Entity-Relationship models : formal specification and comparison

Jean-Luc Hainaut

University of Namur
rue Grandgagnage, 21 - B-5000 Namur (Belgium)

This document is a summary of, and a complement to, the material that was presented in an introductory lecture delivered during the 9th conference on Entity-Relationship Approach. Due to space limit, the diagrams used during the presentation have not been included.

1. MOTIVATIONS AND OBJECTIVES

Much too often, the Entity-Relationship approach is merely perceived as a set of graphical conventions to grossly sketch the data structures of a future database. It is not uncommon to find in some papers or textbooks a strong confusion between the E-R model, diagrammatic representations of E-R constructs, and E-R-based database design methodologies. However, at present time, the E-R model (or E-R models, to be more precise) can be provided with axiomatic definitions, generally pertaining to logic or set theory. Such a sound theoretical foundation is important in order to give the E-R constructs and query languages precise semantics, to compare different E-R models or to design semantics-preserving schema restructuring, to mention only some applications.

This lecture will be based on an extended relational interpretation of E-R models. This interpretation makes use of an extended relational formalism that includes N1NF structures as well as direct representation of entities. This unique formalism will be used to define precisely the concepts that can be found in current E-R models, to examine the notion of E-R schema transformation (e.g. normalization) and to propose some extensions to describe implementation structures. Most of the following material has been presented more formally in [HAINAUT, 89], except for sections 5.7 and 6.

2. INTRODUCTION

Despite an apparent consistency and a common origin, the E-R models proposed in the literature for 15 years present more or less important differences as to their terminology, their graphical representation, and even the very definition of their basic building blocks. Hence the need for a *reference model* according to which any current E-R model can be defined as a specialization.

When compared with E-R formalisms, it must be admitted that the *relational model* seems particularly attractive due to its simple and completely formalized definition. On the other hand, the standard relational model is much too poor to be used as a conceptual modeling tool, essentially because of its lack of direct support for the concept of entity and that of structured value. Several proposals have tried to overcome these limitations [CODD,79] [MAKINOUCHI,77]. We shall use such an extended relational model as a reference model to analyze and extend E-R models. Our purpose **is not to define rules for translating** E-R constructs into relational structures, but to state a semantics-preserving relational interpretation of these constructs. In addition, an E-R reference model **is not one more E-R model**. In particular, it is not intended as a modeling tool to be used by designers. Consequently, its relational-like notation and the lack of consistent and comprehensive graphical reprensetation must not be considered drawbacks.

The presentation is organized into five sections. In the first one (3) the principles of an extended relational formalism are presented. In the second and third sections (4 and 5), the concepts of standard and advanced E-R models are given a relational interpretation. In the fourth section (6), some direct and indirect results of the relational theory are applied to E-R constructs, particularly in the realm of schema transformation. The last section (7) is dedicated to some extensions of the E-R model that allow it to specify technical data structures.

3. AN EXTENDED RELATIONAL MODEL

3.1 Domain, basic domain and subdomain

A domain is any declared set of similar elements. Some domains are basic (or predefined) *value domains* such as *integer, real, date* or *string.* The basic *entity domain*, which is given the name **entities**, is an arbitrary set of concrete or abstract elements that can be used to denote entities of the Universe of Discourse [HALL,76],[CODD,79],[LEE,86]. A subdomain is a named domain defined as a subset of another (sub)domain or of a constructed domain. A named domain is either a basic domain or a subdomain. *Note* : a *relation* is a set of similar elements and can therefore be considered as a domain by itself.

```
NAME : char(25)
STREET : NAME
```

```
PERSON, DEPARTMENT, PRODUCT : entities
WORKMAN, EMPLOYEE : PERSON
```

3.2 Constructed domain

A constructed domain is defined by a *domain constructor*, and is a set elements obtained by applying set and/or relational operators on domains, or the elements of which are explicitly listed. There are four domain constructors.

• The ***cartesian constructor*** defines the cartesian product of one or several domains. Each partnership of a domain in the product is called an *attribute* of the product. When an attribute and its domain are given the same name, the specification of the latter is dropped. CITY and ADDRESS are subdomains based on cartesian constructed domains.
• The ***powerset constructor*** defines the set of subsets of a domain. A constraint is stated about the size of the subsets by declaring their minimum (`min`) and maximum (`max`) size with expression `[min:max]`. Omitting the `min` value means `0`, and _ is denoted with `*`. In particular, the notation `[*]` denotes sets of any size. Examples : CHR-NAMES (elements are sets of 1 to 4 NAME values), GANG (elements are non empty sets of WORKMAN elements) and SET-OF-GANGS (elements of (possibly empty) set of GANG elements).
• The ***algebraic constructor*** defines a set of elements as the result of the evaluation of an algebraic expression on domains (use of union, difference, relational product, project, select, join, intersect, divide, etc). *Note :* the cartesian and powerset constructors are basically algebraic operators; however, due to their importance they are presented separately. CUSTOMER and OPTIONAL-INTEGER are subdomains based on algebraic constructed domains. See also MAN and WOMAN in examples 3.3.
• The ***list constructor*** defines a set of elements comprehensively by the list of the denotations of each of them (DAY-OF-WEEK). The *null set* comprises one special element called the *null* element, denoted by O,/. It is compatible with (and therefore can be added to) any domain.

```
CITY : (ZIP-CODE:integer,CITY-NAME:char(30))
ADDRESS : (NUMBER,STREET,CITY)
```

```
CHR-NAMES : NAME[1:4]
GANG : WORKMAN[1:*]
SET-OF-GANGS : GANG[*]
```

```
CUSTOMER : (PERSON - EMPLOYEE) ≈ DEPARTMENT
```

```
DAY-OF-WEEK:{'SUN','MON','TUE','WED','THU','FRI','SAT'}
OPTIONAL-INTEGER : integer ≈ {O,/}
```

3.3 Relation schema

A relation schema is defined by the cartesian constructor and is given a name. The attributes of the cartesian product are those of the relation schema. Note that the notions of relation schema and of cartesian subdomain are redundant. However, both are maintained due to the relational origin of the current concepts. Moreover, let's remember that our purpose is not to define a new relational model but to lay down a formal framework to describe, compare and extend E-R models.

```
desc-of-PERSON(PERSON,NAME,CHR-NAMES,ADDRESS,SEX,ACCOUNTS:NUMBER[*])
```

```
WOMAN : desc-of-PERSON(SEX='F')[PERSON]
MAN : desc-of-PERSON(SEX='M')[PERSON]
MARRIAGE(wife:WOMAN,husband:MAN,DATE)
DIVORCE(MARRIAGE,DATE)
```

```
SALES(salesman:EMPLOYEE,buyer:CUSTOMER,PRODUCT,SALES-DATE:date)
```

```
LINE(PRODUCT,QTY:integer)
desc-of-ORDER(ORDER,ORDNUM,DATE,CUSTOMER,ADDRESS,LINES:LINE[1:20]
```

3.4 Integrity constraints

We shall mention three classes of constraints that will be of particular importance in the following, namely the *dependency* constraints, the *key* constraints and the *inclusion* constraints.For 1NF relation schemata, the concept of functional, multivalued and join dependency, as well as that of key, are as usual, except that they have been extended to derived relation schemata defined by algebraic constructors (e.g. key of desc-of-ORDER[LINES] and the FD on ST*TJ). When no ambiguity can arise, a key will be denoted by continuously underlining the names of the attributes. A functional dependency will be specified only if it is not implied by the keys. For N1NF schemata, it should be reminded that each value of a powerset attribute that participates in a key is a value set. If elements of these value sets are meant instead, the [*] notation will be used, as in the ACCOUNTS[*] key, stating that no two *desc-of-PERSON* tuples can share the same CODE value in their ACCOUNTS value sets.

key(desc-of-PERSON) : PERSON key(desc-of-PERSON) : NAME,CHR-NAMES,ADDRESS key(desc-of-PERSON) : ACCOUNTS[*] desc-of-PERSON(PERSON,NAME,CHR-NAMES,ADDRESS,SEX,...)
key(DIVORCE) : MARRIAGE
key(desc-of-ORDER[LINES]) : PRODUCT
SALES : buyer,SALES-DATE ⌠∅ salesman
desc-of-ORDER : CUSTOMER ⌠∅ ADDRESS
PROSPECT(SALESMAN,REGION,PRODUCT) PROSPECT : REGION ∅∅ PRODUCT
TJ(TEACHER,SUBJECT) ST(STUDENT,TEACHER) ST*TJ : STUDENT,SUBJECT ⌠∅ TEACHER

The inclusion constraint is an inter-domain constraint stating that a (generally constructed) domain is a subset of (∏), or equals (=), another one.

ST[TEACHER] ∏ TJ[TEACHER]
desc-of-PERS[PERSON] = PERSON
SUPPLY(CUSTOMER,PRODUCT,STORE,DATE) AVAILABILITY(PRODUCT,STORE,DATE) SUPPLY[PRODUCT,STORE,DATE] ∏ AVAILABILITY

4. RELATIONAL EXPRESSION OF BASIC E-R CONSTRUCTS

4.1 Entity type

An entity type E is represented by an entity domain with name E:

```
CUSTOMER : entities
ORDER : entities
```

4.2 Attributes of an entity type

A *descriptive relation schema* named, for instance, desc-of-*E* is defined on the entity domain E - by means of an *entity attribute* with name E - and on relational attributes corresponding to attributes A_i of E. Entity attribute E is, by definition, a relational key of descriptive schema desc-of-*E* . *Note:* decoupling the relational specification of an entity type from that of its attributes allows the definition of entity types without attributes.

```
desc-of-CUSTOMER( CUSTOMER,
                  CUS-NUM : num(6),
                  CUS-NAME : char(25),
                  CUS-ADDRESS : char(40),
                  CUS-ACCOUNT : num(10) )
desc-of-ORDER   ( ORDER,
                  ORDER-NUM : integer,
                  DATE : date)
```

E-R *multivalued* attributes are represented by relational attributes based on a powerset domain (CHR-NAMES); representation of E-R *compound* attributes is based on cartesian domains (ADDRESS). E-R optional attributes expression will be defined on domains including the *null element* (SPOUSE-NAME), or can be defined as multivalued attributes with cardinality [0:1].

```
NAME : char(10)
STD-ADDRESS : ( NUMBER: integer,STREET: NAME,CITY: NAME)
desc-of-PERSON( PERSON,
                PNAME : char(30),
                CHR-NAME : NAME[1:4],
                ADDRESS : STD-ADRESSE,
                SPOUSE-NAME : char(30) ≈ {O,/} )
```

4.3 Relationship type

A *relationship* (*type*) is defined by a collection of 2 or more entities (entity types), each of them playing a specific *role* in this collection. A relationship type R, with roles ri played by entity types Ei is represented by an *associative relation schema* R with attributes ri defined on entity domains Ei. For each attribute Aj of relationship type R, a relational attribute Aj is defined in the relation schema R. If an entity domain appears more than once, the corresponding attributes will be named to solve the ambiguity (recursive or circular relationship types).

```
SENDS ( SENDER: CUSTOMER,
        ORDER)
```

```
SUPPLY( SUPPLIED-ORDER: ORDER,
        SUPPLIED-PRODUCT: PRODUCT,
        SUPPLIER,
        SUPPLIED-QUANTITY: num(5))
```

A more concise representation can be proposed when R is functional (i.e. binary and many-to-one) and has no attributes, by *joining* its associative relation schema with the descriptive relation schema of the *image* entity type. *desc-of-ORDER'* is obtained by joining *SENDS* with *desc-of-ORDER* (see 4.2).

```
desc-of-ORDER'(  ORDER,
                 ORDER-NUM: ..,
                 DATE: ..,
                 SENDER: CUSTOMER)
```

4.4 Entity type identifier

An entity type identifier is described by a candidate key of the corresponding descriptive relation schema. *Hybrid* or *local* identifiers, using an *identifying relationship type* , can be described easily in the concise expression of functional relationship types (e.g. *desc-of-MONTHLY-SALES*). An entity type with a hybrid identifier is sometimes called a *weak entity type*.

```
desc-of-CUSTOMER(CUSTOMER, CUST-NUM, CUST-NAME, etc)
desc-of-MONTHLY-SALES(MONTHLY-SALES, PRODUCT,DATE, QTY)
```

4.5 Cardinality constraints of a relationship type

The cardinality constraint of role `rk` played by entity type `Ek` in relationship type `R` is defined by a couple `i-j` of integers stating that any `Ek` entity must play role `rk` in `m` relationships `R`, such that `i_m_j`. That constraint translates as follows in the relational representation,

$$\forall e \quad E_k, \; i\_|R(r_k=e)|\_j$$

The most common values of `i-j` are `0-1`, `1-1`, `0-_`, `1-_`. In these cases, simpler expressions can be proposed as follows.

- If `i = 1`, then `E = R[rk]`
- If `i = 0`, then the latter constraint does not stand.
- If `j = 1`, then `R(r1, ..,rk, ..,rn)`
- If `j = _`, then the latter constraint does not stand.

An E-R schema SUPPLY in which `ORDER` plays the role `supplied-order` with cardinality `1-1`, `PRODUCT` plays the role `supplied-product` with cardinality `0-_`, `SUPPLIER` plays the role `supplier` with cardinality `0-_`, is losslessly expressed as follows :

```
SUPPLY(   supplied-order: ORDER,
          supplied-product: PRODUCT,
          supplier: SUPPLIER)
ORDER = SUPPLY[supplied-order]
```

5. RELATIONAL EXPRESSION OF EXTENDED E-R CONSTRUCTS

5.1 Extended identifier

Some extensions for entity types : no identifier, more than one identifier, hybrid identifier composed with more than one relationship type (expressed below on the concise notation) or with no attributes.

descr-of-TRANSFER(TRANSFER,FROM,TO,DATE,PRODUCT,QTY)
desc-of-LINE-OF-ORDER(LINE-OF-ORDER, ORDER,PRODUCT,ORD-QTY)
desc-of-EMPLOYEE(EMPLOYEE, corp-EMP-ID, local-EMP-ID,DEPARTMENT,etc)

Some extensions for relationship types : more than one identifier, identifier comprising a strict subset of all *many* roles (i.e. with cardinality i-*), identifier comprising attributes.

ASSIGN(ORDER,PRODUCT,SUPPLIER,ASS-QTY)
LINE-of-ORDER(ORDER,LINE-NUMBER,PRODUCT,ORD-QTY)

5.2 Generalization / Specialization hierarchy [CHEN,76][SMITH,77][CODD,79]

The most straightforward relational expression of such hierarchies is by the entity subdomain.

```
PERSON : entities
MAN : PERSON
WOMAN : PERSON
desc-of-PERSON(PERSON,PERS-NUM,PERS-NAME,PERS-ADDRESS,SEX)
desc-of-MAN(MAN,MILITARY-STATUS)
MARRIAGE(MAN,WOMAN,DATE)
RECORD(PERSON,SKILL)
```

A specialization condition can be stated to precise the common specific properties of the members of a subtype. Covering and disjunction properties can be specified as well.

MAN : desc-of-PERSON(SEX='M')[PERSON]
PERSON = MAN ≈ WOMAN MAN ↔ WOMAN = {}

Inheritance can be given a relational interpretation through natural and outer joins :

Downward inheritance : desc-of-MAN * desc-of-PERSON
Upward inheritance : desc-of-MAN Λ* desc-of-PERSON

5.3 Category [CODD,79][ELMASRI,85]

"A category is a set of entities from one or several entity types, that play a role in a relationship, that generalize other entities, or that specialize other entities" [ELMASRI,85]

```
CAR, TRUCK, PERSON, CORPORATION : entities
OWNER : PERSON ≈ CORPORATION                    (category)
VEHICLE : CAR ≈ TRUCK                           (category)
OWNERSHIP(OWNER, VEHICLE)
EMPLOYEE : entities
FULLTIME-EMPLOYEE : EMPLOYEE                    (category)
```

5.4 Grouping aggregation [CODD,79][HAMMER,81][MEES,87]

A GROUPING, or COVER AGGREGATION is an entity type each entity of which stands for a subset of other entities.

```
SHIP, PLANE, PERSONNEL : entities
CONVOY : SHIP[*]                                (homogeneous GA)
CONVOY' : (SHIP ≈ PLANE ≈ PERSONEL)[*]          (heterogeneous GA)
```

5.5 Multidomain role

A role of a relationship type can be played by an entity of one among several types.

```
PROPERTY(VEHICLE: CAR, OWNER: EMPLOYEE ≈ SERVICE)
```

5.6 Multivalued role [JUNET,87]

A role of a relationship type can be played by a set of entities.

```
REPAIR(FAILURE, GANG: WORKMAN[1:*])
```

5.7 Valence [BAUMANN,89]

This concept extends the role of a relationship type by allowing, among others, an element playing a role to be absent, or to be a set of elements (cardinality constraints can be imposed on the size of this set), or to be drawn from more than one entity type. Many aspects of the valence concept can be specified through multi-domain and multivalued roles. The final *consists-of* example from [BAUMANN,89], section 2 (stating that *a module consists of a code, a documentation and either a PASCAL source or a set of C sources, one of the latter being the "main", the others, if any, being the "includes"*) can be described as follows.

```
MODULE,CODE,DOC,PASCAL-SOURCE,C-SOURCE : entities
consists-of( MODULE,CODE,DOC,
```

```
            source : PASCAL-SOURCE ≈ (  main:C-SOURCE,
                                        include:C-SOURCE[0..*])
          )
```

5.8 Relationship type on relationship types

A role of a relationship type can be played by a relationship type.

```
ORDER,PRODUCT,INVOICE : entities
LINE-of-ORDER(ORDER,PRODUCT,ORD-QTY)
LINE-of-INVOICE(INVOICE,LINE-of-ORDER,INV-QTY)
```

5.9 Complex object

A complex, or structured, object [ABITEBOUL,87] [HULL,87] is made up of values and/or other objects. It can be seen as an entity, some attribute values of which are entities. In current models, that concept is formalized by N1NF or Nested relations, or by special relationships such as *part-of* between objects. A traditional example describes *parts* that can be made up of other *parts* :

```
PART : entities
PART-SUBPART(PART, SUBPARTS: PART[0:*] )
```

```
PART : entities
def-of-PART(PART, COMPOSITION: def-of-PART [0:*] )
```

```
PART(PART-NUMBER, SUBPARTS : PART [0:*] )                (not E/R)
```

5.10 Binary models [BRACCHI,76][VERHEIJEN,82]

Binary models can be compared with E-R models as far as conceptual modeling is concerned. One of the most popular binary models is NIAM [VERHEIJEN,82]. Its concepts can be interpreted as follows : NOLOT –> entity subdomain, LOT –> simple value domain, naming bridge and idea –> binary relation schema, roles –> attributes. Concepts such as subtype, uniqueness, totality, etc, are thoroughly represented in the extended relational model. *Note* : a NIAM relationship has no name, but its two role names are of particular importance. Therefore a rule to name relation schemata systematically has to be adopted.

NOLOT and LOT :

```
  CAR-MAKER, CAR-MODEL, CAR : entities
  DESTROYED-CAR : CAR
  MAKE, COUNTRY-NAME, NAME, CAR-ID : char...
  DATE : date
```

Naming bridges :

```
  CARMAKERhasMAKE(has:CAR-MAKER, of:MAKE)
  CARMAKERhasCOUNTRY(located-in:CAR_MAKER,of:COUNTRY_NAME)
  CARMODELhasNAME(has:CAR-MODEL, of:NAME)
  CARhasID(id-by:CAR,identifies:CAR-ID)
  DESTROYEDCARhasDATE(on:DESTROYED_CAR,of:DATE)
```

Ideas :
`MODELofCARMAKER(made-by:CAR-MODEL,produces:CAR-MAKER)`
`CARofMODEL(has:CAR,of:CAR-MODEL)`

Total and Unique constraints :
`CARMAKERhasMAKE[has] = CAR-MAKER`
`CARMODELhasNAME[has] = CAR-MODEL`
`MODELofCARMAKER[made-by] = CAR-MODEL`
`CARhasID[id-by] = CAR`
`key(MODELofCARMAKER * CARMODELhasNAME) : produces,of`

5.11 Multisets, lists and bags

Due to the set-theoretic nature of the relational interpretation of the E-R concepts, the concepts of multisets or bags (i.e. collections of non necessarily distinct elements) cannot be represented easily. Neither can be the notion of list (collection of ordered elements). These concepts will be found in some extended E-R models [PARENT,86] and in object-oriented models.

6. APPLICATION OF THE RELATION THEORY TO E-R SCHEMATA

The relational nature of the formalism used to express the E-R constructs allows us to apply the results of the relational theory.

6.1 Relational concepts applied to E-R models

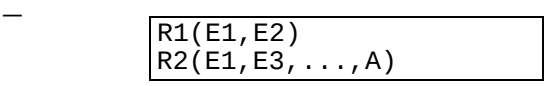
- Functional and Multivalued dependencies in an entity type
- Functional and Multivalued dependencies in a relationship type
- Normal forms of an Entity type
- Normal forms of a Relationship type

6.2 Specific O-based transformations derived from the relational theory

The existence of entity domains, and their use as attributes in descriptive and associative relation schemata require some adaptation and development of the classic results of the theory. We shall outline some examples. Hereafter, A designates a list of 0, 1 or more attributes.

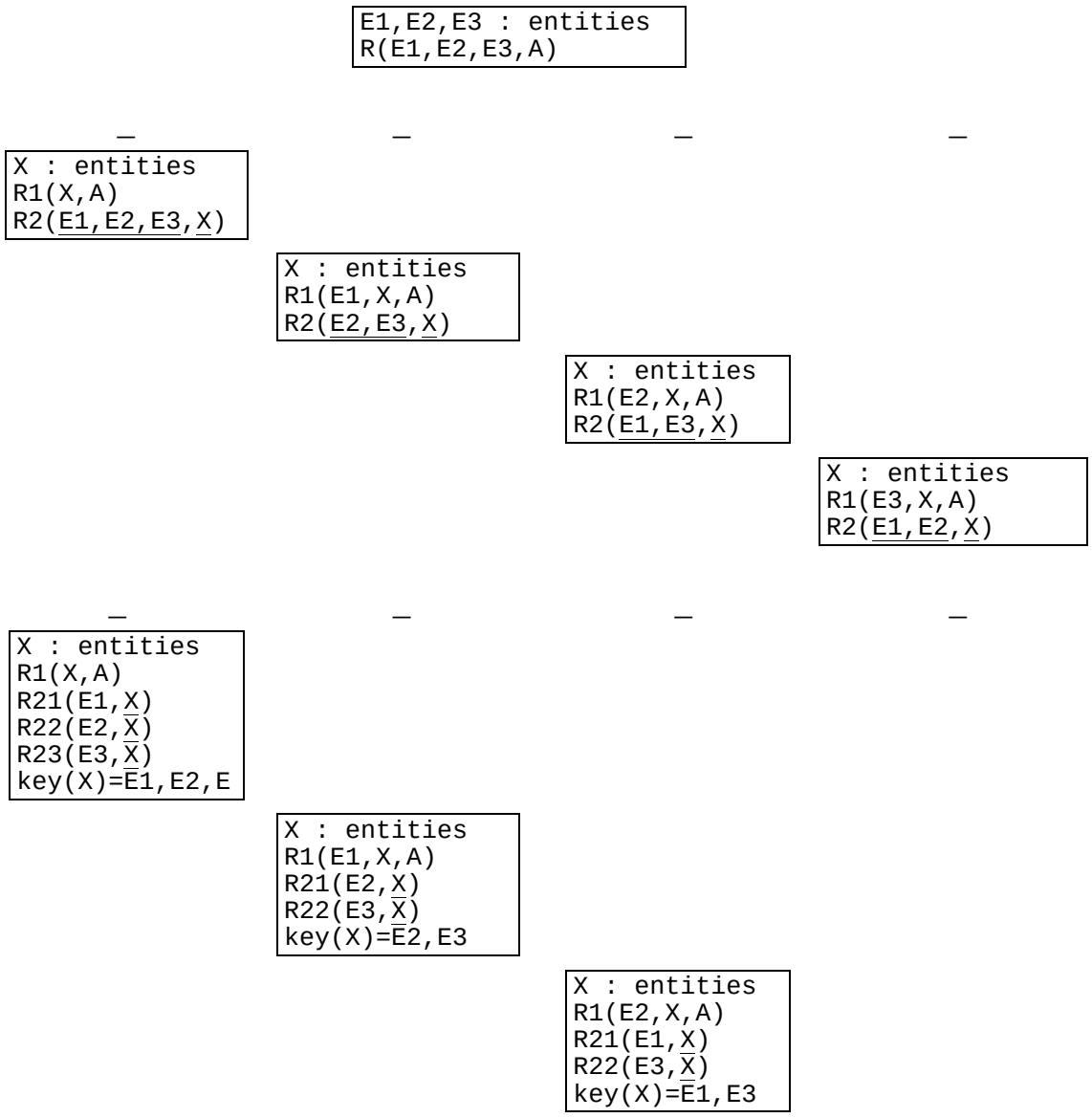
1. *Direct application of the relational theory*. E.g. : decomposition theorem [FAGIN,77] [MAIER,83]. Let's remind that the MD can be a FD.

```
E1,E2,E3,... : entities
R(E1,E2,E3,...,A)
E1 ∅∅ E2
```



2. *Indirect application of the relational theory*. E.g. : the extension transformation [HAINAUT,90]. This general semantics-preserving transformation allows the introduction/removal of an entity type (called X hereafter) in an E-R schema. Some important applications are as follows (with some details dropped). Other applications will be found in the paper mentioned above.

   2.a *Transformation of a n-ary relationship type* (n_2). As an example, the 3-ary relationship type R can be transformed into the following 8 equivalent schemata (other schemata can be produced that way). Note that the last 4 schemata use binary relationship types only.



```
X : entities
R1(E3,X,A)
R21(E1,X)
R22(E2,X)
key(X)=E1,E2
```

Note : a notation such as `key(X)=E2,E3` is a shorthand for `key(R21*R22)=E2,E3`

2.b *Decomposition of an entity type*. The extension transformation allows the transfer of attributes (and of functional relationship types as well) to a new entity type. In the application below it is used to eradicate a transitive FD.

```
E : entities
descr-of-E(E,A1,A2,...,Am,An)
A2 ⊂ A3,...,Am
```

/

```
E,X : entities
descr-of-E(E,A1,X,An)
descr-of-X(X,A2,A3,...,Am)
```

6.3 Some practical examples

1. *Normalization of an entity type*. Elimination of a transitive FD.

```
ORDER : entities
desc-of-ORDER(ORDER,ORDNUM,DATE,CUSTNUM,ADDRESS)
desc-of-ORDER : CUSTNUM ⊂ ADDRESS
```

—

```
ORDER, CUSTOMER : entities
desc-of-ORDER(ORDER,ORDNUM,DATE,CUSTOMER)
desc-of-CUSTOMER(CUSTOMER,CUSTNUM,ADDRESS)
```

2. *Normalization of a relationship type*. Elimination of a MD.

```
PROSPECT(SALESMAN,REGION,PRODUC
T
PROSPECT : REGION ⫫ PRODUCT
```

—

```
VISITS(SALESMAN,REGION)
BUYS(REGION,PRODUCT)
```

3. *Decomposition of a relationship type*. The irreducible relationship type PROGRAM is transformed into equivalent schemata that comprise binary relationship types only (other equivalent schemata could have been produced based on the same principle).

```
TEACHER,SUBJECT,DATE:entities
PROGRAM(TEACHER,SUBJECT,DATE)
```

```
TEACHER,SUBJECT,DATE,PROG:entitie
s
R21(TEACHER,PROG)
R22(SUBJECT,PROG)
R23(DATE,PROG)
key(PROG)=TEACHER,SUBJECT,DATE
```

```
TEACHER,SUBJECT,DATE,SCHEDULE:entitie
s
R21(SUBJECT,SCHEDULE)
R22(DATE,SCHEDULE)
R1(TEACHER,SCHEDULE)
key(SCHEDULE)=SUBJECT,DATE
```

7. EXTENDING THE E-R MODELS TO REPRESENT IMPLEMENTATION CONSTRUCTS

An implementation schema is an abstraction of a DBMS schema. It specifies conceptual structures + access structures. In order to allow the complete specification of a DBMS schema, the E-R model is sometimes complemented by concepts pertaining to the technical realm [HAINAUT,86] [SCHULDT,86]. Two main additions : access path and file.

1. Access path

   Definition : abstraction of mechanisms that give quick, selective access to data satisfying some selection criteria. Examples : index, hash keys, set type (CODASYL), search key, indexed set (CODASYL), parent/child path (DL/1), file coupling (ADABAS)

   Specification : source attributes ⊂∅ target attributes (similarity with the FD concept).

```
desc-of-ORDER(ORDER, ORDNUM, DATE, CUSTOMER)
path(desc-of-ORDER) : ORDNUM ⊂∅ ORDER
path(desc-of-ORDER) : DATE ⊂∅ ORDER
path(desc-of-ORDER) : ORDER ⊂∅ ONUM,DATE
path(desc-of-ORDER) : ORDER ⊂∅ CUSTOMER
path(desc-of-ORDER) : CUSTOMER ⊂∅ ORDER
```

2. File

   Definition : abstraction of any kind of repository for technical entities. Examples : dbspace, tablespace, dataset, area, realm.

   Specification : similarity with the subdomain concept (a file is some set of entities).

```
FILE-OF-ORDER, FILE-OF-CUST : file
ORDER : FILE-OF-ORDER
```

```
CUSTOMER : FILE-OF-ORDER
CUSTOMER : FILE-OF-CUST
```

8. REFERENCES and FURTHER READINGS

ABITEBOUL,87, ABITEBOUL, BEERI, "On the power of languages for the manipulation of complex objects", INRIA technical report, 1987

BAUMANN,89, BAUMANN, "Valences: A new Relationship Concept for the Entity-Relationship Model", Proc. 8th Conf. on Entity-Relationship Approach, Toronto, Oct. 1989

BRACCHI,76, BRACCHI, PAOLINI, PELEGATTI, "Binary logical associations in data modelling", in *Modelling in data base management systems*, North-Holland, 1976

CHEN,76, CHEN, "The entity-relationship model - toward a unified view of data", ACM TODS, Vol. 1, N° 1, 1976

CODD,79, CODD, "Extending the Database Relational Model to capture more meaning", ACM TODS, Vol.4, N°4, 1979

COLLART,88, COLLART, JORIS, "Etude théorique et pratique d'extensions au modèle Entité-Association", Master Thesis, Institut d'Informatique, FUNDP, Namur, 1988 (french)

ELMASRI,85, ELMASRI,WEELDREYER,HEVNER, "The caregory concept : an extension to the entity-relationship model", Data and Knowledge Engineering, Vol.1, n°1, 1985

FAGIN,77, FAGIN, "Multivalued dependencies and a new normal form for relational databases", ACM TODS, Vol. 2, No 3, 1977

HAINAUT,86, HAINAUT, "Conception assistée des applications informatiques - 2. Conception de la base de données", MASSON, Paris 1986

HAINAUT,89, HAINAUT, J-L, *A Generic Entity-Relationship Model*, in Proc. of the IFIP TC8 / WG8.1 Work. Conf. on *Information System Concepts : An In-depth Analysis*, Oct. 1989, Falkenberg & Lindgreen (Ed.), North-Holland.

HAINAUT,90, HAINAUT, "Semantics-preserving Schema Transformation in Object-based Models", Research Report, Institut d'Informatique, FUNDP, March 1989.

HALL,76, HALL, OWLETT, TODD, "Relations and Entities", in *Modelling in Data Base Systems*, North-Holland, 1976

HAMMER,81, HAMMER,MCLEOD, "Database description with with SDM : a semantic database model", ACM TODS, Vol. 6, n°3, 1981

HULL,87, HULL, "A survey of theoretical research on typed complex database objects", in *International Lecture Series in Computer Science*, Academic Press, 1987

JUNET,87, JUNET, "Design and implementation of an extended entity-relationship database management system", in Proc. of the 5th Conf. on E-R approach, North-Holland, 1987

KOZACZYNSKY,87, KOZACZYNSKY, LILIEN, "An extended Entity-Relationship (E^2R) database specification and its automatic verification and transformation", Proc. 6th Entity-Relationship Conf.,1987

LEE,86, LEE, R. M., *Logic, Semantics and Data Modeling : an Ontology*, in Proc. of the IFIP TC2/WG 2.6 Work. Conf. on *Knowledge and Data* (DS-2), North-Holland,1986

MAIER,83, MAIER, "The Theory of Relational Databases", Computer Science Press, 1983

MAKINOUCHI,77, MAKINOUCHI, "A consideration of normal form of Not-Necessarily-Normalized Relations in the Relational Data Model", Proc. 3rd VLDB Conf., Tokio, 1977

MEES,87, MEES, PUT, "Extending a dynamic modelling method using data modelling capabilities : the case of JSD", in Proc. of the 5th Conf. on E-R approach, North-Holland, 1987

PARENT,86, PARENT, SPACCAPITIETRA, "Enhancing the operational semantics of the entity-relationship model", Proc. of IFIP WG 2.6 WC on Data Semantics, North-Holland, 1986

PECKHAM,88, PECKHAM, MARYANSKI, "Semantic Data Models", ACM Comp. Surveys, vol.20,n°3, 1988

SCHEUERMANN,80, SCHEUERMANN, SCHIFFNER, WEBER, "Abstraction capabilities and invariant properties modelling within the E/R approach", in Proc. of the 1st Conf. on E-R approach, North-Holland, 1980

SCHULDT,86, SCHULDT, "ER-based Access Modeling", Proc 5th Int. Conf. on Entity-Relationship Approach, Spaccapietra (Ed.), 1986

SENKO,

SMITH,77, SMITH, SMITH, "Database abstractions : aggregation and generalization", ACM TODS, Vol. 2, N° 2, 1977

VERHEIJEN,82, VERHEIJEN, Van BEKKUM, "NIAM : An information system analysis method", in *Information Systems design methodologies*, North-Holland, 1982

***ABRIAL,74**, ABRIAL, "Data Semantics", in *Data base management*, North-Holland, 1974

***BACHMAN,69,** BACHMAN, "Data Structure Diagrams", Data Base, 1, N°1, ACM SIG on Business Data Processing, 1969

***BODART,88,** BODART, PIGNEUR, "Conception assistée des applications informatiques - 1. Etude d'opportunité et analyse conceptuelle", MASSON, Paris 1988

***CADELLI,87,** CADELLI, CHARLOT, DELCOURT, HAINAUT, "L'atelier de conception de base de données ORGA", Technical report, Institut d'Informatique, FUNDP, Namur, April 1987

***CODASYL,62**
Development Committee : An information algebra, Com. ACM, 5, n° 4, April 1962

***CODD,70,** CODD, "A relational model for large, shared data banks", Commun. ACM Vol. 13, N° 6, 1970

***DATE,85,** DATE, "An introduction to database systems", Vol II, Addison-Wesley, 1985

***DATE,86,** DATE, "An introduction to database systems", Vol I, Addison-Wesley, 1986

***DATE,86b,** DATE, "Relational Databases - Selected writings", Addison-Wesley, 1986

***FIKES,85**, FIKES, KEHLER, "The role of frame-based representation in reasoning", CACM, Vol. 28, n° 9, Sept. 1985

***GYSSEN,88,** GYSSEN, VAN GUCHT, "The powerset algebra as a result of adding programming constructs to the nested relational Algebra", Proc. SIGMOD Conf., Chicago, 1988

***HAINAUT,74,** HAINAUT, LE CHARLIER, "An extensible semantic model of data base and its data manipulation language", Proc. IFIP Congress, North-Holland, 1974

***HAINAUT,81,** HAINAUT, "Theoretical and practical tools for data base design", in Proc. Intern. VLDB conf., ACM/IEEE, 1981

***HAINAUT,87,** HAINAUT, CADELLI, CHARLOT, DELCOURT, "Conception de bases de données - Eléments méthodologiques", Research report, Institut d'Informatique, FUNDP, Namur, March 1987

***HAINAUT,88,** HAINAUT, "Non conventional transformations of data base schemata", Draft Research report, Institut d'Informatique, FUNDP (in french), Dec. 1988

***HAINAUT,89,** HAINAUT, "Semantic equivalence of Data Base Schemata - A new family of formal tools", Research Report, Institut d'Informatique, FUNDP, June 1989 (submitted for public.)

***MOTRO,87,** MOTRO, "Superviews: Virtual Integration of Multiple Databases", IEEE Trans. on Soft. Engin. Vol. SE-13, N°7, July 1987

***TEOREY,86,** TEOREY,YANG,FRY, "A logical design methodology for relational databases using the Extended Entity-Relationship model", ACM Computing Surveys, Vol. 18, N° 2, June 1986.

***SHIPMAN,81,** SHIPMAN, "The functional data model and the data language DAPLEX", ACM TODS, Vol. 6, N° 1, 1981

***ULLMAN,88,** ULLMAN, "Principles of database and knowledge-base systems", Computer Science Press, 1988

***WOODS,75,** WOODS, "What's in a link : foudation for semantic networks", Representation and Understanding, Academic Press, 1975

***YAESCHKE,82,** YAESCHKE, SCHEK, "Remarks on the Algebras on Non First Normal Form Relation", Proc. 1st PODS, Los Angeles, 1982