Local Nulls in Mobile and Distributed Summary Databases: Extended Report*

Darin Chan and John F. Roddick

School of Informatics and Engineering
Flinders University of South Australia
P.O.Box 2100, South Australia
{darin.chan,roddick}@infoeng.flinders.edu.au

Abstract

The concept and semantics of the null value in relational databases has been discussed widely since the introduction of the relational data model in the late 1960s. With the introduction of highly mobile, distributed databases, in order to preserve the accepted soundness and completeness criteria, the semantics of the null value needs to expand to reflect a localised lack of information that may not be apparent for the global/central database. This paper discusses an extension to the notion of nulls to include the semantics of ‘local’ nulls. The paper introduces local nulls in terms of amendments to the relational algebra and examines its impact on the query languages such as SQL.

1 Introduction

Codd and Zaniolo gave the semantics of null values as three-fold – *value unknown*, *value inapplicable* and *no information* (ie. unsure as to either of the others) (Codd 1970, Zaniolo 1984). A variety of research has since expanded on this – see particularly (Codd 1986, Codd 1987, Zaniolo 1984, Roth, Korth & Silberschatz 1989, Date 1986, Biskup 1983). In our work in mobile and distributed databases (Chan & Roddick 2003) we have found a requirement for an additional definition for the null value – that of a ‘local null’.

The context of this research is the maintenance of *maximal completeness* when summarised databases are used in a low capacity mobile environment. We use the term summarised here to refer to any database which holds, in whatever form may be appropriate, a fragment of some ‘relatively global’ database. The term ‘relatively global’ allows for a hierarchy of fragments.

That is, within a summarised database, there are attributes for which a value may exist in the global database but not in the local database. The lack of connectivity in a mobile environment may result in this summarised database being the best that is available and the context of our work is to maximise the usability of the data available.

Local nulls can be loosely defined as items that are not available locally, but may be available from the global database. During periods of good communication nulls can be handled by passing a request to the global database. In cases where the global database is not accessible it would be misleading to return a ‘global’ null value.

This paper discusses an extension to the current notion of null values to include the semantics of nulls found in mobile and distributed systems. The paper introduces local nulls in terms of amendments to the relational algebra and examines its impact on the query languages such as SQL. For the purposes of this paper, local and global null values will be represented as $\varphi$ and $\omega$, respectively.

This paper is structured as follows. Section 2 introduces the notation that will be used within the paper. Section 3 examines the relational algebra in terms of local nulls. Section 4 discusses the changes to query languages such as SQL to include local nulls. Finally, Section 5 will provide a conclusion and a discussion of the use of local nulls.

2 Notation

Maier examined the presence of unknown values in relational databases (Maier 1983). In this section we adopt these conventions as a basis for our modifications to include the notion of local null values. We

---

*This research was partially supported by Australian Research Council SPIRT Grant C00107533, held in collaboration with HP Labs.

---

1This is not an uncommon occurrence with mobile devices. Network inaccessibility may be caused by disconnection, low priority in partial or weak connection mode or intentional disconnection through power management because of limited battery capacity.

2 In this paper we term null values that are (authoritatively) held in the global database as ‘global nulls’ and while ‘local’ null values are non-authoritative values held on distributed and/or mobile devices.
discuss these amended notations and provides a comparison to Maier’s work.

A tuple in a local relation \( t \) is a locally complete tuple (or an LC-tuple) if it holds all data that also exists in the equivalent attributes in the global database. Conversely, a locally partial tuple (or an LP-tuple) contains one or more local nulls. LC-tuples will not contain any local nulls but may, of course, contain global nulls. LP-tuples may contain global and local nulls. The designation of LC and LP-tuples may be extended to include the notation in (Maier 1983) in which a tuple may be designated as a partial tuple or a total tuple depending on whether global nulls exist in the tuple. When an attribute value, \( t(A) \), is not a local null it is then considered to be identical to that in the global database, \( t(A)^L \downarrow \), where \( t(A) \downarrow \) defines an attribute value that is not a global null. Thus, for a set of attributes \( X \), \( t(X)^L \downarrow \) implies \( t(A)^L \downarrow \) for every attribute \( A \in X \). A simplified notation, \( t^L \downarrow \), is then used to define \( t \) as a locally-complete tuple, while a complete tuple consists of no nulls of either type. Similarly, for tuples, \( t \) and \( u \) defined over the same schema, \( t \) locally subsumes \( u \), \( t^L \geq u^L \) if \( \forall u(A)^L \downarrow, u(A) = t(A) \). If \( t^L \geq u^L \) and \( t^L \downarrow \), then \( t \) is a local extension of \( u \), \( t \geq^L u \). For example, the tuple \( < a, \varphi, \varphi > \) is locally subsumed by \( < a, b, \omega > \), and in this case, the tuple is also locally extended by \( < a, b, \omega > \). In comparison to Maier’s work (Maier 1983), tuple \( < a, b, \omega > \) is subsumed and extended by \( < a, b, c > \).

A relation \( r \) is a local complete relation (an LC-relation), \( r^L \downarrow \), when all its tuples are local complete tuples, and a local partial relation (an LP-relation) when its tuples contains one or more local nulls. For a relation scheme \( R \), \( \text{Rel} \uparrow (R)^L \) is a set of all locally partial relations over \( R \), while \( \text{Rel}(R)^L \) is the set of all locally complete relations over \( R \). For relations \( r \) and \( s \) over \( R \), \( r \) locally subsumes \( s \), denoted \( r \geq^L s \), if \( \exists t_s \in s, \forall t_r \in r : t_r \geq^L t_s \). If \( r \) is a local complete relation, then \( r \) is a local extension of \( s \) if every tuple of \( s \) is subsumed by at least one tuple of \( r \), denoted \( r \geq^L s \), and it is a local completion of \( s \) if every tuple of \( s \) is subsumed by exactly one tuple of \( r \), denoted \( r \geq^L s \). For example in Table 1, \( r \) is a local extension of \( s \) since both tuple \( < d, e, f > \) and \( < d, e, n > \) [subsume \( < d, e, \varphi > \), while \( p \) is a local completion of \( s \).]

<table>
<thead>
<tr>
<th>( r ) ( (A \ B \ C) )</th>
<th>( s ) ( (A \ B \ C) )</th>
<th>( p ) ( (A \ B \ C) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a \ b \ c)</td>
<td>(a \ b \ c)</td>
<td>(a \ b \ c)</td>
</tr>
<tr>
<td>&lt;d \ j \ c&gt;</td>
<td>&lt;d \ j \ c&gt;</td>
<td>&lt;d \ j \ c&gt;</td>
</tr>
<tr>
<td>\varphi \ l \ m</td>
<td>\varphi \ l \ m</td>
<td>\varphi \ l \ m</td>
</tr>
<tr>
<td>g \ \varphi \ i</td>
<td>g \ \varphi \ i</td>
<td>g \ \varphi \ i</td>
</tr>
</tbody>
</table>

Table 1: Example of Local Extension and Local Completion

3 Note that the local and global schemata may differ. For example, the local database may possess a restricted subset of attributes.

31 Set Theory Operations

Constraints on global nulls are usually such that they do not appear in any component of a candidate key (Maier 1983). Similarly, any component of a candidate key for any summary databases should not contain any local nulls. This is important as primary keys are used as indexes to retrieve information when it is lacking from the summary database.

3 Local Nulls and Relational Algebra

The relational algebra provides a set of operations to be used in the manipulation and retrieval of data from a database (Codd 1970). As with global nulls, the presence of local nulls in relations requires an extension to the current relational algebra.

31 Set Theory Operations

Two global relations, \( R(A_1, A_2, ..., A_n) \) and \( S(B_1, B_2, ..., B_m) \), are considered union compatible if they have the same degree of \( n \) and if \( \forall i \in n : dom(A_i) = dom(B_i) \). Given this, it is possible for set operations to be applied to these relations. For locally partial relations where local nulls exist, we need to show that the use of set operators are plausible and that the results gracefully degrade as local nulls are resolved. Since local nulls are extensions of the traditional null concept, it is reasonable to conclude that set theory operations should be possible for locally partial relations.

The conventional set theory operations of union \( \cup \), intersection \( \cap \) and set difference \( - \) operate over two union compatible tables \( r \) and \( s \) (see Table 2):

Table: Union Compatible Tables

<table>
<thead>
<tr>
<th>( r ) ( (A \ B \ C) )</th>
<th>( s ) ( (A \ B \ C) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a \ \varphi \ \varphi)</td>
<td>(a \ \varphi \ \varphi)</td>
</tr>
<tr>
<td>\varphi \ j \ \varphi</td>
<td>\varphi \ j \ \varphi</td>
</tr>
<tr>
<td>g \ h \ i</td>
<td>g \ h \ i</td>
</tr>
</tbody>
</table>

Table 2: Union Compatible Tables

For these three operations, only duplicate tuples will be removed from the final relation. However, for relations with global nulls, Codd introduced the null substitution principle to reduce redundancy in the relation (Codd 1979). This was further discussed in (Zaniolo 1984, Biskup 1983, Maier 1983) for global nulls. For locally partial relations, this principle is similarly applicable to remove redundancy with respect to local nulls instead of global nulls. That is, if for two tuples \( t \) and \( u \), \( t \geq^L u \), then \( u \) will be removed from the relation (see Table 3 for examples).

In related work, Rice and Roddick discuss a similar technique of reducing redundancy where compatible tuples are compared such that only the more informative tuple is kept (Rice & Roddick 2000). In their technique, however, tuples are not required to be locally or globally partial. In cases where the unique primary keys of a relation is available, it is possible to
Table 3: Operations using null substitution principle

Table 4: Keyed Set Operations

Table 5: Value Evaluation of $x \cap_k y$ and $x \cup_k y$

3.2 Select and Project Operations

In general, the select operation selects a tuple or a group of tuples that satisfies a certain condition, as denoted by $\sigma_{<\text{condition}>}(R)$ (Elmasri & Navathe 2000). With locally partial relations, the same form may be used, i.e. $\sigma_{<\text{condition}>}(R)$ where $R$ is a locally partial relation.

A selection over Table 6 $\sigma_{A=B}$ would result in rows 1 and 2 being selected, while rows 3-5 are not. To include row 3 into the result, a new operator ‘??’ is introduced. The operator ‘??’ states that, for two attribute values A and B,

$$A ?= B \iff (A = \varnothing) \lor (B = \varnothing)$$ (10)

Moreover, a new equality operator (‘=’) may now be introduced to include both answers that are true.
Projection operations are done over particular columns. That is, a projection over certain attributes will produce a new relation which has only those attributes. Since this does not include an evaluation of local nulls against a value, it is possible to include locally unknown solutions. That is, a projection over certain attributes will produce a new relation which has only those attributes that may be true, with respect to the conditions of the join. For example, Table 9 shows two relations that may be used for joining. Assuming a conventional join over the attributes $A$ and $F$, (ie. $R \bowtie_{A=F} S$), the relation in Table 10 would be produced. Methods to generalise joins over attributes where global nulls exists had been discussed in the literature (Zaniolo 1984, Codd 1975, Lacroix & Pirotte 1976, Maier 1983). These generalised join states that there may also exist tuples where it is globally unknown whether it is true, in addition to tuples that are true. That is, using the null substitution principle suggested in (Codd 1975), it is possible to include tuples where the equivalence evaluation over attribute values that are globally null indicates a globally unknown solution. These tuples are included since they may suggest that the solution may be true. For example, Table 11 shows one of the proposed generalised join, denoted $\bowtie_{G}$, over the attributes $A$ and $F$, ie.

\[
R \bowtie_{A=F} S
\]

For this example, tuples 1 to 5 are global unknown solutions. In fact, $R \bowtie_{A=F} S$ may be considered equivalent to a conventional join where $R \bowtie (A=F) \cup (A=\varphi) \cup (F=\varphi) \ S$. The local join over the

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 10: Conventional Join

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 11: Generalised Join

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 10: Conventional Join

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 11: Generalised Join

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 10: Conventional Join

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 11: Generalised Join

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 10: Conventional Join

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 11: Generalised Join

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 10: Conventional Join

\[
\begin{array}{ccccccccccc}
(A) & (B) & (C) & (D) & (E) & (F) & (G) & (H) \\
\hline
a & b & c & d & l & a & d & k \\
\end{array}
\]

Table 11: Generalised Join
This may be seen in Table 12. Similarly, a generalised local join is also proposed, denoted $\Join^{GL}$. That is,

$$ R \Join^{GL} (A=F) \cup (A=\varnothing) \cup (F=\varnothing) \cup (A=\varnothing) \cup (F=\varnothing) S $$ (15)

$$ R \Join^{LLO} = R \Join^{(A=F) \cup (A=\varnothing)} S $$ (16)

$$ R \Join^{RLO} = R \Join^{(A=F) \cup (F=\varnothing)} S $$ (17)

A summary of join operations, including both conventional and the proposed local operations, and their corresponding conditions over the attributes $A, B$ is shown in Table 13. A procedure for the division operation may be found in (Elmasri & Navathe 2000). Space precludes a discussion of division as it applies to local nulls in this paper but please see the associated technical report.

### 3.4 Division Operation

A procedure for the division operation may be found in (Elmasri & Navathe 2000). Here, an example is provided to show the results of dividing two tables, $R \div S$ to produce $T$ (Table 14).

$$ R \ (A \ B) \ S \ (A) \ T \ (B) $$

Table 14: Division Example

However, when undertaking division on tables where local nulls are involved, maybe true evaluations would also be included. For example, $R \div Q$ would produce a result, $T$, which maybe true since there is a possibility that the local null value is the value $b$ (Table 15).

Table 15: Local Null Division Example

### 4 Local Nulls and SQL

To take advantage of the notion of locality using local nulls, extensions to the query language is now required - we give some examples in SQL. For example, it is now possible to extend SQL such that we are now able to query for tuples that are locally unknown.

**SELECT**

```sql
FROM R
WHERE A <> 'b'
```

It is also possible to allow queries that results in tuples that are true and locally unknown. That is, using the new operator for relational algebra, '==', a new query statement such as

**SELECT**

```sql
FROM R
WHERE A == 'b'
```

may be constructed. For local joins,

$$ R \Join^{L} S \equiv R \Join^{(A==B)} S \equiv \sigma(A==B)(R \times S) $$ (18)

In which case, we may now convert it to

**SELECT**

```sql
FROM R, S
WHERE A == B
```

Additionally, it is also possible to convert the '==' operator further.

$$ R \Join^{L} S \equiv R \Join^{(A=B) \cup (A=\varnothing) \cup (B=\varnothing)} S \equiv \sigma(A=B)(R \times S) $$ (19)

which in SQL is equivalent to

**SELECT**

```sql
FROM R, S
WHERE A == B
```

This now provides a way to separate local unknown solutions from true solutions. Similarly, for generalised local joins,

$$ R \Join^{GL} S \equiv R \Join^{(A=B) \cup (A=\varnothing) \cup (B=\varnothing)} S \equiv \sigma(A=B)(R \times S) $$ (20)

which in SQL is equivalent to

**SELECT**

```sql
FROM R, S
WHERE A == B
```

For the other local operations, a conversion using the equivalent condition algebra as shown in Table 13 is possible and Table 16 provides a summary.
### Table 13: Summary of Local Join Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Join</td>
<td>$A = B$</td>
</tr>
<tr>
<td>Local Join</td>
<td>$A == B$</td>
</tr>
<tr>
<td>Left Outer Join</td>
<td>$(A = B) \cup (A = \omega)$</td>
</tr>
<tr>
<td>Left Local Outer Join</td>
<td>$(A = B) \cup (A = \omega) \cup (A = \varphi)$</td>
</tr>
<tr>
<td>Right Outer Join</td>
<td>$(A = B) \cup (B = \omega)$</td>
</tr>
<tr>
<td>Right Local Outer Join</td>
<td>$(A = B) \cup (B = \omega) \cup (B = \varphi)$</td>
</tr>
<tr>
<td>Generalised Join</td>
<td>$(A == B) \cup (A = \omega) \cup (B = \omega)$</td>
</tr>
<tr>
<td>Generalised Local Join</td>
<td>$(A == B) \cup (A = \omega) \cup (B = \omega)$</td>
</tr>
</tbody>
</table>

### Table 16: Summary of Local Join Operations in SQL

<table>
<thead>
<tr>
<th>Operations</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Join</td>
<td>SELECT * FROM R,S WHERE A==B</td>
</tr>
<tr>
<td>Left Local Outer Join</td>
<td>SELECT * FROM R LEFT LOCAL JOIN S WHERE A=B</td>
</tr>
<tr>
<td>Right Local Outer Join</td>
<td>SELECT * FROM R RIGHT LOCAL JOIN S WHERE A=B</td>
</tr>
<tr>
<td>Generalised Local Join</td>
<td>SELECT * FROM R,S WHERE A=B OR A=\varphi OR B=\varphi OR A=\omega OR B=\omega</td>
</tr>
</tbody>
</table>

5 Conclusion and Discussion

In this paper, we introduced the concept of local nulls in relations. We showed that it is possible to manipulate relations with these nulls through the use of the relational algebra with some modification. It is then possible to extend the SQL language to identify locally unknown solutions and, importantly, to provide users with a definable response to queries that include them.

The concept of local nulls is mainly applicable within a distributed or mobile database system, where a particular piece of information may exist in multiple databases within that system. Because local nulls are considered to be temporary nulls that replace actual values within a database in order to conserve storage capacity, it is possible for attribute values to be nulls, locally in a database, while other databases may have its actual value. In mobile databases, where the storage capacity may be limited, it is useful to have the ability to store only parts of the database that are used often. For values that are not stored, local nulls may be used to allow a database to determine whether the information being sought is to be found in other databases within its system.

In other work, we have enhanced the system discussed in (Chan & Roddick 2003), however, it has become clear that a gradation of null value, perhaps using hierarchies of values or ranges, may also be useful.

References


Codd, E. (1987), ‘More commentary on missing information in relational databases (applicable and inapplicable information)’, *SIGMOD Record* 16(1).

Date, C. (1986), Null values in database management, in ‘Relational Database: Selected Writings’, Addison-Wesley, Reading, MA.


