First Order Normal Form for Relational Databases and Multidatabases

Witold Litwin M.Ketabchi Ravi Krishnamurthy
H.P.Labs, 1501 Page Mill Rd.
Palo Alto, CA 94303

1 Introduction

There has been considerable research effort in understanding the concept of a normal form introduced in [Codd70]. Admittedly, any relation in a well designed relational database should be in some normal form (e.g., Boyce-Codd normal form), and at least in 1NF. Some update anomalies and redundancies are then avoided. Normalization, however, does not improve the retrieval capabilities of a database, rather it requires additional equijoins in the query. Any relation that can be derived using a relational language, from unnormalized relations, can be derived from the corresponding decomposition as well. In other words, if D is a database, the expressible set of D, [Codd70], is the same regardless of the normality of relations in D. We recall that the expressible set is the set of all relation that can be obtained from those in D (called named set), through a 1-st order predicate calculus language, i.e., a relational language [Codd70].

Normal forms apply to relations individually. It appears that a similar concept can apply to a relational database as a whole. We propose a normal form for a relational database that we call 1-st order normal form (1ONF). We show that databases not in 1ONF may present severe restrictions for queries, unlike in the case of traditional normal forms. This can occur, even if each relation is as normalized as it is known. In practice, the expressible set of a database D not in 1ONF can be a strict subset of an equivalent 1ONF database. Similar situation can occur in the multidatabase environment in presence of semantic heterogeneity due to the database autonomy. Since it can be a natural trend to design relational (multi)databases not in 1ONF, users should be aware of this form, as they are of the traditional ones.

1.1 Rationale

Consider the well known S-P database about suppliers and parts.

```
S  (S#, SNAME)
P  (P#, PNAME)
SP (S#, P#, QTY)
```

Here S# is a key of each supplier, for which we assume a meaningful abbreviation (e.g., H for Hewlett Packard).

As an alternate, consider that a person in charge of orders keeps a separate paper file per supplier, which is not an uncommon practice. Suppliers can also have slightly different ways to deal with the orders (e.g., different set of attributes), justifying this approach. A natural tendency is to model these files as relations, e.g.:

```
H  (O#,P#,QTY)
IB (O#,P#,QTY)
DE (O#,P#,QTY,DDATE)
.... (etc.)
```

The relation names here uniquely determine the suppliers, i.e., O# is the key representing the order number. We assume the other two relations, (namely S and P are unchanged. The resulting scheme is normalized but may pose difficulty (if not deem impossible) posing some queries that would be possible in the previous scheme (with the DDATE attribute viewed as an addition). This is true for the following queries:

- Names of suppliers where parts are ordered, but QTY is equal to 30,
- Names of suppliers that could deliver the same part as a supplier where the part was ordered with QTY equal to 30.

The reason is that all these queries require a join on relation names; i.e., the relation name is a value for the join. Such joins cannot be formulated in SQL, nor in any relational language, since the 1-st order calculus does not support such operations. The only way to express the corresponding queries is to replace the joins with ORs or UNIONs, and clauses with the

corresponding constants, e.g. S.S# = 'H' OR S.S# = 'IB',... Such a query would not only be tedious but will be adequate for only some states of the database. If a supplier is added, then the query will have to be reformulated. Thus no relational query can express the above queries or any query requiring a join on relation names.

The S-P example is not an unusual case. An example using the historical stock market database was presented in [Kri91] with identical problems. Similar problems occur for probably any classical RDB application:

- Consider the well known C-S-P database, dealing with courses, students and professors. Courses are in classrooms that often have individual planning sheets indicating, opening and closing hours, special equipment to turn on/off, etc. A natural tendency is to have a relation per classroom, labeled with the classroom name.

- Consider the airlines database with a relation indicating flights and corresponding companies. Companies often have their own terminals. A natural tendency is to have also a relation per company, labeled with the airline key, e.g., TWA, giving more details about the flights. This is what one usually sees at a TV monitor in an airline terminal.

```
S   S#    SNAME        P   P#  PNAME
    H     HwPk             22  PC
    IB    MBI              33  Printer
                           44  cable

H   O#    P# QTY         IB  O#  P#  QTY
    a2    22 75              q5  22  150
    a9    33 200             q3  44  75

Reference Relation:

R   RelName   KeyVal   AttrName   AttrVal
        S        H      SNAME      HwPk
        S        IB     SNAME      MBI
        P        22     PNAME      PC
        P        33     PNAME      Printer
        P        44     PNAME      Cable
        H        a2      p#        22
        H        a9      p#        33
        H        a2     QTY        75
        H        a9     QTY        200
        IB       q5      p#        22
        IB       q3      p#        44
        IB       q5     QTY        150
        IB       q3     QTY        75
        S        H      S#         H
        S        IB     S#         IB
.... + tuples for the key values for
         all  tuples in all relations.
```

2 First Order Normal Form

2.1 Principle

To avoid these problems, one has to consider a database as the whole and transform some relations. This process of transforming relations is referred to as *database normalization*. One has to create a single relation using which any conceivable query be expressable as a first order expression. This relation is called *reference relation*. Then such a single relation for the database becomes the frame of reference for testing for 1ONF. Intuitively, a given relational scheme is in 1ONF if for every relational query that can be posed against the reference relation there is a corresponding relational query that can be posed on the given relational scheme. If not, then some relations or attributes have to be unified into one relation or attributes.

Consider the S-P database with one relation for each supplier, although for the sake of example we only consider H and IB. Then the relation R shown below would be the reference relation for the following sample database.

The reference relation provides a basis for evaluating the deficiency of the above scheme. In particular, we can ask example queries mentioned above. This deficiency can be corrected by unifying the H and IB relations as it was in the SP relation. The main outcome of such a unification is the transformation of some relation or attribute names (e.g., H,IB) into values.

Note the duality between a relation normalization and a database normalization. While a relation normalization breaks a relation into several relations, the database normalizations merges several relations into one. The price for a relational normalization are more procedural queries, the benefit is less redundancies and so usually less storage, and less update anomalies. The database normalization allows in contrast for queries that are otherwise cumbersome or impossible to formulate, while the price to pay are additional null values, and so, perhaps, more storage. However, the database normalization does not bring additional tuples to the database, i.e., is independent of relations normalization.

2.2 Formal definition

A relational database has a schema that defines relations, attributes and the tuples in the database conform to the schema.

Definition
A reference relation for a database consist of a quadruple relation with the following attributes:

1. `RelName`: The name of the relation.
2. `KeyVal`: The key value of the tuple (single value constructed for multiple column keys).
3. `AttrName`: The attribute name.
4. `AttrVal`: The value of the attribute for the attribute name in the tuple corresponding to the key value in the relation specified by the relation name.

This relation is populated by a tuple for every non-key value of every tuple in every relation. For that tuple, the relation name, attribute name the key value and the value of the attribute are associated with the tuple.

A given relational database scheme is in 1ONF if every relational query that can be posed against the reference relation can also be formulated against the given database scheme. However, typically, not all queries that can be posed against the reference relation are of interest to the application. Thus, we relax the definition to state that a database is in 1ONF if all queries of interest that can be formulated towards its reference relation, can also be formulated towards the database schema.

2.3 Multidatabase case

It should be noted that similar problems occur in the multidatabases. A database D that is to be interoperated with some D', might map a concept as an attribute name, while D' maps it as a value. This is a particular case of the semantic heterogeneity that can exist in multidatabases[Kri91]. It could happen in our example if the orders were not in S-P, but in a separate database, let it be O, managed by another department. Queries we discussed would become multidatabase queries to S-P and O databases. Even if each database were in 1ONF, the expressiveness of multidatabase queries would be affected.

Further, the database name itself may have information content, especially if there are large number of databases in a multidatabase system. Then it would be necessary to quantify and state condition on the database names.

In multidatabases, we therefore face not only the database normalization, but the multidatabase normalization. However, because of the autonomy of the databases in multidatabase environment, it can be impossible to apply the multidatabase normalization. It can be concluded that even if a relational language, e.g. SQL, suffices for each database D from some multidatabase M, since each D is in 1ONF, it can be insufficient for the entire M, if it was not in 1ONF. The general solution in this new environment is therefore to enhance relational languages into higher order languages [Lit89, Kri91].

The multidatabase normalization can be extended from the 1ONF by including the name of the database in the reference relation.

3 Conclusion

A relational database can have any of its relations in a normal form, and still present surprising anomalies with respect to expressiveness of queries. To avoid these problems, a relational database has to be in 1ONF. The database designers should be aware of the corresponding normalization principles. Since, 1ONF enhances the power of retrievals, and retrievals are usually the most frequent operation, 1ONF is a very important normalization.

In the multidatabase environment, there is also the need for the multidatabase first order normalization, if relational languages are to be used for multidatabase queries. Because of the autonomy, it may however be impossible to enforce it. Schematic discrepancies will therefore often exist and the general solution are higher order languages. This conclusion should be of interest to the designers and users of new relational languages proposed for multidatabase environment such as SQL- Access [Bal91], or Apple's DAL.

References

[Bal91] Balboni, J., H. "SQL Access Overview." *Proc. of IEEE Compcon-91*, pp114-119 1991.

[Codd70] Codd, E., F. "A relational Model of Data for Large Shared Data Banks", *CACM*, 13, 6, (June 1970).

[Kri91] Krishnamurthy, R., Litwin, W., Kent, W. "Language Features for Interoperability of Databases with Schematic Discrepancies" *Proc. of ACM-SIGMOD Conf.* SIGMOD Record, 20, 2, (June 1991), 40-49.

[Lit89] Litwin, W. & al. "MSQL : A Multidatabase Language", *Information Science*, 48, 2, (July 1989).