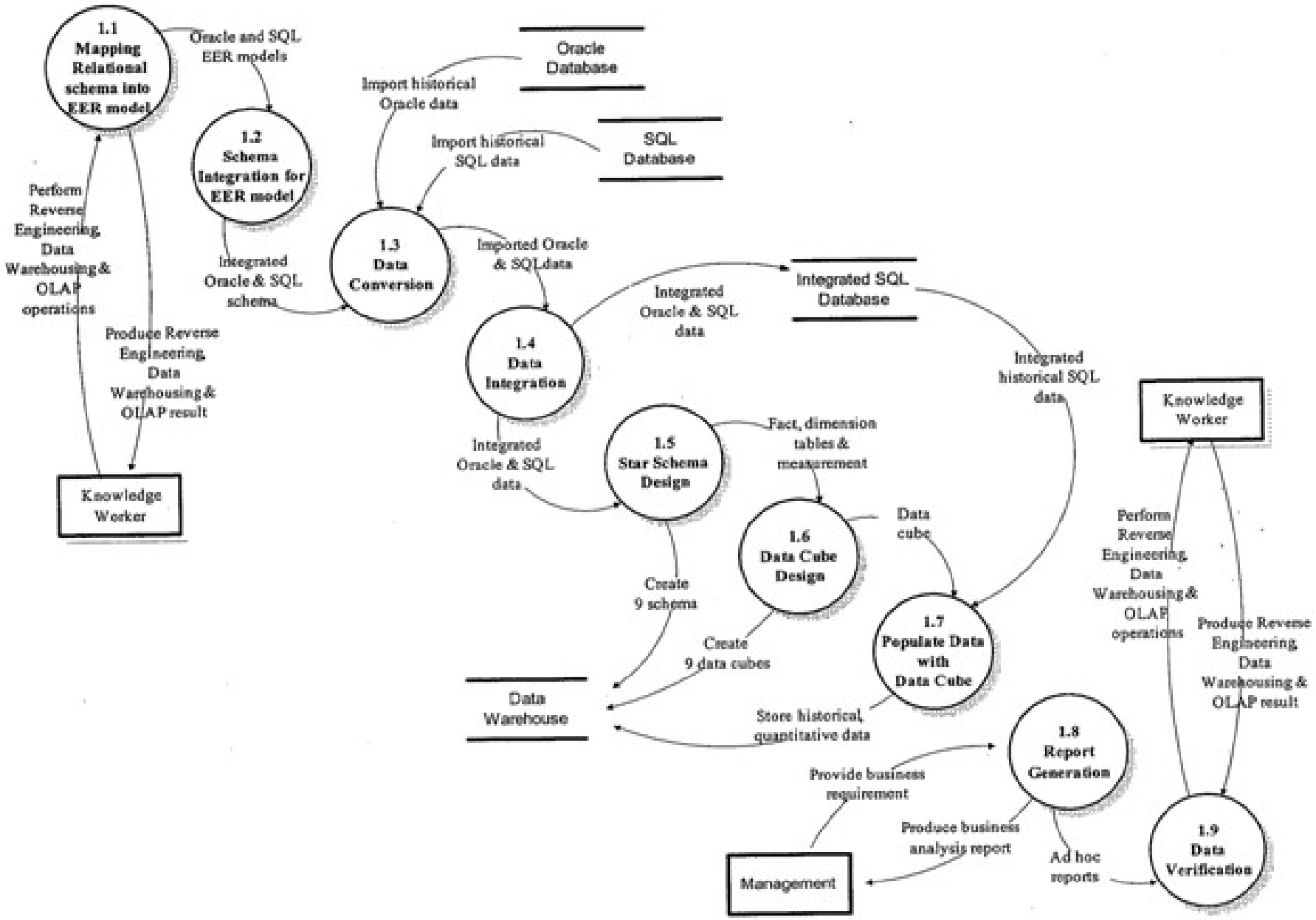
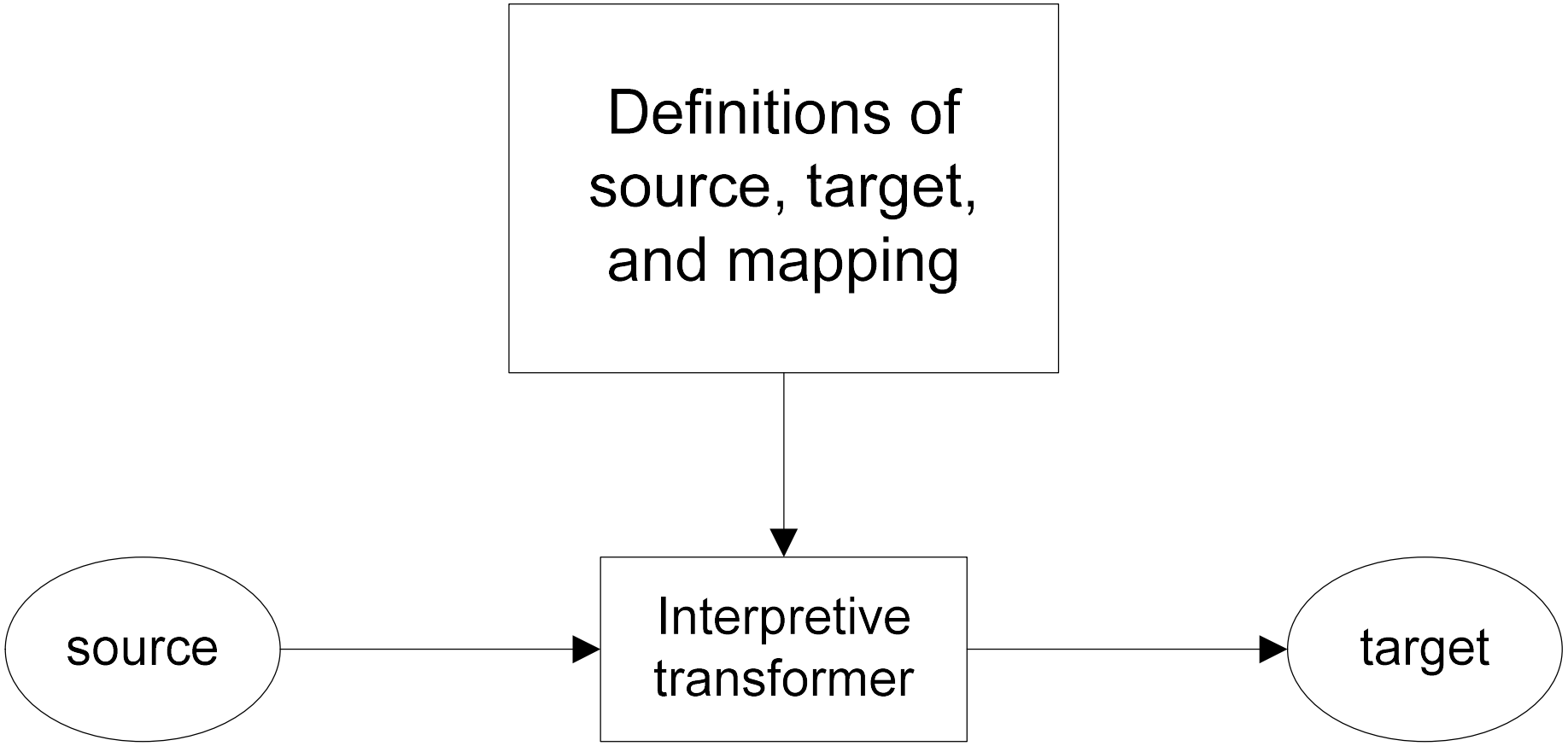
Methodology for Data warehousing with OLAP



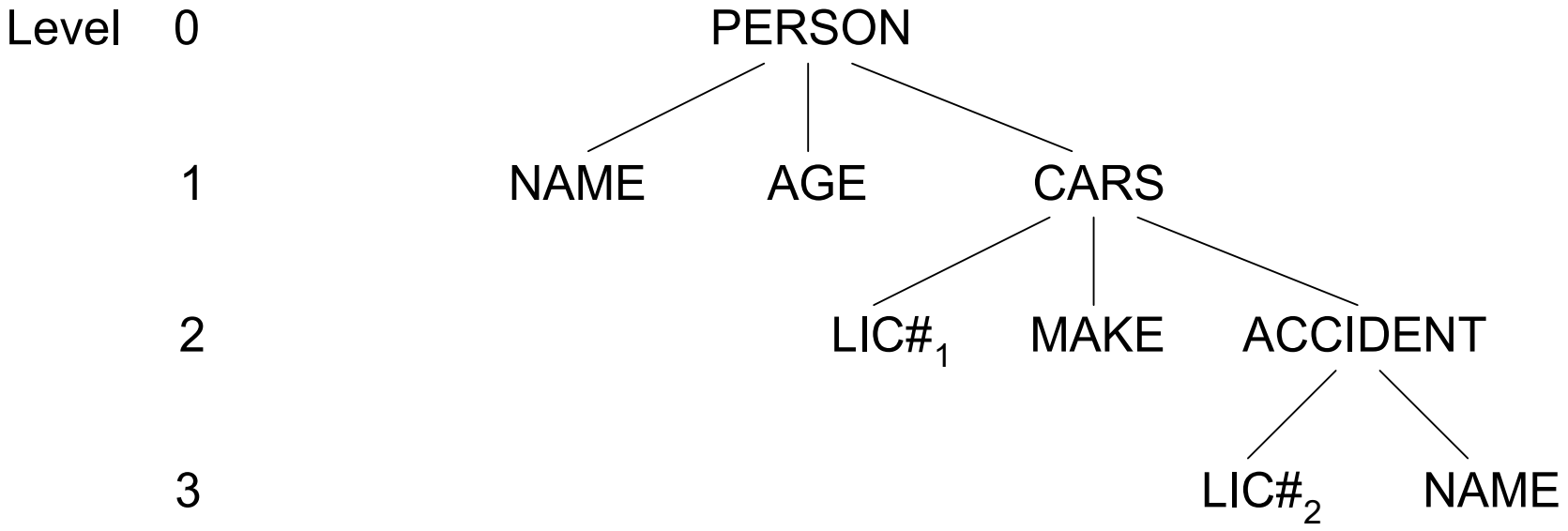
Data conversion: Customerized Program Approach

Một cách tiếp cận chung cho việc chuyển đổi dữ liệu là xây dựng các chương trình tùy biến để chuyển dữ liệu từ một môi trường này sang một môi trường khác. Tuy nhiên, cách tiếp cận này là cực kỳ tốn kém bởi vì nó đòi hỏi một chương trình riêng biệt được viết cho từng M tập nguồn và N tập đích. Hơn nữa, các chương trình này chỉ được sử dụng một lần. Kết quả là, việc hoàn toàn phụ thuộc vào chương trình tùy biến cho dữ liệu chuyển đổi là không thể quản lý, quá tốn kém và mất thời gian

Interpretive Transformer approach of data conversion



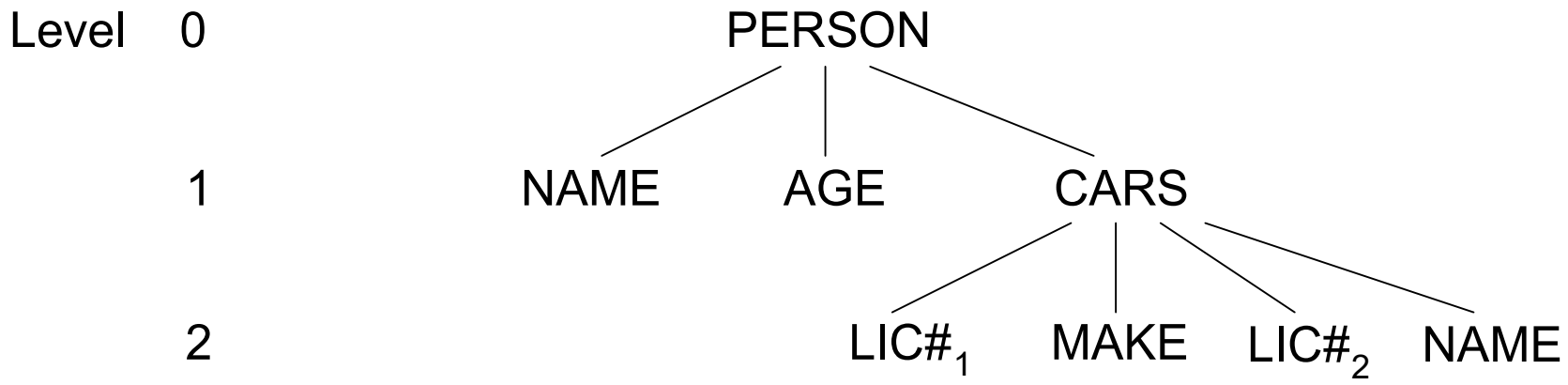
For instance, the following Cobol structure



can be expressed in using these SDDL statements:

Data Structure	PERSON <NAME, AGE>
Instance	CARS <LIC#1, MAKE>
N-tuples	ACCIDENTS <LIC#2, NAME>
Relationship	PERSON-CAR <NAME, LIC#1>
N-tuples	CAR-ACCIDENT <LIC#1, LIC#2>

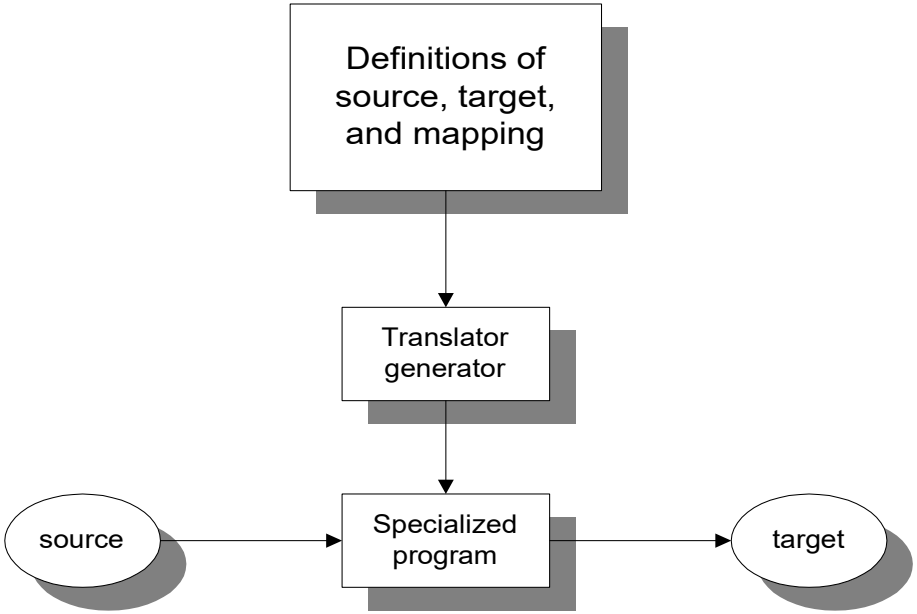
To translate the above three levels to the following two levels data structure:



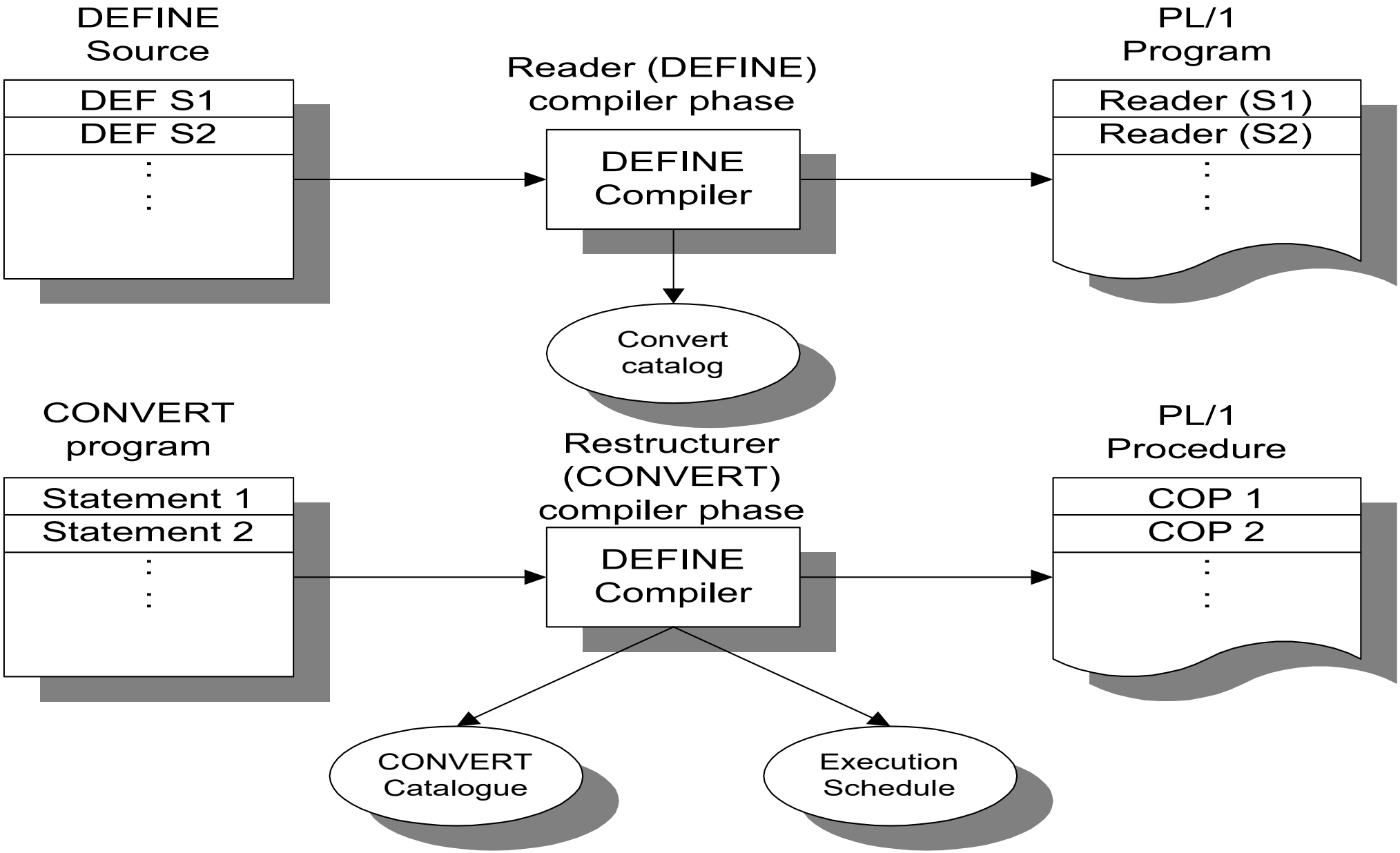
the TDL statements are

```
FORM NAME FROM NAME
FORM LIC#1 FROM LIC#1
:
FORM PERSON IF PERSON
FORM CARS IF CAR AND ACCIDENT
```

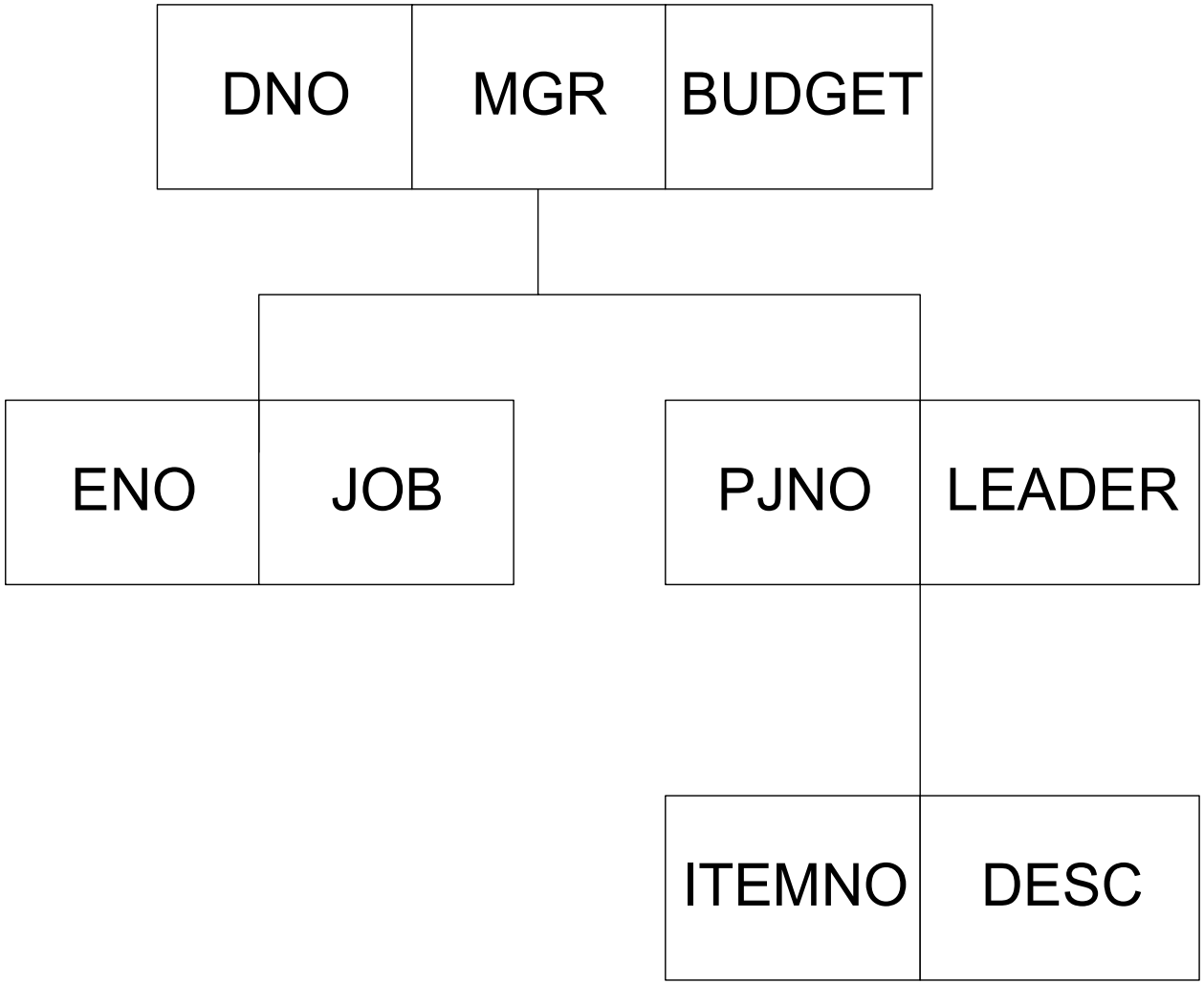
Translator Generator Approach of data conversion



DEFINE and CONVERT compile phase



As an example, consider the following hierarchical database:



Its DEFINE statements can be described in the following where for each DEFINE statement, a code is generated to allocate a new subtree in the internal buffer.

```
GROUP DEPT:
    OCCURS FROM 1 TIMES;
    FOLLOWED BY EOF;
    PRECEDED BY HEX ‘01’;
    :
    END EMP;
    GROUP PROJ:
        OCCURS FROM 0 TIMES;
        PRECEDED BY HEX ‘03’;
    :
    END PROJ;
END DEPT;
```

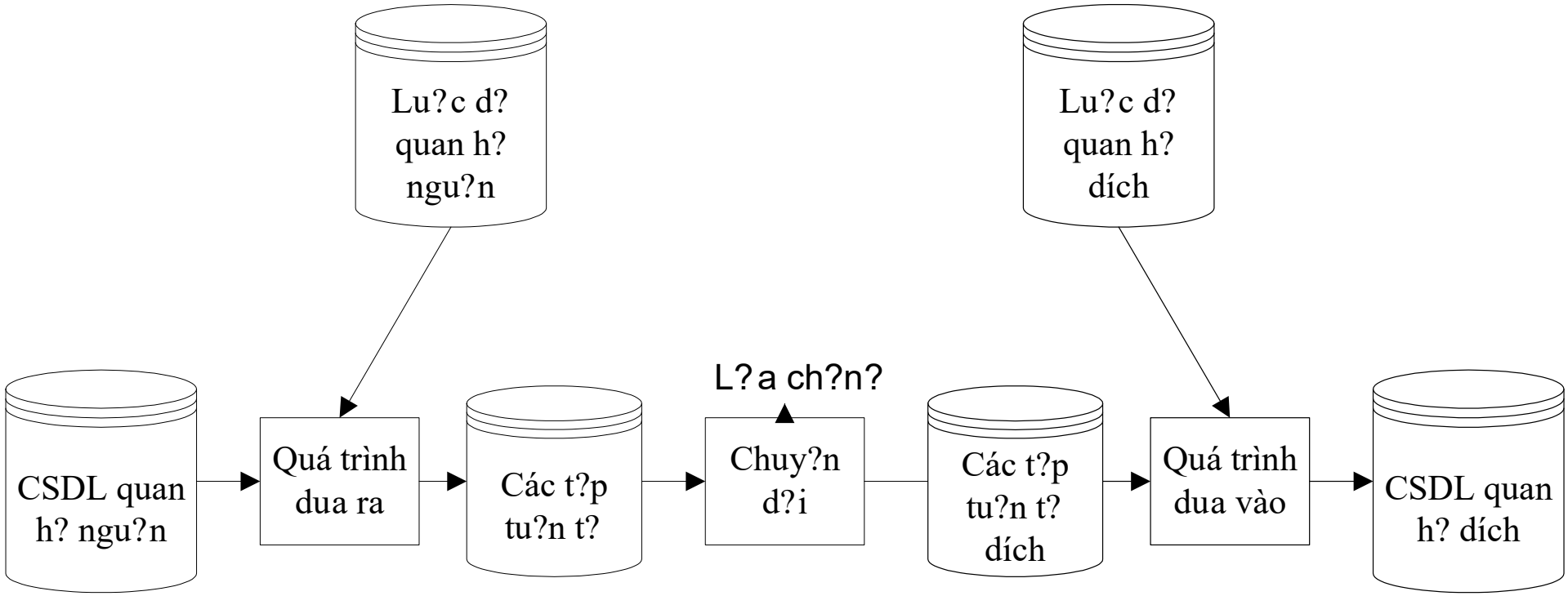
For each user-written CONVERT statement, we can produce a customized program. Take the DEPT FORM from the above DEFINE statement:

T1 = SELECT (FROM DEPT WHERE BUDGET GT ‘100’);

will produce the following program:

```
/* PROCESS LOOP FOR T1 */
DO WHILE (not end of file);
   CALL GET (DEPT);
   IF BUDGET > ‘100’
        THEN CALL BUFFER_SWAP (T1, DEPT);
END
```

Data conversion: Logical Level Translation Approach



System flow diagram for data conversion from source relational to target relational

Business requirements

- Một công ty có 2 khu vực hoạt động A và B
- Mỗi khu vực có văn phòng riêng
- Mỗi văn phòng duyệt một số chuyến đi trong một năm
- Mỗi nhân viên đi nhiều chuyến công tác trong 1 năm
- Trong mỗi chuyến công tác, mỗi nhân viên cần thuê xe cho việc di chuyển
- Mỗi chiếc xe được thuê đều có thể chở được nhiều nhân viên
- Một nhân viên có thể là một người quản lý hoặc một kỹ sư

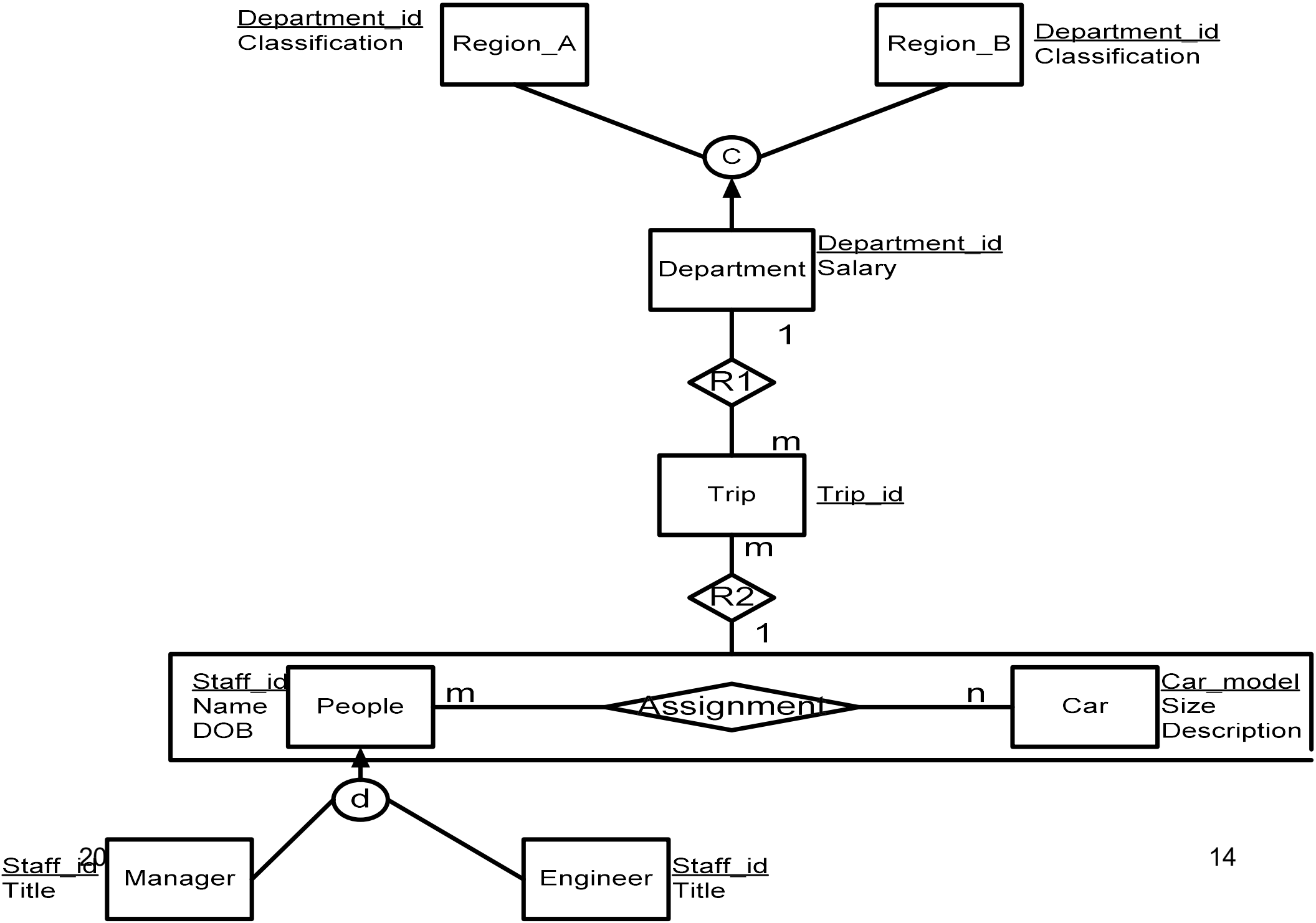
Data Requirements

Relation Department(*Department_id, Salary)
Relation Region_A (Department_id, Classification)
Relation Region_B (Department_id, Classification)
Relation Trip (Trip_id, *Car_model, *Staff_id, *Department_id)
Relation People (Staff_id, Name, DOB)
Relation Car (Car_model, Size, Description)
Relation Assignment(*Car_model, *Staff_id)
Relation Engineer (*Staff_id, Title)
Relation Manager (*Staff_id, Title)

ID: Department.Department_id $\subseteq$ (Region_A.Department_id $\cup$ Region_B.Department_id)
ID: Trip.Department_id $\subseteq$ Department.Department_id
ID: Trip.Car_model $\subseteq$ Assignment.Car_model
ID: Trip.Staff_id $\subseteq$ Assignment.Staff_id
ID: Assignment.Car_model $\subseteq$ Car.Car_model
ID: Assignment.Staff_id $\subseteq$ People.Staff_id
ID: Engineer.Staff_id $\subseteq$ People.Staff_id
ID: Manager.Staff_id $\subseteq$ People.Staff_id

Where ID = Inclusion Dependence, Underlined are primary keys and “*” prefixed are foreign keys.

Extended Entity Relationship Model for database Trip



Schema for target relational database Trip

Create Table Car
(CAR_MODEL character (10),
SIZE character (10),
DESCRIPT character (20),
STAFF_ID character (5),
primary key (CAR_MODEL))

Create Table Depart
(DEP_ID character (5),
SALARY numeric (8),
primary key (DEP_ID))

Create Table People
(STAFF_ID character (4),
NAME character (20),
DOB datetime,
primary key (STAFF_ID))

Create Table Reg_A
(DEP_ID character (5),
DESCRIP character (20),
primary key (DEP_ID))

Create Table Reg_B
(DEP_ID character (5),
DESCRIPT character (20),
primary key (DEP_ID))

Schema for target relational database Trip (continue)

Create Table trip
(TRIP_ID character (5),
primary key (TRIP_ID))

Create Table Engineer
(TITLE character (20),
STAFF_ID character (4),
Foreign Key (STAFF_ID) REFERENCES People(STAFF_ID),
Primary key (STAFF_ID))

Create Table Manager
(TITLE character (20),
STAFF_ID character (4),
Foreign Key (STAFF_ID) REFERENCES People(STAFF_ID),
Primary key (STAFF_ID))

Create Table Assign
(CAR_MODEL character (10),
Foreign Key (CAR_MODEL) REFERENCES Car(CAR_MODEL),
STAFF_ID character (4),
Foreign Key (STAFF_ID) REFERENCES People(STAFF_ID),
primary key (CAR_MODEL,STAFF_ID))

ALTER TABLE trip ADD CAR_MODEL character (10) null

ALTER TABLE trip ADD STAFF_ID character (4) null

ALTER TABLE trip ADD Foreign Key (CAR_MODEL,STAFF_ID) REFERENCES Assign(CAR_MODEL,STAFF_ID)

1.4. Data Integration
Rule 1: Merge by Union

Relation Ra

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rb

$\underline{A_1}$	A_3
a_{13}	a_{31}
a_{14}	a_{32}

==>

Relation Rx

$\underline{A_1}$	A_2	A_3
a_{11}	a_{21}	null
a_{12}	a_{22}	null
a_{13}	null	a_{31}
a_{14}	null	a_{32}

Rule 2:
Merge classes by generalization

Relation Ra

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rb

$\underline{A_1}$	A_3
a_{13}	a_{31}
a_{14}	a_{32}

==>

Relation Rx1

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rx2

$\underline{A_1}$	A_3
a_{13}	a_{31}
a_{14}	a_{32}

Relation Rx

$\underline{A_1}$	A_2	A_3
a_{11}	a_{21}	null
a_{12}	a_{22}	null
a_{13}	null	a_{31}
a_{14}	null	a_{32}

Rule 3:
Merge classes by inheritance

Relation Rb

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rxb

$\underline{A_1}$	A_2	A_3
a_{11}	a_{21}	a_{31}
a_{12}	a_{22}	null

==>

Relation Ra

$\underline{A_1}$	A_3
a_{11}	a_{31}

Relation Rxa

$\underline{A_1}$	A_3
a_{11}	a_{31}

Rule 4:
Merge classes by aggregation

Relation Ra

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rx

$\underline{A_1}$	$\underline{A_3}$	A_5
a_{11}	a_{31}	a_{51}
a_{12}	a_{32}	a_{51}

Relation Rb

$\underline{A_3}$	A_4
a_{31}	a_{41}
a_{32}	a_{42}

Relation Ry

$\underline{A_5}$	A_6
a_{51}	a_{61}
a_{52}	a_{62}

==>

Relation Rx1

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rx'

$\underline{A_1}$	$\underline{A_3}$	$*A_5$
a_{11}	a_{31}	a_{51}
a_{12}	a_{32}	a_{51}

Relation Rx2

$\underline{A_3}$	A_4
a_{31}	a_{41}
a_{32}	a_{42}

Relation Ry

$\underline{A_5}$	A_6
a_{51}	a_{61}
a_{52}	a_{62}

Ry liên kết với thực thể ghép Rx (Ra n-n Rb)

Lý do phải define khóa ngoại A5 rõ ràng hơn (quan hệ n-1 với Ry) so với ban đầu vì ban đầu các Relation kia không phải chung 1 cơ sở dữ liệu, khi cho vào chung 1 cơ sở dữ liệu mà nó có liên kết với Ry (có primary key là A5) thì A5 phải thành khóa ngoại.

Rule 5:
Merge classes by categorization

Relation Ra

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rb

$\underline{A_1}$	A_3
a_{13}	a_{31}
a_{14}	a_{32}

==>

Relation Rx

$\underline{A_1}$	$\underline{A_4}$
a_{11}	a_{41}
a_{12}	a_{42}
a_{13}	a_{43}
a_{14}	a_{44}

A4 để phân loại (categorize)

Relation Ra

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rb

$\underline{A_1}$	A_3
a_{13}	a_{31}
a_{14}	a_{32}

Rule 6:
Merge classes by implied relationship

Relation Ra

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rb

$\underline{A_3}$	A_1
a_{31}	a_{11}
a_{32}	a_{12}

==>

Relation Xa

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Xb

$\underline{A_3}$	$*A_1$
a_{31}	a_{11}
a_{32}	a_{12}

Rb có A1 không phải khóa mà Ra có A1 là khóa
=> Quan hệ 1-n giữa Xa và Xb khi đổ chung vào 1 CSDL

Rule 7:
Merge relationship by subtype (is–a)

Relation Ra

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rb

$\underline{A_3}$	$*A_1$
a_{31}	a_{11}
a_{32}	a_{12}

Relation Ra'

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Rb'

$\underline{A_3}$	A_1
a_{31}	a_{11}
a_{33}	null

==>

Relation Xa

$\underline{A_1}$	A_2
a_{11}	a_{21}
a_{12}	a_{22}

Relation Xc

$*\underline{A_3}$	$*A_1$
a_{31}	a_{11}
a_{32}	a_{12}

Relation Xb

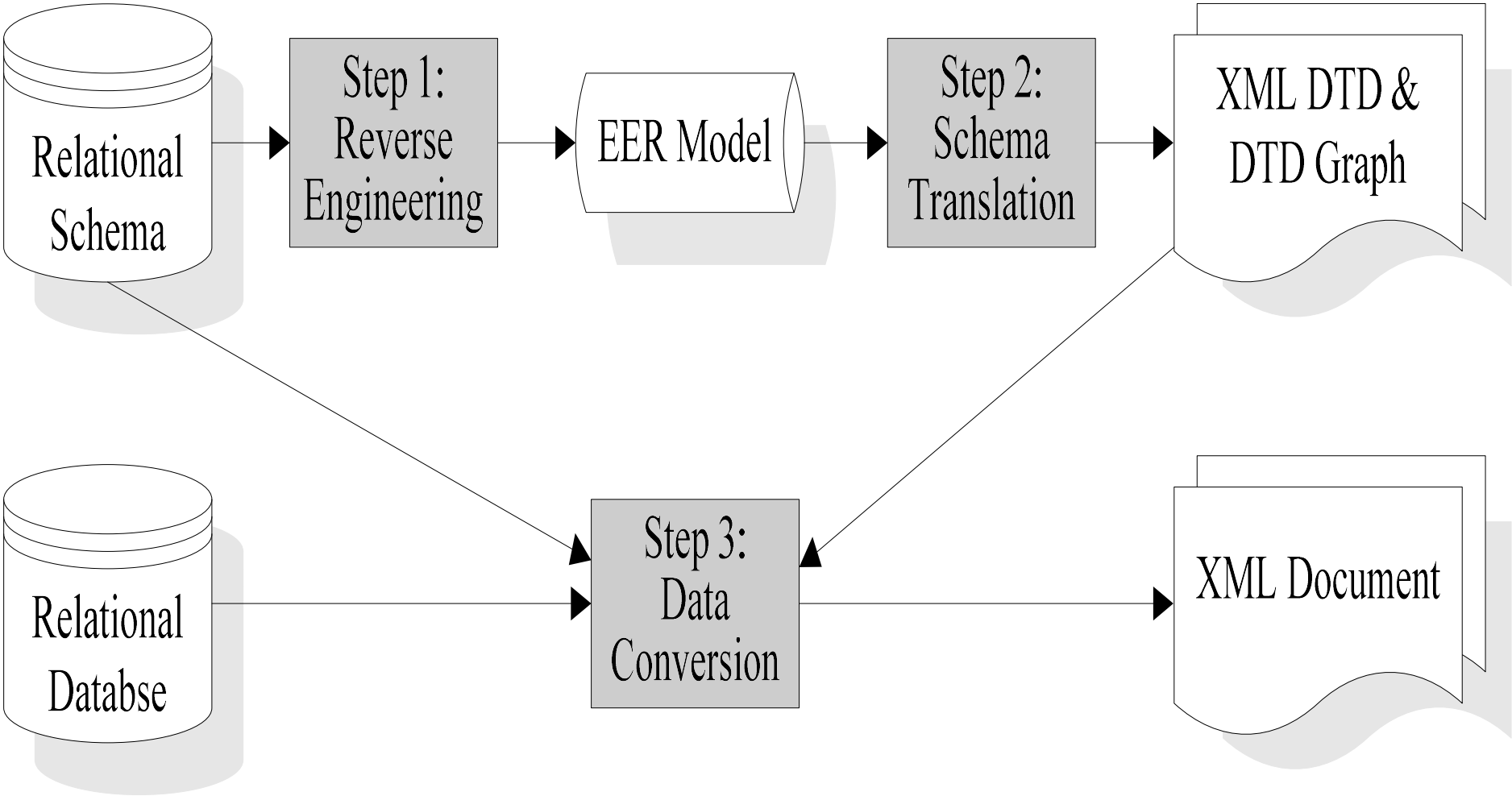
$\underline{A_3}$	A_1
a_{31}	a_{11}
a_{32}	a_{12}
a_{33}	null

(Ra, Rb); (Ra', Rb') ở 2 nguồn khác nhau

Ra -> Xa; Rb -> Xc

Bản chất X3 là hợp của Rb và Rb', là cha của Xc (mang A3) và Xa (mang A1)

Data conversion from relational into XML (có thể thi)



Architecture of Re-engineering Relational Database into XML Document

Methodology of converting RDB into XML

Là kết quả của việc phiên dịch lược đồ, chúng ta dịch một mô hình thực thể liên kết sang một khung nhìn khác của lược đồ XML là DTD dựa trên phần tử gốc được lựa chọn của chúng. Cho mỗi lược đồ XML được dịch, chúng ta có thể đọc các quan hệ nguồn tương ứng một cách tuần tự bằng các câu lệnh nhúng SQL bắt đầu từ một quan hệ cha. Các bộ sau đó có thể được tải vào văn bản XML dựa vào cấu trúc DTD của XML. Chúng ta sau đó có thể đọc các bộ quan hệ con và tải chúng vào văn bản XML

Step 1: Reverse Engineering Relational Schema into an EER Model

By use of classification tables to define the relationship between keys and attributes in all relations, we can recover their data semantics in the form of an EER model.

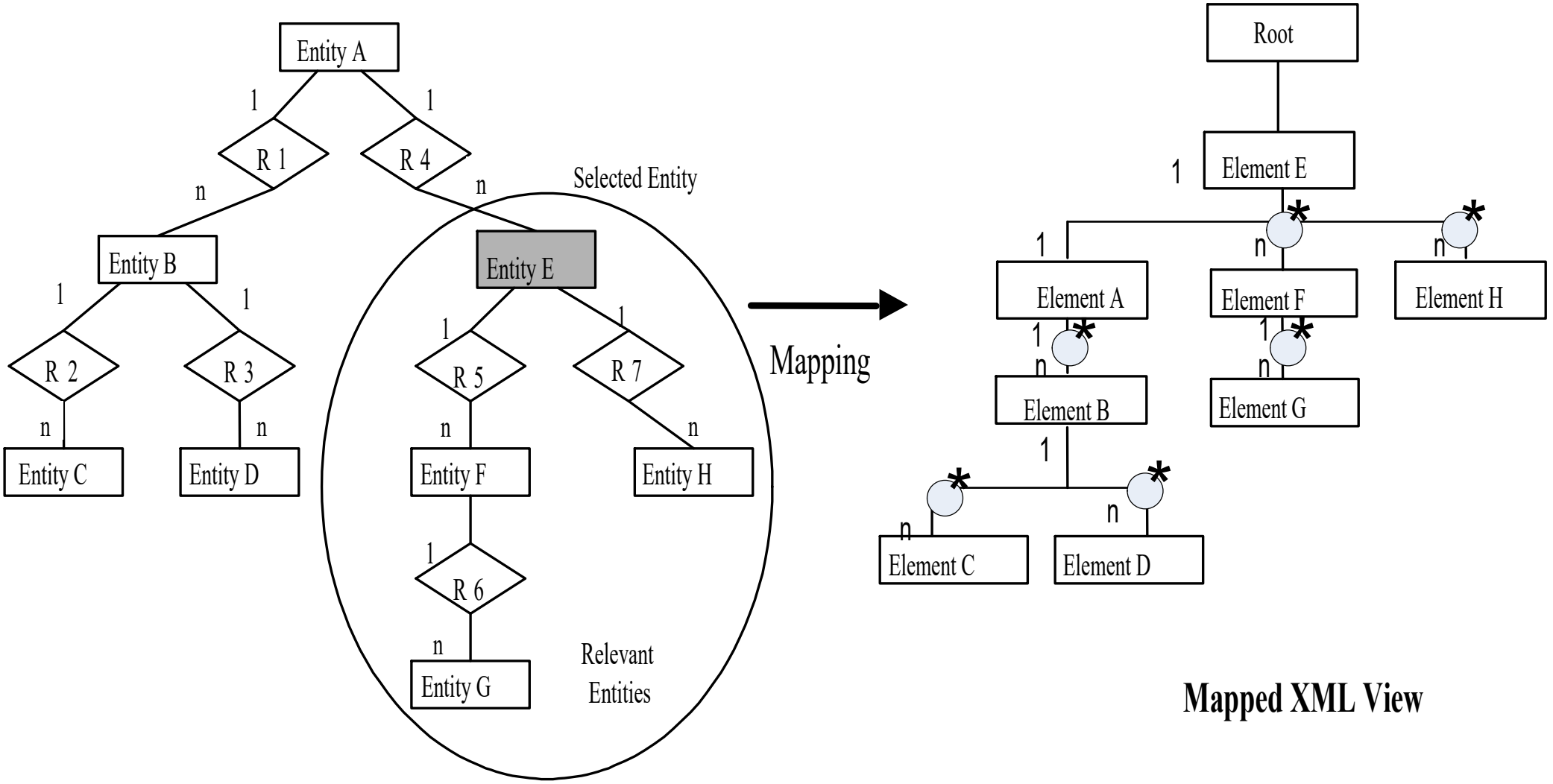
Step 2: Data Conversion from Relational into XML document

We can map the data semantics in the EER model into DTD-graph according to their data dependencies constraints. These constraints can then be transformed into DTD as XML schema.

Step 2.1 Defining a Root Element

To select a root element, we must put its relevant information into an XML schema. Relevance concerns with the entities that are related to a selected entity by the user. The relevant classes include the selected entity and all its relevant entities that are navigable.

In an EER mode, we can navigate from entity to another entity in correspondence to XML hierarchical containment tree model.



EER Model

Nhấc E lên làm Root nhưng vì có quan hệ n-1 với A nên phải sinh ra Root giả để không mất ngữ nghĩa (n-1 thành nhiều cái 1-n)

Khi đó, bên DTD E và A quan hệ 1-1

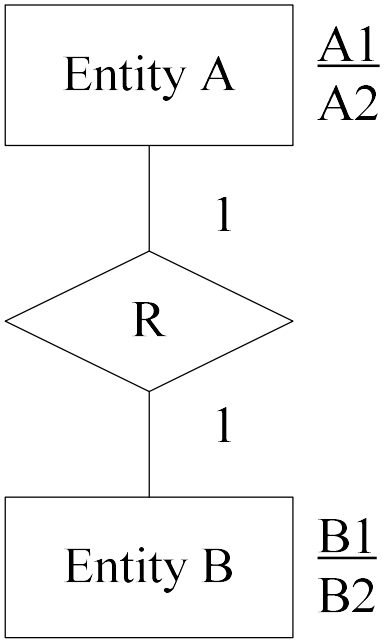
Step 2.2 Mapping Cardinality from RDB to XML

Trong DTD chúng ta dịch:

- quan hệ 1-1 vào phần tử cha và phần tử con
- quan hệ 1-n vào phần tử cha và phần tử con có xuất hiện nhiều lần
- Trong quan hệ n-n, chúng được ánh xạ vào DTD của một cấu trúc phân cấp với một định danh ID và tham chiếu định danh IDREF.

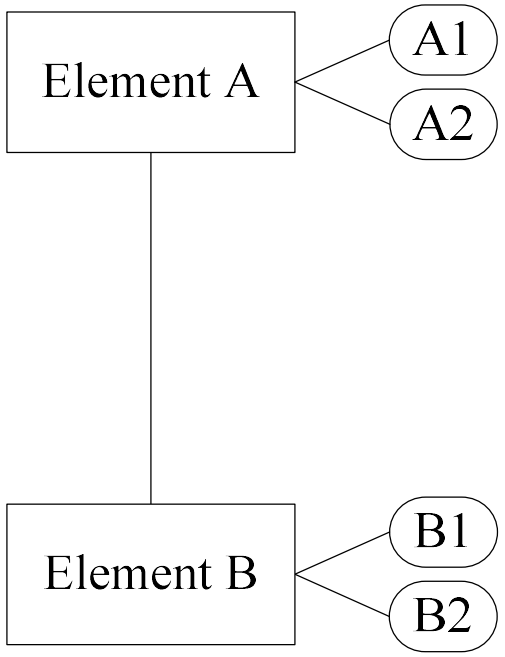
One-to-one cardinality

EER Model



Schema Translation

DTD Graph



Relational Schema

Relation A(<u>A1</u>, A2)
Relation B(<u>B1</u>, B2, *A1)

DTD

```
<!ELEMENT  A(B)>
<!ATTLIST A A1 CDATA #REQUIRED>
<!ATTLIST A A2 CDATA #REQUIRED>
<!ELEMENT  B EMPTY>
<!ATTLIST B B1 CDATA #REQUIRED>
<!ATTLIST B B2 CDATA #REQUIRED>
```

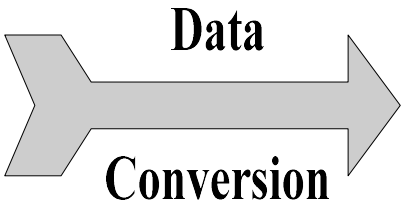
One-to-one cardinality

Relation A

A1	A2
a11	a21
a12	a22

Relation B

B1	B2	*A1
b11	b21	a11
b12	b22	a12

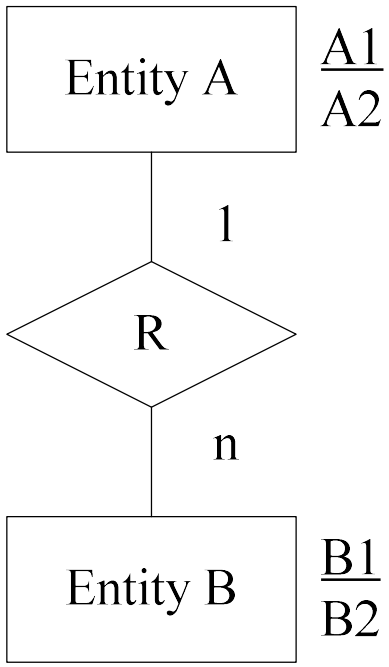


XML Document

```
<A A1="a11" A2="a21">
   <B B1="b11" B2="b21"></B>
</A>

<A A1="a12" A2="a22">
   <B B1="b12" B2="b22"></B>
</A>
```

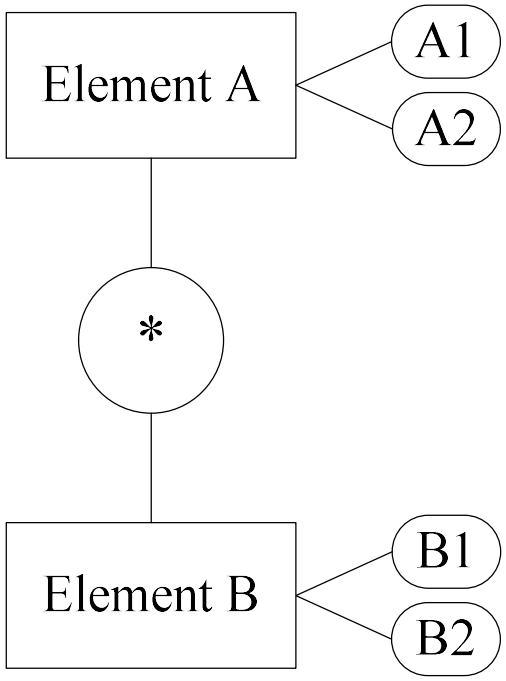
One-to-many cardinality

EER Model



Schema Translation

DTD Graph



Relational Schema

Relation A(<u>A1</u>, A2)
Relation B(<u>B1</u>, B2, *A1)

DTD

```
<!ELEMENT  A(B)*>
<!ATTLIST A A1 CDATA #REQUIRED>
<!ATTLIST A A2 CDATA #REQUIRED>
<!ELEMENT  B EMPTY>
<!ATTLIST B B1 CDATA #REQUIRED>
<!ATTLIST B B2 CDATA #REQUIRED>
```

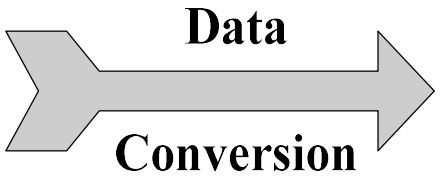
One-to-many cardinality

Relation A

A1	A2
a11	a21
a12	a22

Relation B

B1	B2	*A1
b11	b21	a11
b12	b22	a12
b13	b23	a12

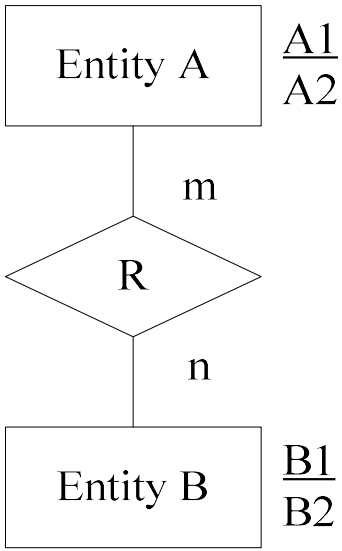


XML Document

```
<A A1="a11" A2="a21">
    <B B1="b11" B2="b21"></B>
</A>

<A A1="a12" A2="a22">
    <B B1="b12" B2="b22"></B>
    <B B1="b13" B2="b23"></B>
</A>
```

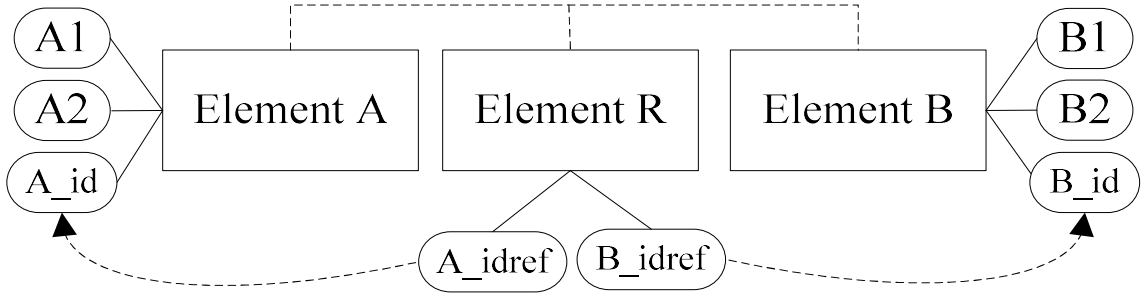
Many-to-many cardinality

EER Model



Schema Translation

DTD Graph



Relational Schema

Relation A(<u>A1</u>, A2)
Relation B(<u>B1</u>, B2)
Relation R(*<u>A1</u>, *<u>B1</u>)

A_id, B_id là sinh mới (có sự liên kết với R). Chúng thuộc kiểu ID theo XML (khóa giả).

DTD

```
<!ELEMENT A EMPTY>
<!ATTLIST A A1 CDATA #REQUIRED>
<!ATTLIST A A2 CDATA #REQUIRED>
<!ATTLIST A A_id ID #REQUIRED>
<!ELEMENT R EMPTY>
<!ATTLIST R A_idref IDREF #REQUIRED>
<!ATTLIST R B_idref IDREF #REQUIRED>
<!ELEMENT B EMPTY>
<!ATTLIST B B1 CDATA #REQUIRED>
<!ATTLIST B B2 CDATA #REQUIRED>
<!ATTLIST B B_id ID #REQUIRED>
```

Many-to-many cardinality

Relation A

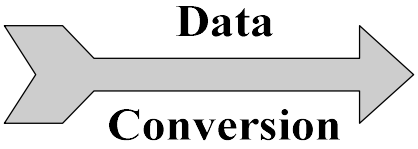
A1	A2
a11	a21
a12	a22

Relation B

B1	B2
b11	b21
b12	b22

Relation R

*A1	*B1
a11	b11
a12	b12
a11	b12



XML Document

```
<A A1="a11" A2="a21" A_id="1"></A>
<B B1="b11" B2="b21" B_id="2"></B>
<R A_idref="1" B_idref="2"></R>

<A A1="a12" A2="a22" A_id="3"></A>
<B B1="b12" B2="b22" B_id="4"></B>
<R A_id="3" B_idref="4"></R>

  <R A_id=”1" B_idref=”4"></R>
```

Case Study

Consider a case study of a Hospital Database System. In this system, a patient can have many record folders. Each record folder can contain many different medical records of the patient. A country has many patients. Once a record folder is borrowed, a loan history is created to record the details about it.

Hospital Relational Schema

Relation Patient (HK_ID, Patient_Name)

Relation Record_Folder (Folder_No, Location, *HKID)

Relation Medical_Record (Medical_Rec_No, Create_Date, Sub_Type, *Folder_No)

Relation Borrower (*Borrower_No, Borrower_Name)

Relation Borrow (*Folder_No , *Borrower_No)

Where underlined are primary keys, prefixed with “*” are foreign keys

Giải thích

- Record_Folder chứa các Record (hồ sơ) bệnh nhân, gồm mã Folder, nơi Folder cất và mã bệnh nhân
- Medical_Record chứa hồ sơ bệnh nhân, gồm mã hồ sơ, ngày tạo, kiểu hồ sơ, nằm trong folder nào.
- Borrower tức người mượn (Bác sĩ/ y tá/…)
- Người mượn mượn (Borrow) cả folder.

Relational Data

Patient table

E3766849	Smith

Record_Folder table

F_21	Hong Kong	E3766849
F_24	New Territories	E3766849

Borrower Table

Borrower_no	Borrower_name
B1	Johnson
B11	Choy
B21	Fung
B22	Lok

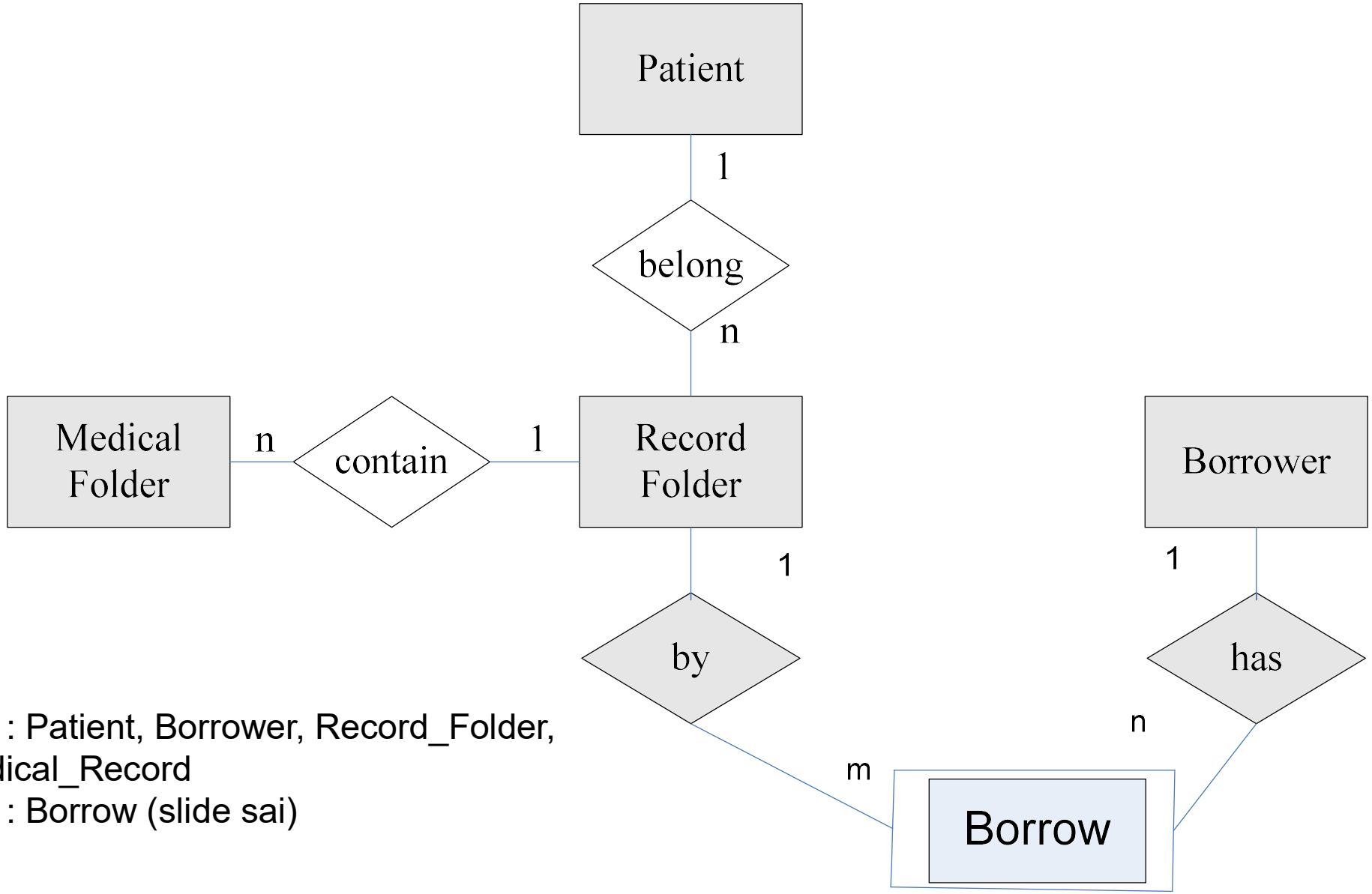
Medical_Record Table

Medical_Rec_no	Create_Date	Sub_type	Folder_no
M_311999	Jan-01-1999	W	F_21
M_322000	Nov-12-1998	W	F_21
M_352001	Jan-15-2001	A	F_21
M_362001	Feb-01-2001	A	F_21
M_333333	Mar-03-2001	A	F_24

Borrow table

F_21	B1
F_21	B11
F_21	B21
F_21	B22
F_24	B22

Step 1 Reverse engineer relational database into an EER model

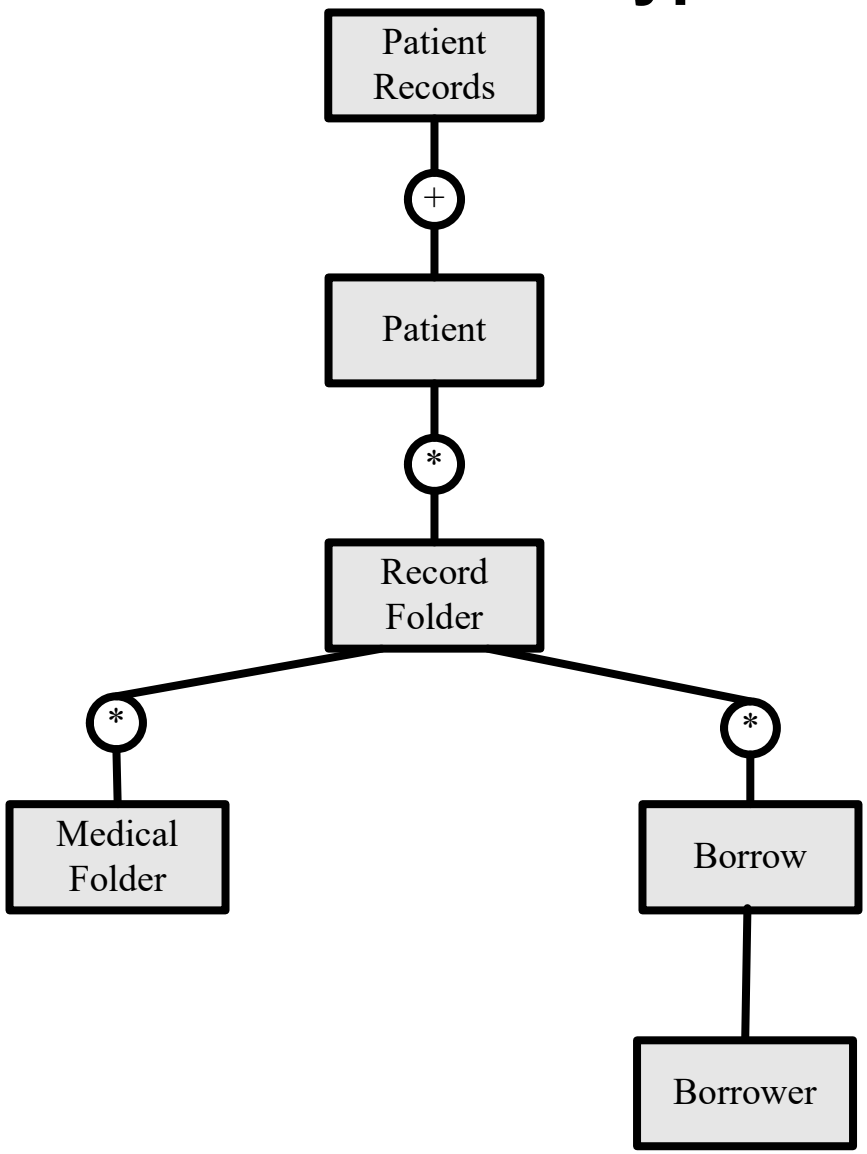


PR1: Patient, Borrower, Record_Folder, Medical_Record
SR1: Borrow (slide sai)

Step 2 Translate EER model into DTD Graph and DTD

In this case study, suppose we concern the patient medical records, so the entity Patient is selected. Then we define a meaningful name for the root element, called Patient_Records. We start from the entity Patient in the EER model and then find the relevant entities for it. The relevant entities include the related entities that are navigable from the parent entity.

Translated Document Type Definition Graph



+ tức xuất hiện 1 -> nhiều lần
* tức xuất hiện 0 -> nhiều lần

Translated Document Type Definition

```
<!ELEMENT Patient_Records (Patient+)>

<!ELEMENT Patient (Record_Folder*)>
<!ATTLIST Patient       HKID                CDATA           #REQUIRED>
<!ATTLIST Patient       Patient_Name                CDATA           #REQUIRED>

<!ELEMENT Record_Folder (Borrow*, Medical_Record*)>
<!ATTLIST   Record_Folder Folder_No                 CDATA           #REQUIRED>
<!ATTLIST   Record_Folder Location                  CDATA           #REQUIRED>

<!ELEMENT Borrow (Borrower)>
<!ATTLIST   Borrow Borrower_No          CDATA           #REQUIRED>

<!ELEMENT Medical_Record EMPTY>
<!ATTLIST Medical_Record Medical_Rec_No             CDATA           #REQUIRED>
<!ATTLIST Medical_Record Create_Date                CDATA           #REQUIRED>
<!ATTLIST Medical_Record Sub_Type                   CDATA           #REQUIRED>

<!ELEMENT Borrower EMPTY>
<!ATTLIST Borrower Borrower_name                    CDATA           #REQUIRED>
```

Transformed XML document

```
Patient_Records>
<Patient Country_No="C0001" HKID="E3766849" Patient_Name="Smith">
    <Record_Folder Folder_No="F_21" Location="Hong Kong">
        <Borrow Borrower_No="B1">
            <Borrower Borrower_name="Johnson" />
        </Borrow>
        <Borrow Borrower_No="B11">
            <Borrower Borrower_name="Choy" />
        </Borrow>
        <Borrow Borrower_No="B21">
            <Borrower Borrower_name="Fung" />
        </Borrow>
        <Borrow Borrower_No="B22">
            <Borrower Borrower_name="Lok" />
        </Borrow>
        <Medical_Record Medical_Rec_No="M_311999" Create_Date="Jan-1-1999",  Sub_Type="W"></Medical_Record>
        <Medical_Record Medical_Rec_No="M_322000" Create_Date="Nov-12-1998", Sub_Type="W"></Medical_Record>
        <Medical_Record Medical_Rec_No="M_352001" Create_Date="Jan-15-2001", Sub_Type="A"></Medical_Record>
        <Medical_Record Medical_Rec_No="M_362001" Create_Date="Feb-01-2001", Sub_Type="A"></Medical_Record>
    </Record_Folder>
    <Record_Folder Folder_No="F_24" Location="New Territories">
        <Borrow Borrower_No="B22">
            <Borrower Borrower_name="Lok" />
        </Borrow>
        <Medical_Record Medical_Rec_No="M_333333" Create_Date="Mar-03-01", Sub_Type="A"></Medical_Record>
    </Record_Folder>
</Patient>
```

Reading Assignment

Chapter 4 Data Conversion of “Information Systems Reengineering and Integration” by Joseph Fong, Springer Verlag, pp.160-198.

Lecture review question 6

How do you compare the pros and cons of using “Logical Level Translation Approach” with “Customized Program Approach” in data conversion?

CS5483 Tutorial Question 6

Convert the following relational database into an XML document:

Relation Car_rental

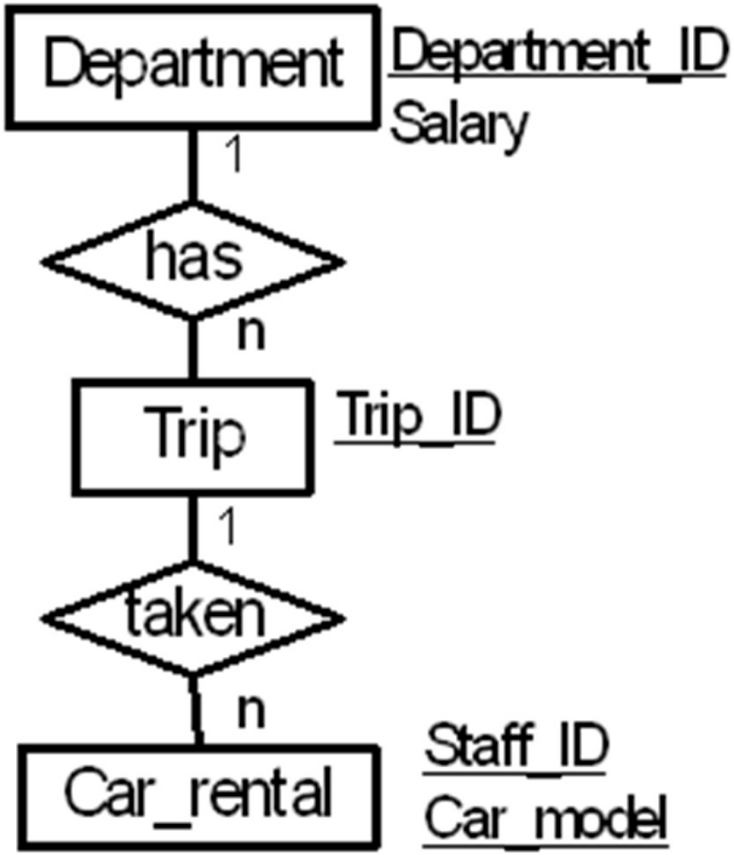
Car_model	Staff_ID	*Trip_ID
MZ-18	A002	T0001
MZ-18	B001	T0002
R-023	B004	T0001
R-023	C001	T0004
SA-38	A001	T0003
SA-38	A002	T0001

Relation Trip

Trip_ID	*Department_ID
T0001	AA001
T0002	AA001
T0003	AB001
T0004	BA001

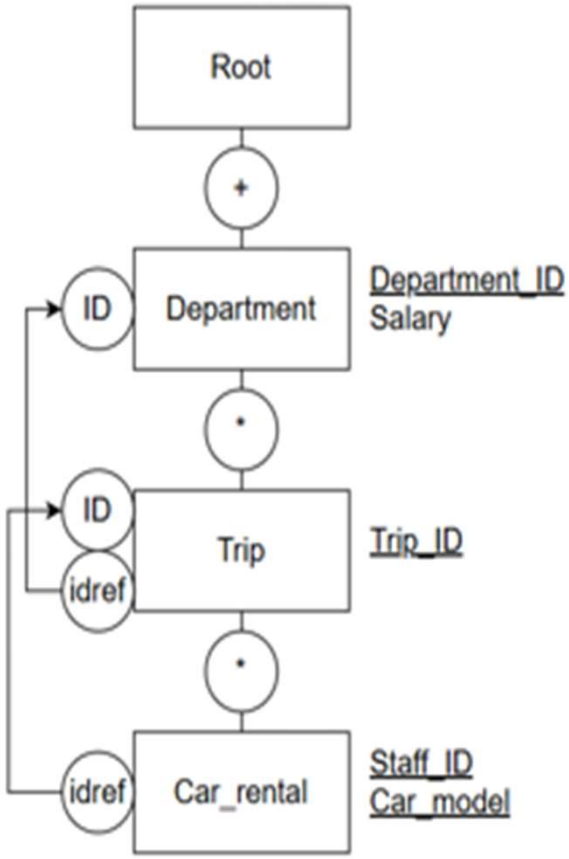
Relation Department

Department_ID	Salary
AA001	35670
AB001	30010
BA001	22500



An ER model for car rental

DTD Graph (thực ra cũng không cần Root vì Department không có ngữ nghĩa nhiều):



Root nếu có thì quan hệ với Department nên vẽ dấu *
Bỏ cái id, idref không cần thiết lắm

Translated Document Type Definition

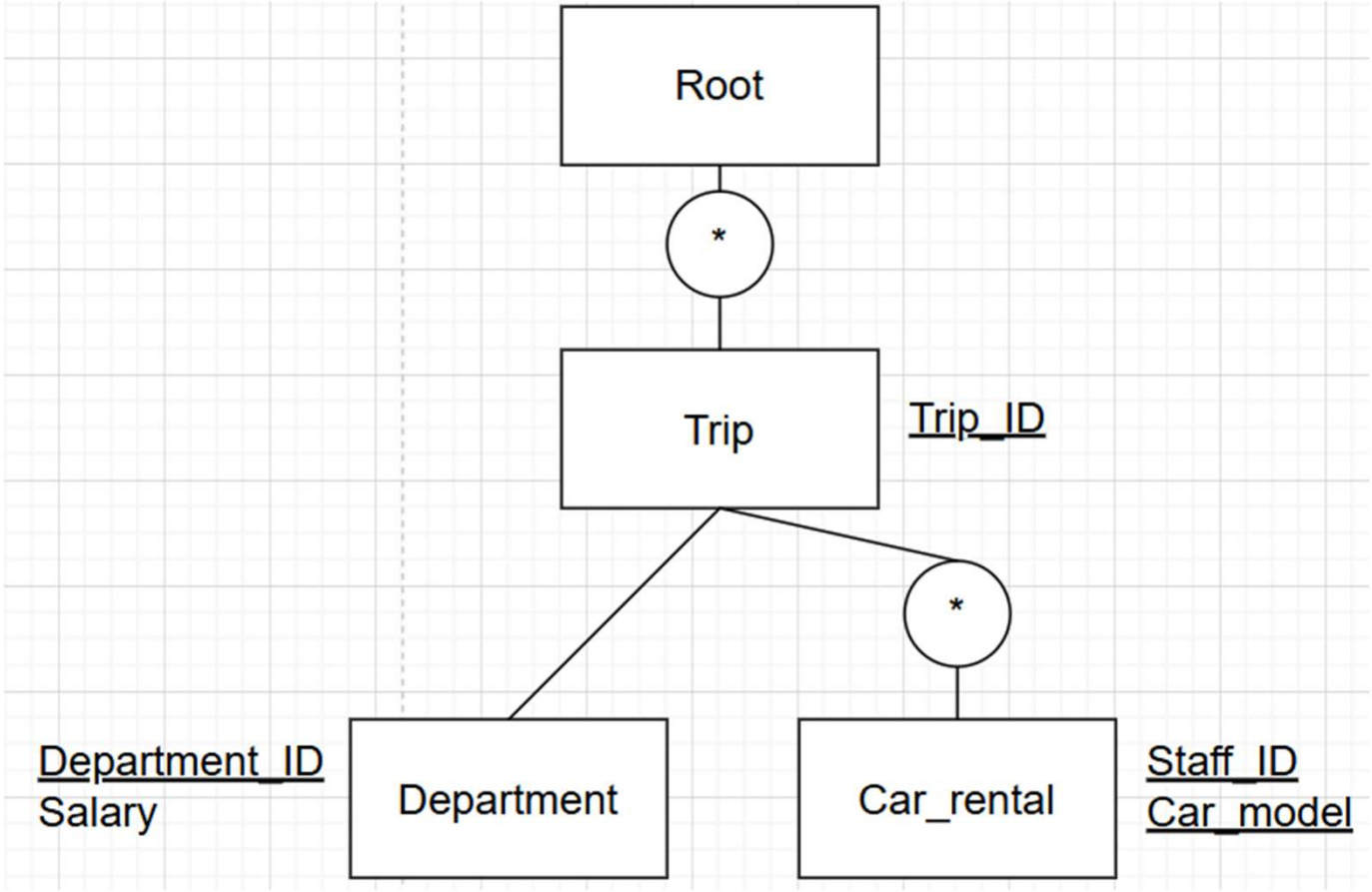
```
<!ELEMENT Department (Trip*)>
<!ATTLIST Department
    Department_ID CDATA #REQUIRED
    Salary CDATA #REQUIRED>

<!ELEMENT Trip (Car_rental*)>
<!ATTLIST Trip
    Trip_ID CDATA #REQUIRED>

<!ELEMENT Car_rental EMPTY>
<!ATTLIST Car_rental
    Staff_ID CDATA #REQUIRED
    Car_model CDATA #REQUIRED>
```

(Cách này là dùng Department bên trên, có thể để Trip hoặc Car-Rental đều được)

Transformed XML document

```
<Department Department_ID="AA001" Salary="35670">
    <Trip Trip_ID="T001">
        <Car_rental Car_model="MZ-18" Staff_ID="A002"/>
        <Car_rental Car_model="R-023" Staff_ID="C001"/>
        <Car_rental Car_model="SA-38" Staff_ID="A002"/>
    </Trip>
    <Trip Trip_ID="T002">
        <Car_rental Car_model="MZ-18" Staff_ID="B001"/>
    </Trip>
</Department>
<Department Department_ID="AB001" Salary="30010">
    <Trip Trip_ID="T003">
        <Car_rental Car_model="SA-38" Staff_ID="A002"/>
    </Trip>
</Department>
<Department Department_ID="BA001" Salary="22500">
    <Trip Trip_ID="T004">
        <Car_rental Car_model="R-023" Staff_ID="C001"/>
    </Trip>
</Department>
```

Cách làm dùng Trip:



Translated Document Type Definition

```
<!ELEMENT Trip (Car_rental*, Department)>
<!ATTLIST Department
   Trip_ID CDATA #REQUIRED>

<!ELEMENT Department EMPTY>
<!ATTLIST Trip
   Department_ID CDATA #REQUIRED
   Salary CDATA #REQUIRED >

<!ELEMENT Car_rental EMPTY>
<!ATTLIST Car_rental
   Staff_ID CDATA #REQUIRED
   Car_model CDATA #REQUIRED>
```

Chú ý với cách dùng Trip, trong 1 thẻ <Trip> không thể có nhiều thẻ <Car-Rental> cùng cấp với 1 thẻ <Department> vì <Trip> cùng cấp với <Department>, việc để nhiều thẻ <Car-Rental> cùng cấp <Department> là không hợp lý.

Có 2 cách xử lý tình huống này:

1. Gộp nhiều thẻ <Car-Rental> trong 1 thẻ <Car-Rentals> rồi mới để trong <Trip>
2. Tách mỗi thẻ <Car-Rental> tương ứng 1 <Trip>

Transformed XML document

```
<Trip Trip_ID="T0001">
    <Department Department_ID="AA001" Salary="35670"></Department>
    <Car_rentals>
        <Car_rental Car_model="MZ-18" Staff_ID="A002"></Car_rental>
        <Car_rental Car_model="R-023" Staff_ID="B004"></Car_rental>
        <Car_rental Car_model="SA-38" Staff_ID="A002"></Car_rental>
    </Car_rentals>
</Trip>
<Trip Trip_ID="T0002">
    <Department Department_ID="AA001" Salary="35670"></Department>
    <Car_rentals>
        <Car_rental Car_model="MZ-18" Staff_ID="B001"></Car_rental>
    </Car_rentals>
</Trip>
<Trip Trip_ID="T0003">
    <Department Department_ID="AB001" Salary="30010"></Department>
    <Car_rentals>
        <Car_rental Car_model="SA-38" Staff_ID="A001"></Car_rental>
    </Car_rentals>
</Trip>
<Trip Trip_ID="T0004">
    <Department Department_ID="BA001" Salary="22500"></Department>
    <Car_rentals>
        <Car_rental Car_model="R-023" Staff_ID="C001"></Car_rental>
    </Car_rentals>
</Trip>
```