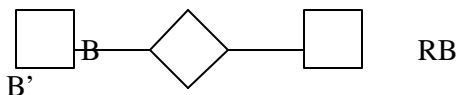
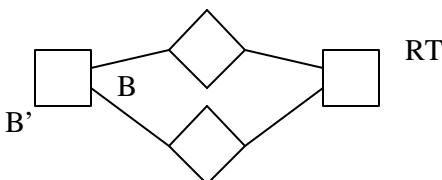
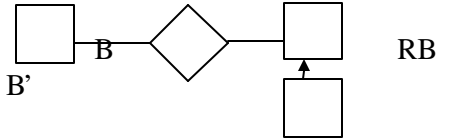
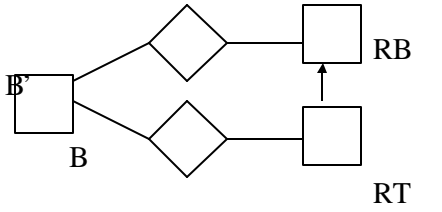
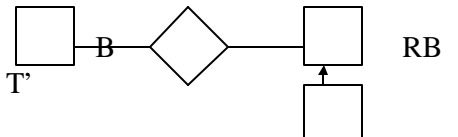
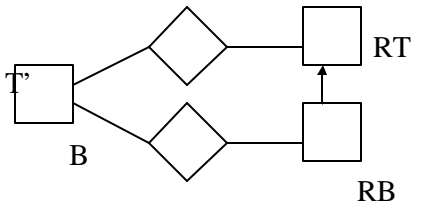
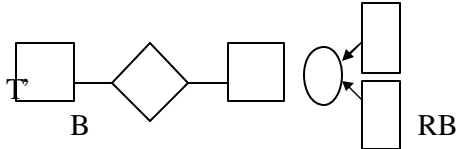
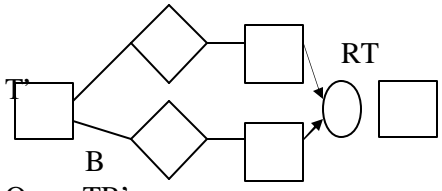
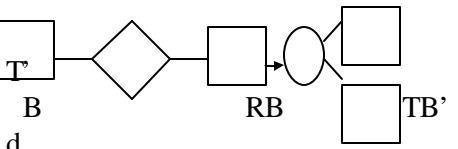
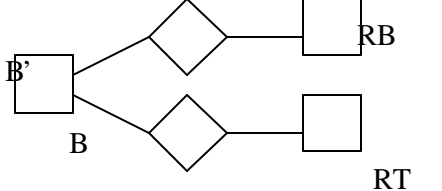
R_T can be usually eliminated by inheritance with entities have a subclass/superclass relation or overlap from two schema.

ii) $R_T \neq R_B$ (i.e. different relationship name).

RT is usually added in B-Schema and cannot be eliminated in this case.

We will give a brief explanation in 1.1 as an example. Other cases are mostly similar and should be intuitive.

1.1 If $T = B$

	$RT = RB$	$RT \neq RB$
a. $T' = B'$	 <p>RT has been already contained in B-Schema.</p>	 <p>RT cannot be eliminated.</p>
b. $T' \subset B'$	 <p>RT is contained through inheritance.</p>	 <p>RT cannot be inherited.</p>
c. $T' \supset B'$	 <p>Similar to above.</p>	 <p>Similar to above.</p>
d. $T' \cap B' \neq \emptyset$	 <p>$O = \text{overlap generalization}$ Since T' and B' overlap, we can make it as a generalization.</p>	 <p>Generalize T' and B'.</p>
e. $T' \cap B' = \emptyset$	 <p>$d = \text{disjoint generalization}$</p>	

	Since $RT=RB$, there must be some kind of relation between T' and B' though they are disjoint.	For $RT \neq RB$, T' and B' are disjoint, we can treat it separately.
--	---	--

1.2 If $T \subset B$

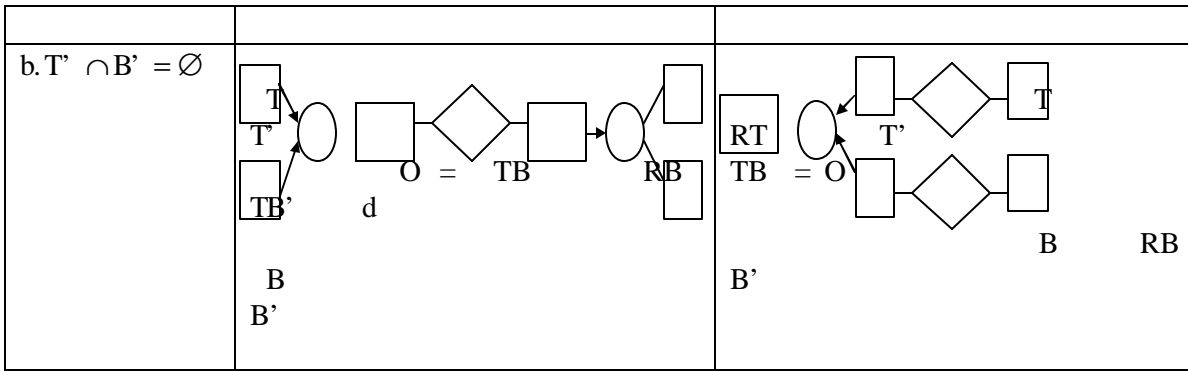
	$RT = RB$	$RT \neq RB$
a. $T' \subset B'$		
b. $T' \supset B'$	<p>RT</p> <p>T'</p> <p>Create a new entity E which is a subclass of B.</p>	
c. $T' \cap B' \neq \emptyset$		
d. $T' \cap B' = \emptyset$		

1.3 If $T \supset B$

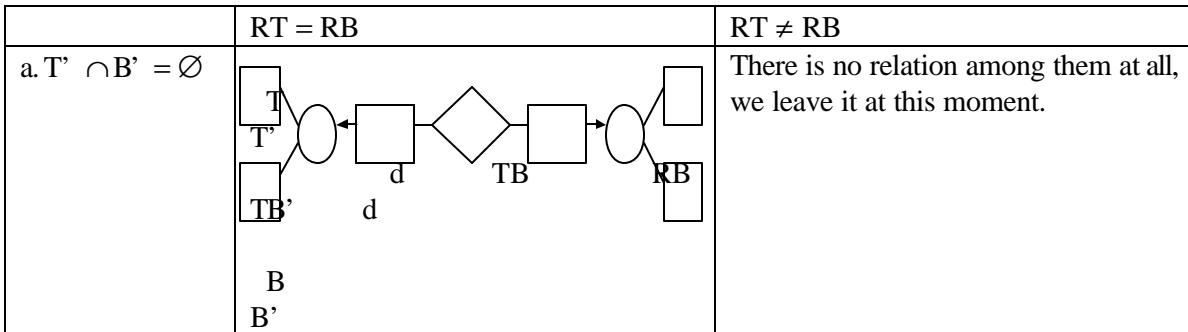
	RT = RB	RT ≠ RB
a. $T' \subset B'$	<p>RT T</p> <p>RB</p> <p>B'</p>	<p>RT</p> <p>B'</p> <p>B</p> <p>RB</p>
b. $T' \supset B'$	<p>RT</p> <p>B'</p> <p>B</p>	<p>RT</p> <p>RB</p> <p>B'</p> <p>B</p>
c. $T' \cap B' \neq \emptyset$	<p>T'</p> <p>T</p> <p>TB' = O</p> <p>B'</p> <p>B</p> <p>RB</p>	<p>T'</p> <p>T</p> <p>O = TB'</p> <p>B'</p> <p>B</p> <p>RB</p>
d. $T' \cap B' = \emptyset$	<p>T'</p> <p>T</p> <p>TB'</p> <p>B'</p> <p>B</p> <p>RB</p> <p>d</p>	<p>T'</p> <p>T</p> <p>B'</p> <p>B</p> <p>RB</p>

1.4 if $T \cap B \neq \emptyset$

	RT = RB	RT ≠ RB
a. $T' \cap B' \neq \emptyset$	<p>T</p> <p>T</p> <p>TB' = O</p> <p>B</p> <p>B'</p> <p>O = TB</p> <p>RB</p>	<p>RT</p> <p>TB</p> <p>O = TB'</p> <p>RB</p> <p>B'</p> <p>B</p>

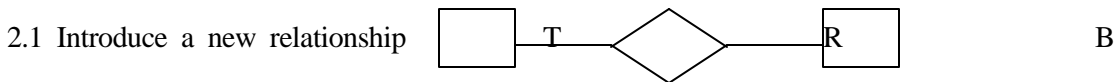


1.5 if $T \cap B = \emptyset$



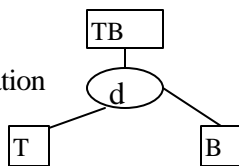
Rule 2: Handling the isolated entities

Users have to choose an entity in T-Schema for the combination in following ways.

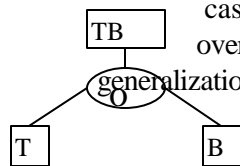


2.2 Introduce a new generalization

case 1:
disjoint
generalization

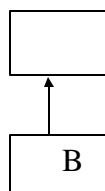


case 2:
overlap
generalization

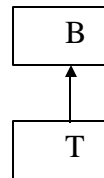


2.3 Introduce a subclass relationship

case 1



T



case 2

6 A Case Study

In a bank, there are existing databases with different schemas: one for the customers, another for the mortgage loan contracts and a third for the index interest rate. They are used by various application in the bank. However, there is a need to integrate them together for an international banking loan system. the followings are the three source schemas shown in Figure 9.

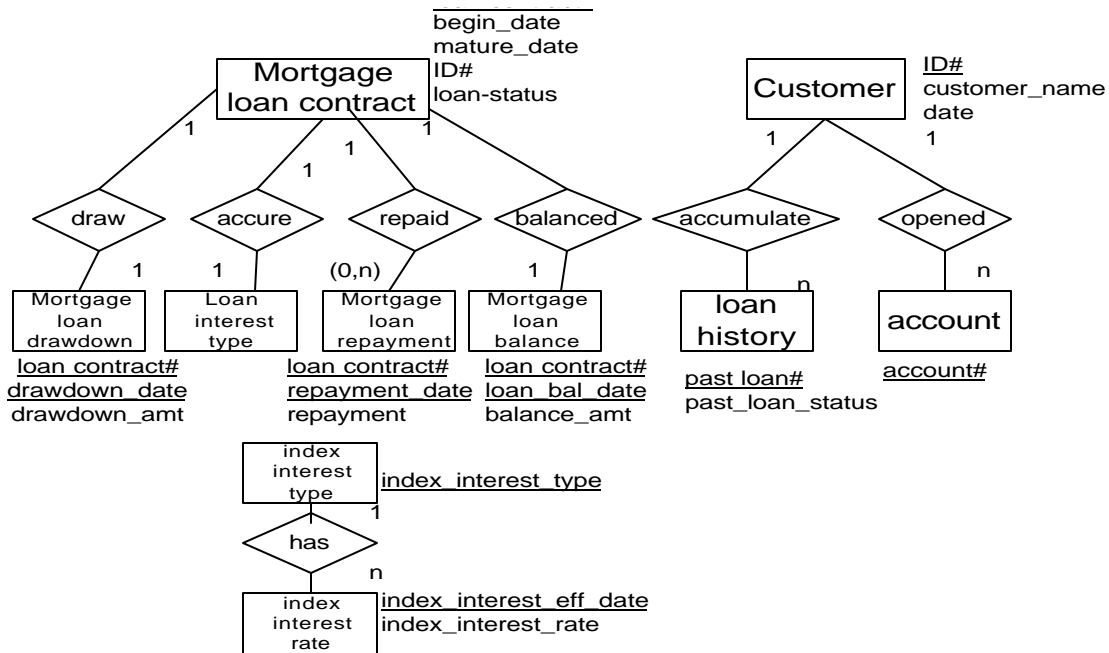


Figure 9 EER models of the Mortgage Loan System

Step 1. B-Schema Analysis.

By following the methodology, in the first iteration, the mortgage loan schema will be merged with the customer schema in the first iteration as follows:

- There are two synonyms: Loan_status and Balance_amt such that the Loan_status can be derived from the Balance_amt. As a result, we can replace Loan_status by Balance_amt with a stored procedure to derive Loan_status value from Balance_amt.
- There is an implied relationship between these two schemas such that ID# used as attribute in loan schema but as an entity key in customer schema. Thus, we can derive cardinality from the implied relationship between these entities, and integrate the two schemas into one EER model.

In the second iteration, the intermediate integrated schema will be merged with the index rate schema. There is a overlap generalization between the two schemas such that a loan must be on fixed and index interest rate. Thus, by joining the integrated schema and the index rate schema with overlap generalization, the two schemas can be integrated in a B-schema in Figure 10.

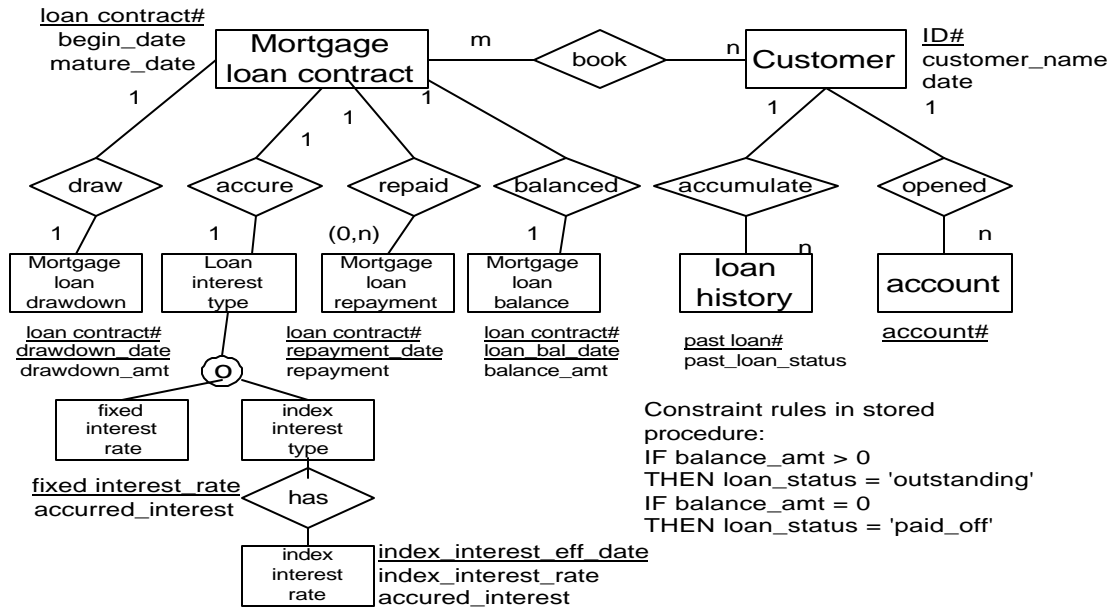


Figure 10 Integrated B-schema for mortgage loan

Step 2. T-Schema analysis

T-Schema from the joint data and functional analysis involving the data flow diagram captures the new application process requirements and the data analysis takes care of the data requirements for the new application. In this case study, here is a functional bank car loan system. There is one main process of car loan processing which can be decomposed into five sub-processes in the loan system: car loan booking, car loan index rate update, car loan repayment, car loan balance and car loan index interest type update as shown in Figure 11.

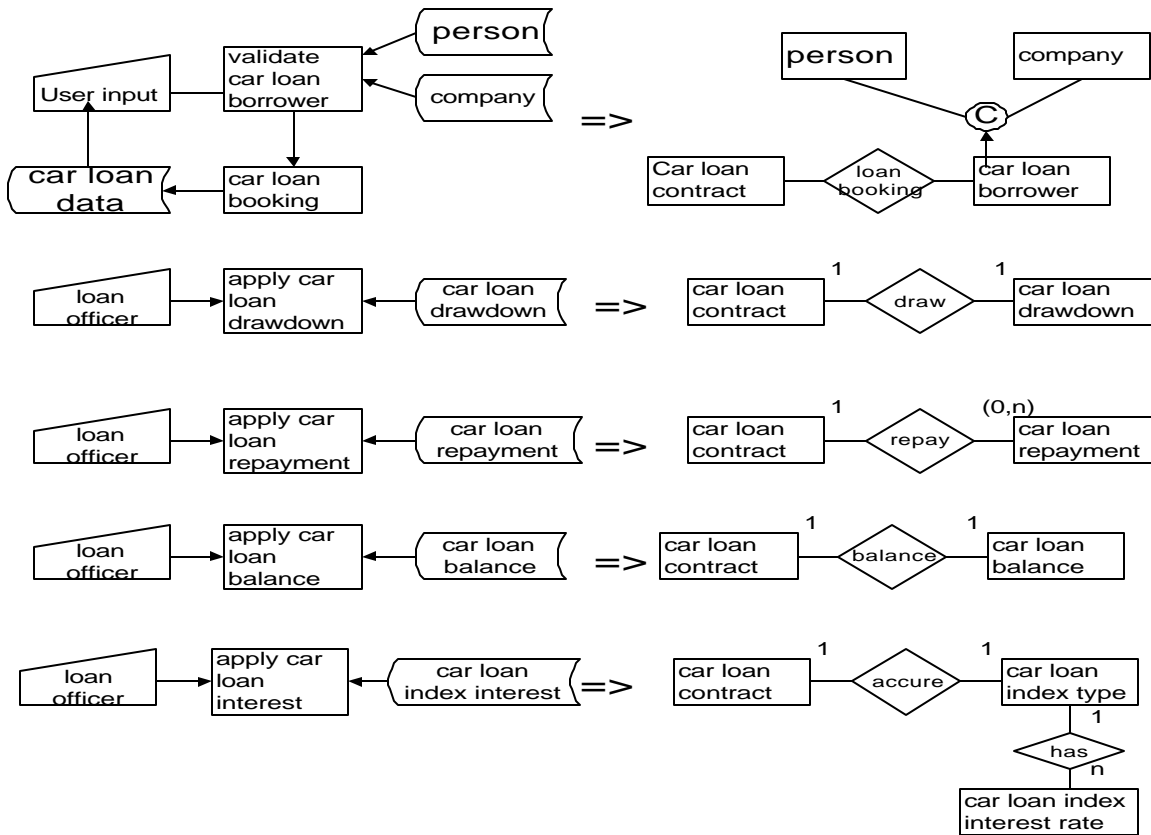


Figure 11 T-schema sub-processes and their data

2.2 Data analysis

After the refinement of the functional schema, the refined data schema can be shown in Fig 12.

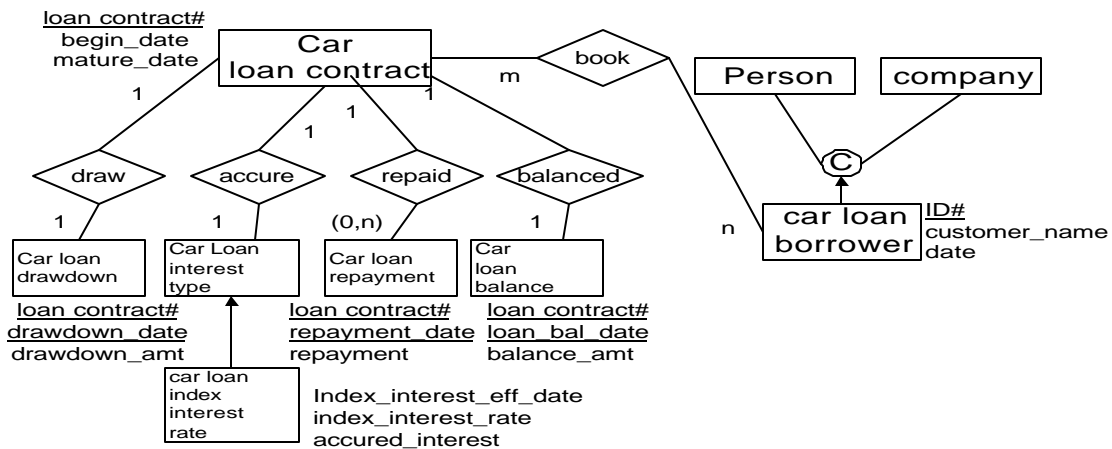


Figure 12 T-schema for car loan system

Step 3. Mutual completeness check.

The derived data schema contains a new data requirement of overseas customer, but does not contain an account record with respect to the B-schema. After the comparison between the data

schema and the B-schema, we need to solve synonym conflict such that customer and car loan borrowers are synonyms, and the date of customer and the date of the car loan borrower are homonyms as cusotmer_record_date and car_registration_date in Figure 13.

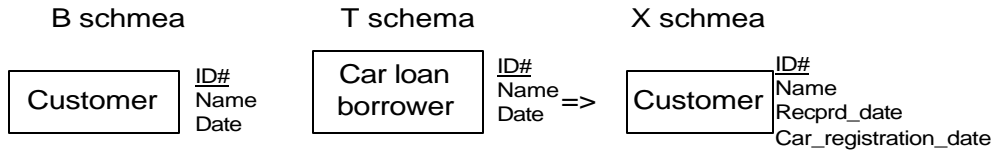
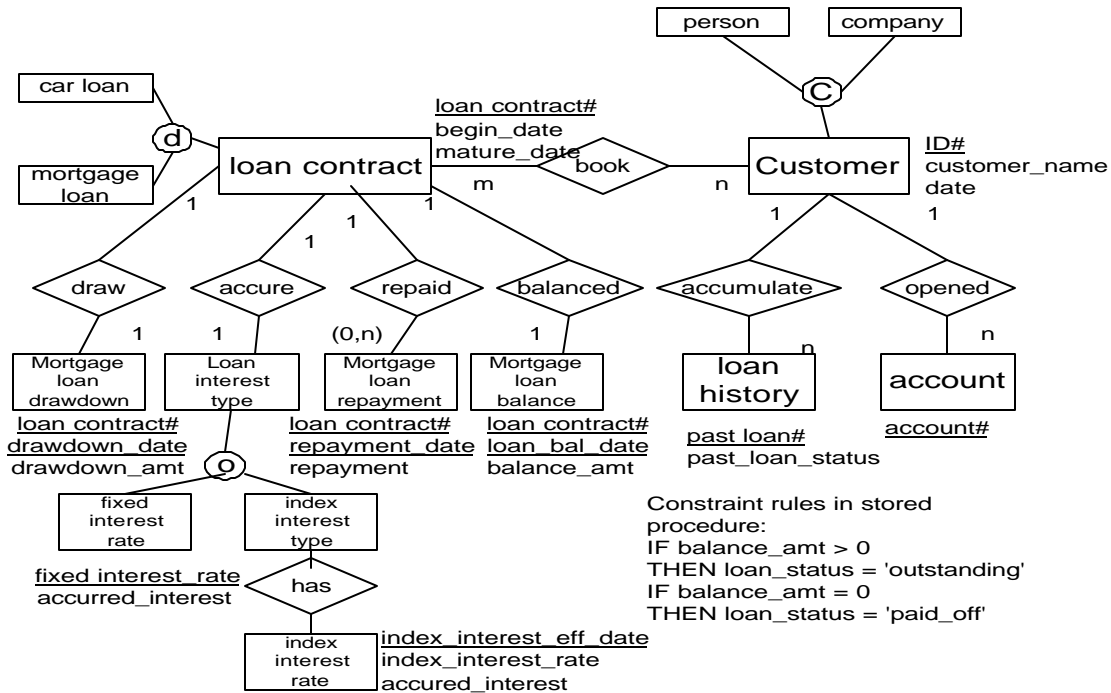


Figure 13 Synonyms and Homonyms conflict solution

The refined B-schema contains the existing data requirements with Account record plus a new data requirements of different kinds of customer and two disjoint loans: mortgage loan and car loan, as shown in Figure 14.



7 Conclusion **Figure 14 Integrated schema for new application**

A mutual consistency checking methodology has been presented to integrate existing databases to support new applications. The approach is featured with a combined methodology which uses both traditional bottom-up method, and top-down method. The rational behind such a combined methodology of conducting database schema integration is the fact that top-down approach can complement the bottom-up approach of schema integration by taking into account both data and function requirements of new applications. Thus the integrated schema can readily support new applications while continuing to serve existing ones.

It is important to extend the scope of conflict resolution performed by the mutual consistency checker. When the difference between the T-schema, and B-schema is more significant, the system may need to evolve the integrated schema and propagate the changes back to the individual local schemas. Likewise, the T-schema enables us to filter out unused ones from the integrated schema; a view mechanism is useful here and can be developed on top of the integrated schema. For further research, the feasibility of automating much of the schema analysis and integration process should be investigated by developing a set of heuristic rules which can be utilized by the expert system component.

Reference

- Batini, C., Lenzerini, M. and Navathe, S.(1986) A Comparative Analysis of Methodologies for Database System Integration, ACM Computing Survey, Vol 18, No 4.
- Batini, C., Ceri, S. and Navathe, S.(1992) Conceptual Database Design: An Entity-Relationship Approach, The Benjamin/Cummings Publishing Company, Inc, p 227-243.
- Elmasri R. and Navathe, S.(1989) Fundamentals of Database Systems, Benjamin/Cummings pub.
- Fong, J.(1992) Methodology for schema translation from hierarchical or network into relational, Information and Software Technology, p159-174, Vol 34, No 3, 1992.
- Fong, J., Karlapalem, K. and Li, Q.,(1994)A practitioners approach to schema integration for new database applications, Proceeding of the 5th International Hong Kong Computer Society Database Workshop, p136-154.
- Korth, H. and Silberschatz, A.(1991)Database System Concepts (2nd edition), McGraw-Hill.
- McLoed, D. and Heimbigner, D.(1980) A Federated Architecture for Database Systems, Proceedings of the AFIPS National Computer Conference, vol 39, AFIPS Press, Arlington, VA.
- Ozkarahan, E.(1990)Database Management: Concepts, Design and Practice, Prentice-Hall.
- Özsu, M. and Valduriez, P.(1991)Principles of Distributed Database Systems, Prentice-Hall International Edition, p428-430.
- Senn, J.(1989)Analysis & Design of Information Systems (2nd edition), McGraw-Hill.
- Sheth A. and Larson, J.(1990)Federated Database Systems for Managing Distributed Heterogeneous, and Autonomous Databases, ACM Computing Survey, Vol 22, No 3.
- Ullman, J.(1982)Principles of Database Systems (2nd edition), Computer Science Press