CONVERTIBILITY IN DATABASES

J.H. ter BEKKE
Department of Mathematics, Delft University of Technology, 2628 BL Delft, The Netherlands

Received 28 April 1980; revised version received 14 August 1980

Relational database, data semantics, convertibility

1. Introduction

The relational model for databases [2] is based on a small number of mathematical concepts. For integrity reasons (i.e. to avoid unmanageable collections of data) some other concepts have been added to these. Examples are functional dependency and primary key. The latter concepts enable us to formalize invariant properties of objects from the reality and to maintain these properties in the database.

The concept of primary key reflects a property of an individual object but also a property of an object type. Because a specific type of primary key is connected with a specific base relation, it is stated that the relation contains data about a certain type of object. Unfortunately, this concept insufficiently guarantees that every object in the database corresponds with exactly one object in reality. Therefore, the concept of convertibility in databases is introduced. Convertibility suggests that base relations should be interpreted as collections of assertions (consisting of subject and predicate) instead of as collections of predicates (as usual in relational database theory). This concept visualizes the evaluation in data modelling from programming [5] to databases [3].

The concept of convertibility will be introduced with an example. This leads to new arguments for using only normalized relations (i.e. with simple attributes) in schemas of relational databases.

2. Basic concepts

A *domain* is a set of values of similar type. Let D_1, D_2, ..., D_n be a number of domains. The *cartesian product* $D_1 \times D_2 \times \cdots \times D_n$ is the set of n-tuples $(t_1, t_2, ..., t_n)$ such that $t_i \in D_i$. A *relation* R defined on these n domains is a (time-varying) subset of the cartesian product. An *attribute* represents the use of a domain within a relation. A *relational database* is a collection of relations defined on a number of simple (i.e. non-decomposable) domains. A *base relation* is defined independently of other relations in the database, i.e. a base relation is not derivable from other base relations. The collection of definitions of all base relations is called the *schema* of the database. The *content* of the database is the actual collection of tuples.

Suppose R is a base relation with (collections of) attributes X and Y. The attribute Y of R is *functionally dependent* on X of R if and only if each X-value in R has associated with it precisely one Y-value in R (at any one time). A *primary key* is a minimal collection of attributes with the property that each value denotes exactly one tuple of that relation. A *simple key* is a primary key which consists of only one simple attribute. A *compound key* is a primary key which consists of more than one attribute. A *foreign key* is an attribute which acts as a primary key in another relation.

3. Additional concepts

A database may be conceived as a model of a certain part of the reality. Anything from that reality which is relevant to be known or perceived is called an *object*. Any relevant characteristic of an object is called a *property*. An *objecttype* is a collection of objects with the same properties. The following table relates these concepts to basic concepts:

Concept in reality	Concept in database
object	tuple
objecttype	relation
property	attribute

Some of these concepts are used in the following example, in which a schema contains the following base relation:

object(city, person).

By indicating the primary key of this base relation we denote the type of objects that are represented in this relation: only properties belonging to this objecttype may occur in the base relation. If the primary key is underlined, then

object(city, person)

contains data about cities, alternatively

object(person, city)

contains data about persons. Note that the definitions above differ only in their primary keys.

4. Convertibility

Each object is essentially identified by its properties. Just as in reality, this involves some drawbacks in databases. First, it becomes unpractical to use an enumeration of properties to refer to an object. Secondly, it is possible that properties may change while we like to consider the object as before. To avoid these difficulties, primary keys (i.e. names) are used in databases for identification purposes. These primary keys represent a number of invariant properties and/or a number of added attributes (i.e. not user controlled). Examples might be birthdates and serial numbers in registrations about persons. The consequences of the introduction of this concept were partly given by Codd. The following example will show this.

Let a base relation contain data about persons. For each person the relation contains its name and city. Besides this an attribute called member id. has been added to these properties to simplify the registration. This attribute is the primary key of the base relation. Hence, the following content would be admissible:

person(member id.,	name,	city)
(0105,	Jansen,	Leiden)
(0106,	Peters,	Delft)
(0107,	Jansen,	Leiden).

With this content, registration cannot proceed satisfactorily. Suppose a transaction has to be processed on behalf of Jansen from Leiden. This transaction (which may result in a modification to the content of another base relation) cannot be processed when the properties name and city are used for identification, since these are ambiguous while the base relation meets the primary key property.

The previous ambiguity can be caused by:

– *Incomplete perception of the reality.*

In this case the base relation does not represent the proper collection of properties. The following base relation resolves this:

person(member id.,	name,	city,	birthdate)
(0105,	Jansen,	Leiden,	420412)
(0106,	Peters,	Delft,	450723)
(0107,	Jansen,	Leiden,	680605).

– *Redundant representation of the reality.*

Before entering a new tuple in a base relation, it must be confirmed that the tuple represents a new object. This procedure would result in the following content:

person(member id.,	name,	city)
(0105,	Jansen,	Leiden)
(0106,	Peters,	Delft).

The previous example illustrates that for each base relation with simple key the following *concept of convertibility* must hold:

– The collection of properties is functionally dependent on the primary key (identification rule).

– The primary key is functionally dependent on the collection of properties (confirmation rule).

A user controlled primary key may be conceived as a property. It is obvious that for such primary keys the concept of convertibility also holds.

Convertibility can also easily be extended for base relations with compound keys. Since all attribute values of compound keys must occur in other base relations as primary key values (according the rule of referential integrity [3]), compound keys may be conceived as (derivable) properties. It is trivial that in such a situation the concept of convertibility also holds.

The principle of convertibility also has consequences for allowable values in relational databases. It is commonly accepted that a primary key may not contain any nil-value (i.e. value at present unknown). It is clear that according to the concept of convertibility, nil-values in non-key attributes may not prevent confirmation.

There are two important currents in data structuring. First, data structures are studied in programming [5], where objects are identified by means of their properties (the concept of primary key is unknown). For example the definition of the objecttype person could be as follows:

person(name, city),

Data about a person can be conceived as a predicate. It could be described as 'having the properties of a name and a city'. Although the definition above contains a term for identification of the objecttype, no such term exists for individual objects.

Another important current in data structuring is identified by relational databases. The concept of primary key has always played a major role in databases [2]. With the introduction of this concept the requirement of identification by means of the properties, as illustrated above, has been neglected. However, the suggested close correspondence between relational calculus and relational databases has resulted in an inadequacy as illustrated below.

Let a base relation be defined as follows:

person(member id., name, city).

A tuple in a base relation is conceived as a predicate. The predicate could be described as 'having the properties of a member id., a name and a city', thereby completely neglecting the concept of primary key.

Another interpretation is obvious. Using the concept of convertibility, a tuple in a base relation could be conceived as an assertion [1]. This assertion consists of two components: a subject (representing the name of an object in the database) and a defining predicate (representing all properties of an object in the database). In the example above the subject would be 'a member id.' and the defining predicate would be 'having the properties of a name and a city', as in programming.

5. Consequences

In the previous section we have shown the importance of confirmation in databases. Just like other basic operations (update, delete and insert) this new operation also has consequences for the structure of base relations.

Normal forms have been introduced in databases mainly to avoid certain anomalies of modification operations. However, this was not the case with the first normal form. This normal form was mainly introduced for two reasons: to simplify implementation and to use non-procedural languages for querying the database [2]. With respect to the first normal form we can extend the arguments.

Let a base relation be defined as follows:

person(member id., name,
(repeating group: address)),

with member id. as primary key, name as property and address as a repeating group of properties. For confirmation only one combination of name and address is needed. This means that there exists no essential difference between a name and an address, so the definition above is not admissible for the schema of a relational database. (Note that splitting up this relation into normalized relations [2] does not solve the problem.)

According to the principle of convertibility, it is evident that a relation

person((repeating group: member id.),
name, address)

also is inadequate for the schema of a relational database.

Acknowledgment

The author is grateful to the referee and to F. Remmen and W. Kooyenga for comments on an earlier version of this paper.

References

[1] J.H. ter Bekke, M.S.A.: Modelling with Simple Assertions (in preparation).

[2] E.F. Codd, A relational model of data for large shared data banks, Comm. ACM 13 (6) (1970) 377–387.

[3] E.F. Codd, Extending the relational model to capture more meaning, ACM Trans. Database Systems 4 (4) (1979) 397–434.

[4] C.J. Date, An Introduction to Database Systems (Addison-Wesley, Reading, MA, 2nd ed., 1977).

[5] C.A.R. Hoare, Notes on data structuring, in: APIC Studies in Data Processing No. 8: Structured Programming (Academic Press, New York, 1972) 83–174.