Integrated storage and querying of spatially varying data quality information in a relational spatial database

M.G.S.M Zaffar Sadiq and Matt Duckham
Department of Geomatics, University of Melbourne, Australia

Accepted for Transactions in GIS, November 19 2008

Short title RSVQ: Relational DBMS for Spatial Variation in Quality

Keywords Spatial data quality; spatial database; relational database management system; relational algebra; SQL; sub-feature variation

ABSTRACT
This paper investigates a new, integrated technique for storing and retrieving spatially varying data quality information in a relational spatial database. Rather than storing global data quality statements, the system enables data quality information to be referenced to a spatial framework, individual spatial objects, or even parts of spatial objects. The integrated model, called as RDBMS for Spatial Variation in Quality (RSVQ), allows flexible storage of spatially varying data quality information, and seamless querying irrespective of the underlying storage model. RSVQ is founded on a formal model of relational databases, defining a new derived, polymorphic query operator $\lhd$ to join quality data with spatial data. The $\lhd$ operator is implemented in an extension to SQL as a new WITHQUALITY keyword. A performance evaluation of RSVQ was conducted, using an Oracle Spatial database and a case study of cadastral data for parts of Victoria, Australia. The results of this evaluation demonstrated that the system is practical and efficient for a wide range of queries, as well as indicating the performance trade-offs associated with the different data quality storage models. Using the integrated RSVQ approach provides the potential for a single, consistent, database engine for a wide range of existing and proposed spatial data quality management systems.

1 INTRODUCTION

Uncertainty has long been recognized as an endemic feature of geographic information (Chrisman 1984; Blakemore 1984; Fisher 1989; Goodchild 1995). As a consequence, managing spatial data quality is acknowledged as a "critical component" of spatial data handling (Chrisman 1991). Over recent years, a range of ingenious techniques and systems have been proposed that enable data quality information to be referenced to geographic space, individual geographic objects, or even parts of objects. However, these systems typically adopt just one of these referencing systems. As a result, they do not allow users the flexibility to choose the most appropriate referencing system for their particular application, convert between different referencing systems, or easily query data quality stored against different reference systems.

This paper proposes and tests an *integrated* approach to efficiently storing and querying spatial data quality information, irrespective of the different representations of spatial data quality. The key problems addressed by this approach are the definition of:

(i) a flexible model for storage of spatial data quality information that can be referenced to geographic space, individual geographic objects, or even parts of objects; and
(ii) an integrated query mechanism, allowing users or applications efficiently to access quality information without knowledge of the underlying storage models.

The ultimate objective of this work is to provide the framework of a spatial data quality management "engine," capable of supporting the existing range of higher-level tools and techniques for using spatial data quality. In keeping with this goal, our approach is deliberately founded on conventional and widely accessible relational database theory and technology.

Section 2 reviews the existing systems in the literature for spatial data quality management, highlighting three fundamentally different representations of spatial data quality. Section 3 presents the integrated model for storing and querying data quality, building on established formalisms for relational database design and operations. Section 4 then describes the implementation of the model, including the definition of a simple extension to standard SQL for querying spatial data quality. To evaluate the performance of the implementation, especially with respect to storage and query efficiency, a case study of spatially varying data quality information in a cadastral application is presented in section 5. Finally, section 6 concludes the paper with a discussion of the key outcomes of this research as well as avenues for future work.

2 BACKGROUND

Much of today's research in spatial data quality is founded on the elements of spatial data quality, initially established in work by the NCDCDS in 1988 (National Committee for Digital Cartographic Data Standards 1988) and explored further in publications such as Guptill and Morrison (1995), Aalders (2002). The familiar elements of spatial data quality (SDQ), including positional and attribute accuracy, lineage, logical consistency, and completeness, have been adapted and extended, but form the basis of many national and international spatial data quality standards (see Moellering 1997).

Irrespective of *what* elements of data quality are of interest, the representations of spatial data quality used in existing spatial data quality management systems can be classified into three distinct categories, termed here *per-feature*, *feature-independent*, and *feature-hybrid* quality (Sadiq and Duckham 2007).

2.1 Per-feature quality

A range of approaches to representing spatial data quality have taken as their starting point a *per-feature* representation of quality, where each geographic feature stored in the database can be associated with quality information related to that feature. A "feature" can be defined as the digital object associated with a geographical entity (Usery 1996), such as the representation of a land parcel in a cadastral GIS. In the context of relational databases, it is easy to think of a feature as a record in a table relating to a specific geographic entity.

Research that has adopted the per-feature model includes Gan and Shi (2002), who developed an Error Metadata Management System (EMMS) to manage spatial data quality throughout the entire lifecycle of spatial data. Qiu and Hunter (2002) developed a model for storing data quality at different levels within an object-oriented hierarchy of features (e.g., for aggregate features or their individual components). Adapting established OLAP tools from databases, Devillers et al. (2005) developed sophisticated tools for management of multidimensional spatial data quality information. All these methods are built on the individual geographic feature as an atomic unit for data quality.

While the per-feature approach is conceptually and computationally efficient, it does not allow *sub-feature* variation to be stored. In many cases, spatial data quality may vary not just between different features, but within the geographic bounds of a single feature (Wong and Wu 1996). For example, a cadastral land parcel may have different positional accuracy information associated with the different vertices (sub-features) which make up that parcel. This information cannot be represented using the per-feature model.

2.2 Feature-independent quality

A contrasting approach to the per-feature model of data quality is to store spatial data quality information as a separate "layer" in the database, independently of the spatial data itself. Cartographers and geologists have stored and depicted quality information as separate *reliability* diagrams even before the advent of spatial databases (see Fisher 1991). Similarly, the National Committee for Digital Cartographic Data Standards (1988) proposed that the variation in positional accuracy could be reported through a quality overlay (reliability diagram). More recently, Heuvelink (1996, 1998) developed an influential field-based model of spatial data quality to store spatial data quality as separate spatial data layers, as well as techniques for propagating error through different spatial analyses.

A feature-independent approach allows spatial variation in quality to be stored independently of geographic features in the database. This approach has the clear advantage over per-feature quality that sub-feature variation in data quality can be represented. However, in the context of spatial databases, an important disadvantage of the feature-independent approach is that quality information is only *implicitly* associated with geographic features via their spatial location. In order to subsequently associate quality information with any particular feature, a *spatial join* is needed. Spatial joins are amongst the most computationally expensive spatial database operations. Consequently, the feature-independent model is expected to be much less efficient than the per-feature model when querying spatial databases for information about particular features' data quality.

2.3 Feature-hybrid quality

Combining the per-feature and feature-independent approaches to spatial data quality, some authors have investigated storing sub-feature variation in data quality, while explicitly associating that quality information with the particular feature(s) to which it refers (termed here *feature-hybrid* quality). For example, *measurement-based GIS*, introduced by Buyong et al. (1991) and extended by Goodchild (2002), Navratil et al. (2004), store quality information (specifically accuracy and precision) down to the level of individual survey observations. This information can then be propagated back to the complex vector geometries of spatial features via standard survey adjustment procedures. Similarly to Qiu and Hunter (2002), Duckham (2001) takes advantage of the hierarchical structure of object-oriented database to store data quality information at multiple levels. Unlike Qiu and Hunter (2002), Duckham (2001) proposed storing data quality down to sub-feature levels, potentially associating quality information with a feature's component line segments or vertices.

Feature hybrid models of data quality do allow sub-feature variation to be represented, at the same time as being linked to individual geographic features in the database. Like feature-independent approaches, feature-hybrid quality presents clear advantages over per-feature quality because sub-feature spatial variation can be represented. Unlike feature-independent quality, feature-hybrid quality has the advantage that it does not require expensive spatial joins to associate data quality information with individual features. However, this advantage comes at a cost. Linking stored data to both its location and its referring feature is more likely to lead to redundancy in the data, for example where the different features share parts of geometries (like vertices or edges at shared boundaries). In turn, such redundancy makes consistency harder to enforce (for example where a geographic feature is updated, the associated quality information must also be updated to ensure consistency is maintained).

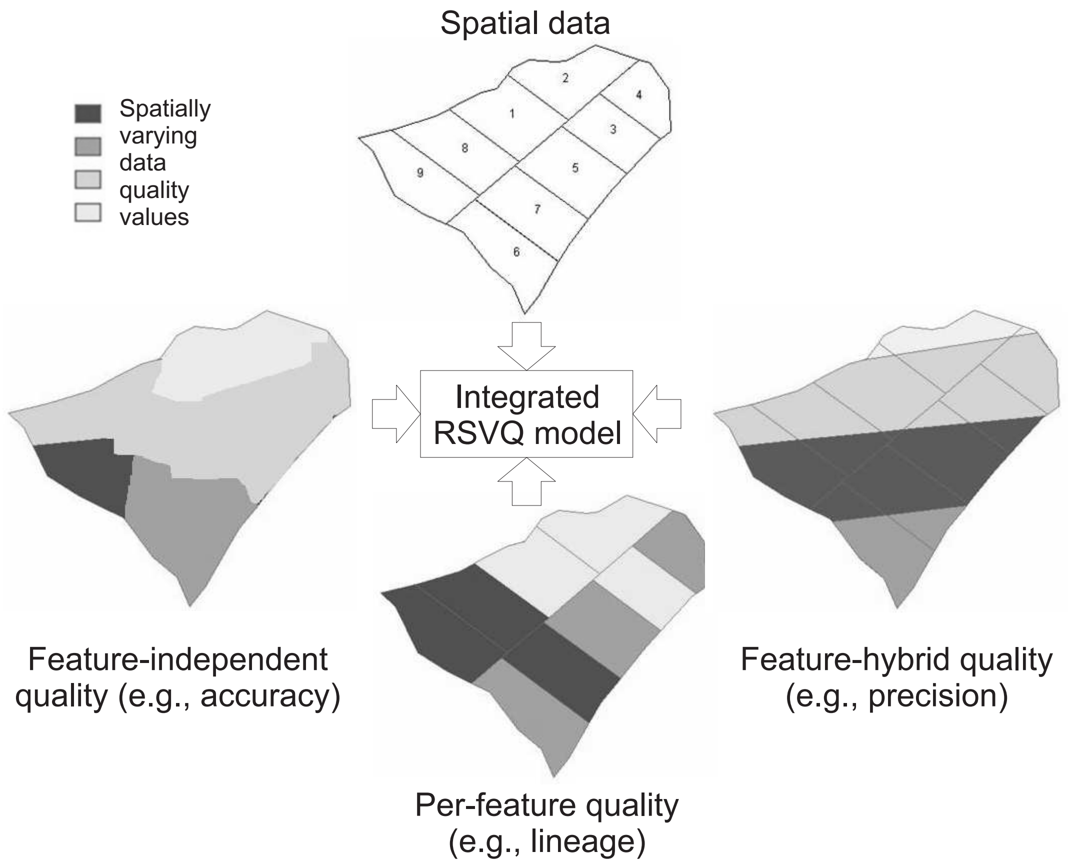
2.4 Abstractive and representative quality

The three classes of spatial data quality discussed above cover a substantial block of research into spatial data quality elements, including positional, attribute, or semantic accuracy, precision, lineage, bias, timeliness and currency. However, there do exist some types of spatial data quality that may not be adequately modeled by these three categories. For example, logical consistency can be represented not as data to associate with (sub-) features, but rather as a *process* of assessment (Kainz 1995). While the outcome of a logical consistency test could be stored using any of the models above (e.g., associate the results of a logically consistency test with a feature or sub-feature), the test itself cannot be represented simply as data, and requires a rather different, process-oriented type of representation.

In a similar vein, Duckham (2001) makes a distinction between *abstractive* and *representative* quality elements. Abstractive quality concerns imperfections in spatial data connected with the classification and definition of geographic features, like tests for logical consistency or completeness. Representative quality concerns metadata connected with particular instances of features, like positional accuracy or lineage. From this viewpoint, the three categories of spatial data quality are all examples of representative quality, not abstractive quality. Accordingly, while the approach adopted in this research claims to be able to support any representative quality element, it can not claim to enable storage of the procedures for testing abstractive quality directly (only the results of any particular test). This limitation can potentially be addressed through a number of further ways, and is discussed in the conclusions as one of the directions for further work.

2.5 Summary

Numerous examples of the different models of (representative) spatial data quality (per-feature, feature-independent, and feature-hybrid) can be found in the literature. However, no single model is *a priori* superior to the others, each having their relative advantages and disadvantages: per-feature is expected to be most efficient in terms of storage space and querying speed, but only where sub-feature variation in data quality is not an issue; feature-independent is simple and efficient to store, but potentially computationally expensive to query; feature-hybrid is efficient to query, but more complex with associated database redundancy and consistency problems. The challenge, then, is to design an integrated spatial database that allows the storage and querying of each of these models spatial data quality, including sub-feature variation in quality, irrespective of which underlying type of quality is being used. The following section provides an abstract description of a data model, RSVQ, that is designed to achieve exactly this. However, the model is limited in scope to elements of data quality that can be represented as data items. It does not support representations of some elements of data quality, like logical consistency, that are typically represented as *assessment procedures*, rather than data items.


Spatial data
Spatially varying data quality values
Integrated RSVQ model
Feature-independent quality (e.g., accuracy)
Feature-hybrid quality (e.g., precision)
Per-feature quality (e.g., lineage)

[htb]

Figure 1. Integrated approach to spatial data quality storage in RSVQ

3 RSVQ MODEL

Building on standard relational database models, this section develops a formal model of spatial data quality storage in a relational spatial database. Using existing spatial relational database technology ensures that the resulting model is as generic as possible, potentially forming the database engine that can support any of the existing and proposed systems for spatial data quality management, reviewed in section 2.

The model is based on standard relational algebra, the foundation of every relational database management system (RDBMS), and is termed RSVQ (RDBMS for Spatial Variation in Quality). The key idea behind RSVQ is to integrate the three main representations of spatial data quality identified in section 2: per-feature (PF), feature-independent (FI) and feature-hybrid (FH) quality. Figure 1 illustrates the idea behind the approach, showing three different representations of accuracy associated with a small set of land parcels.

3.1 Database design

Relational databases are in essence collections of tables (relations). This section uses *relation schemes* to describe the general design of our RSVQ database. A relation scheme is a common structure in relational database design that specifies the name of the table (relation), followed in parentheses by the names and domains for any attributes in that table (e.g., tablename(attribute1: integer, attribute2: string)).

In general, features (spatial objects) in a relational spatial database are stored in a *feature relation* (layer) l using a relation scheme of the form:

$$\mathsf{l}(\underline{\mathsf{id}}: \mathsf{int}, \mathsf{geom}: \mathsf{GEOMETRY}, ...)$$

where id is an identifier for the geospatial objects and geom is the geometry of those objects (and "..." indicates that any number of non-spatial attributes may also be stored alongside a particular feature). $\mathcal{L}$ denotes the set of all feature relations in a database.

PF quality can be modeled using a *per-feature quality relation* qf with a relation scheme of the form:

$$\mathsf{qf}(\underline{\mathsf{oid}}: \mathsf{int}, ...)$$

[htb]

Behavior of $\mathsf{l} \lhd \mathsf{q}$	$\mathsf{q} \in \mathcal{Q}_f$	$\mathsf{q} \in \mathcal{Q}_g$	$\mathsf{q} \in \mathcal{Q}_h$
If $\langle \mathsf{q}, \mathsf{l} \rangle \in \mathsf{m}$	**B1**	n/a	**B1**
Otherwise	**B3**	**B2**	**B2**

Table 1. Summary of behaviors for polymorphic quality operator

where oid is a foreign key which relates quality records to objects in a specific feature relation. Each tuple in a per-feature quality relation stores information about a particular quality record along with the identifier of the object to which that quality record refers. $\mathcal{Q}_f$ denotes the set of all per-feature quality relations in a database.

FI quality can be modeled using a *feature independent quality relation* qg with a relation scheme of the form:

$$\mathsf{qg}(\underline{\mathsf{qid}}\text{: int, qgeom: GEOMETRY, ...})$$

where qid is a quality identifier, and qgeom is a geometry attribute that describes the spatial characteristics of each quality feature. The geometry may be any supported geometry type (e.g., points, lines, polygons). Each tuple in a feature independent quality relation stores information about a particular quality record $q \in Q$ along with the set of locations that map to that quality record. $\mathcal{Q}_g$ denotes the set of all feature independent quality relations.

Finally, FH quality can be modeled using a *feature hybrid quality relation* qh with a relation scheme of the form:

$$\mathsf{qh}(\underline{\mathsf{qid}}\text{: int, oid: int, qgeom: GEOMETRY, ...})$$

where qid is a quality identifier, oid is a foreign key which relates qualities to features in a specific feature relation, and qgeom is a geometry attribute that describes the spatial characteristics of each quality feature. Each tuple in a feature hybrid quality relation stores information about a particular quality record along with the set of locations that map to that quality record and the identifier of the object to which that quality record refers. $\mathcal{Q}_h$ denotes the set of all feature hybrid quality relations.

One further component is required for the database design. Quality information stored using either the per-feature or feature-hybrid model relates to a specific feature relation (with matching feature identifiers). Information about which feature relation "matches" with which per-feature or feature-hybrid quality relation must be stored in the RSVQ system. To achieve this, the RSVQ model uses a *metadata* table

$$\mathsf{m}(\underline{\mathsf{qtable}}\text{: string, ftable: string})$$

which identifies the unique feature relation (ftable) that each per-feature or feature-hybrid quality relation (qtable) that stores the quality of. Feature-independent quality relations are excluded from the metadata table because they do not refer to a specific feature relation (i.e., they are independent of features).

3.2 Quality operator

Based on the high-level database design above, a quality operator $\lhd$ can now be formalized as a relational algebra operator. Fundamental relational algebra operations include unary operators such as project (π, select a subset of attributes from a table), restrict (σ, select a subset of rows from a table), rename (ρ, rename attributes in a table); and binary operators such as union ($\cup$, combine all the rows from two tables), and join ($\bowtie$, form the product of two tables by pairing all rows subject to some join condition). For more information on relational algebra the reader is directed to any standard database textbook (e.g., Elmasri and Navathe 1994).

The relational quality operator $\lhd$ is defined as a derived binary relational taking two arguments: a feature relation $\mathsf{l} \in \mathcal{L}$ and a quality relation $\mathsf{q} \in \mathcal{Q}$, where $\mathcal{Q} = \mathcal{Q}_f \cup \mathcal{Q}_g \cup \mathcal{Q}_h$. The operator returns a single joined relation $\mathsf{l} \lhd \mathsf{q}$ where the qualities from relation q are associated with related features from l. Like any relational operator the result of applying the quality operator $\lhd$ to two tables is itself a relation. In fact, the relation $\mathsf{l} \lhd \mathsf{q}$ is a type of join having attributes $att(\mathsf{q}) \cup att(\mathsf{l})$ (where $att(\mathsf{r})$ is the set of attributes of a relation r).

In essence, the $\lhd$ operator annotates the geospatial objects in a feature relation l with associated quality information from a quality relation q; $\mathsf{l} \lhd \mathsf{q}$ can be read "annotate feature table l with quality from table q". To achieve this with any of the three different types of quality relations, PF, FH, and FI, the $\lhd$ operator is polymorphic, exhibiting different behaviors depending on the precise arguments supplied. The three different behaviors, B1, B2, and B3, are detailed below, and summarized in Table 1.

3.2.1 B1 behavior

The B1 behavior is triggered when the quality table q is PF or FH quality ($\mathsf{q} \in \mathcal{Q}_f \cup \mathcal{Q}_h$) and q contains quality that matches the supplied feature table l ($\langle \mathsf{q}, \mathsf{l} \rangle \in \mathsf{m}$). The B1 behavior of the $\lhd$ operator, $\mathsf{l} \lhd^{B1} \mathsf{q}$, implements a natural spatial join of the quality relation q and the feature relation l based on their shared object identifiers, as follows:

$$\mathsf{l} \lhd^{B1} \mathsf{q} \equiv \mathsf{l} \bowtie_{\mathsf{id}=\mathsf{oid}} \mathsf{q}$$

3.2.2 B2 behavior

The B2 behavior is triggered when the quality table q is FI or FH quality ($\mathsf{q} \in \mathcal{Q}_g \cup \mathcal{Q}_h$) and q contains quality that does match the supplied feature table l ($\langle \mathsf{q}, \mathsf{l} \rangle \notin \mathsf{m}$). The B2 behavior of the $\lhd$ operator, $\mathsf{l} \lhd^{B2} \mathsf{q}$, implements a spatial join of the quality relation q and the feature relation l based on their shared geometries, as follows:

$$\mathsf{l} \lhd^{B2} \mathsf{q} \equiv \mathsf{l} \bowtie_{\mathsf{geom} \cap \mathsf{qgeom} \neq \varnothing} \mathsf{q}$$

where the condition $\mathsf{geom} \cap \mathsf{qgeom} \neq \varnothing$ tests whether the geometries associated with objects and quality records spatially overlap (i.e., in this context $\cap$ is the spatial intersection of geometries rather than set-based intersection).

3.2.3 B3 behavior

The B3 behavior is triggered for the specific case where the quality table q is PF quality ($\mathsf{q} \in \mathcal{Q}_f$) but does not match the supplied feature table l ($\langle \mathsf{q}, \mathsf{l} \rangle \notin \mathsf{m}$). The B3 behavior of the $\lhd$ operator, $\mathsf{l} \lhd^{B3} \mathsf{q}$, implements a combined spatial and natural join of the quality relation q, the supplied feature relation l, and the feature relation l' that is matched with q (i.e., the unique l' such that $\langle \mathsf{q}, \mathsf{l'} \rangle \in \mathsf{m}$).

$$\mathsf{l} \lhd^{B3} \mathsf{q} \equiv \pi_{att(\mathsf{q}) \cup att(\mathsf{l})} (\rho_{\mathsf{tid/id,g/geom}} (\mathsf{l'} \bowtie_{\mathsf{id}=\mathsf{oid}} \mathsf{q}) \bowtie_{\mathsf{geom} \cap \mathsf{g} \neq \varnothing} \mathsf{l})$$

3.3 Example queries

To illustrate the use of the quality operator, consider the following relation scheme:

- Lin(qid, comment): per-feature quality relation storing lineage;
- Prec(qid, qgeom, precision): feature-independent quality relation storing coordinate precision information for polygon vertices;
- PosAcc(qid, oid, qgeom, acc): feature-hybrid quality relation storing positional accuracy information;
- Reg(id, geom, cat): a feature table of regions, where $\langle \mathsf{Lin}, \mathsf{Reg} \rangle \in \mathsf{m}$ and $\langle \mathsf{PosAcc}, \mathsf{Reg} \rangle \in \mathsf{m}$.

Based on this relation scheme, the following example queries illustrate the usage of the $\lhd$ operator in combination with some other standard relational algebra operators.

Query 1: What positional accuracy information is associated with region feature with id=5?

$$\sigma_{\mathsf{id}=5}(\mathsf{Reg} \lhd \mathsf{PosAcc})$$

In this case, the $\lhd$ operator will execute the B1 behavior, since PosAcc is FH quality and matched with Reg. The resulting table will contain the ID and geometry of feature number 5, along with the (possibly spatially varying) positional accuracy information for that feature.

Query 2: What is the precision associated with polygon vertices of parcels with area greater than 500m^2?

$$\pi_{\mathsf{precision}}(\sigma_{\mathsf{geom.area}>500}(\mathsf{Reg} \lhd \mathsf{Prec}))$$

Note that although the relational algebra expression for query 2 looks very similar to that in query 1, the actual implementation used to instantiate the $\lhd$ operator will be different (B2 behavior as opposed to B1 behavior in query 1, because Prec is of type FI).

Query 3: List all the lineage information relating to a rectangular window r (spatial range query).

Assuming a relation r(id: int, geom: int) which contains one tuple, $\langle 1, r\rangle$, the relational algebra expression below will retrieve the answer to this query (executing the B1 behavior as the relation Line is of type FH).

$$\pi_{\mathsf{comment}}(\mathsf{r} \lhd \mathsf{Lin})$$

This type of query is important as it illustrates that even though per-feature and feature-hybrid quality tables must have a single matched feature relation (specified in the metadata table) m, it is not a requirement of the RSVQ model that PF and FH quality tables can only be joined with their match; they may also be joined with unmatched tables, such as in a window query.

4 IMPLEMENTATION

Having specified in section 3 the formal relational algebra model for storing and querying spatially varying data quality information, the RSVQ system was implemented based on this model on top of an existing spatial RDBMS (Oracle Spatial 10g).

4.1 WITHQUALITY keyword

The relational algebra used as the basis for the design of the quality operator $\lhd$, is also the foundation of the standard database query language SQL (International Organization for Standardization 1999). As a result, a natural mechanism for implementing the quality operator $\lhd$ was to design a new keyword, called WITHQUALITY, to extend the standard SQL SELECT statement. As for any extension to SQL, the design of the WITHQUALITY keyword took care to ensure that the characteristic SELECT-FROM-WHERE structure and syntax of SQL was preserved (cf. Egenhofer 1994). The WITHQUALITY keyword is designed to be used with a feature table (layer) and a quality table, just like the quality operator $\lhd$. In addition, the WITHQUALITY keyword allows an alias name to enable other parts of a SQL SELECT statement to refer to the table resulting from annotating spatial features with data quality information. The general syntax for the quality query operator is:

$$\langle \textit{layer table name} \rangle\ \mathsf{WITHQUALITY}\ \langle \textit{quality table name} \rangle\ \langle \textit{alias name} \rangle$$

The semantics of the WITHQUALITY keyword derive directly to the underlying quality operator $\lhd$: l WITHQUALITY q x annotates the geospatial objects in a feature relation l with associated quality information from a quality relation q, renaming the resulting table x. In short, l WITHQUALITY q x can be interpreted "annotate table l with quality from q (renaming the result x)" (cf. section 3.2).

The WITHQUALITY keyword can occur in place of any ordinary table name in an SQL SELECT statement (i.e., after FROM keyword). More specifically, the most basic form of the SQL SELECT statement is:

$$\mathsf{SELECT} A\ \mathsf{FROM}\ T$$

where $T = t_1, t_2, ..., t_n$ is a comma separated list of one or more table names from the database, and A is a comma separated list of one or more attributes from tables in T (or a wildcard $*$ for all attributes from T). The WITHQUALITY operator extends this form by allowing $T = s_1, s_2, ..., s_n$, where each s_i is either the name of a table t in the database, or the names of two tables connected by the WITHQUALITY operator, t_1 WITHQUALITY t_2.

Just as conventional relational algebra statements can be mapped directly to SQL SELECT statements, so relational algebra statements that include the $\lhd$ operator can be mapped directly to SQL SELECT statements incorporating the extended WITHQUALITY keyword. For example, the three relational algebra expressions given in section 3.3 would be written as SQL queries as follows:

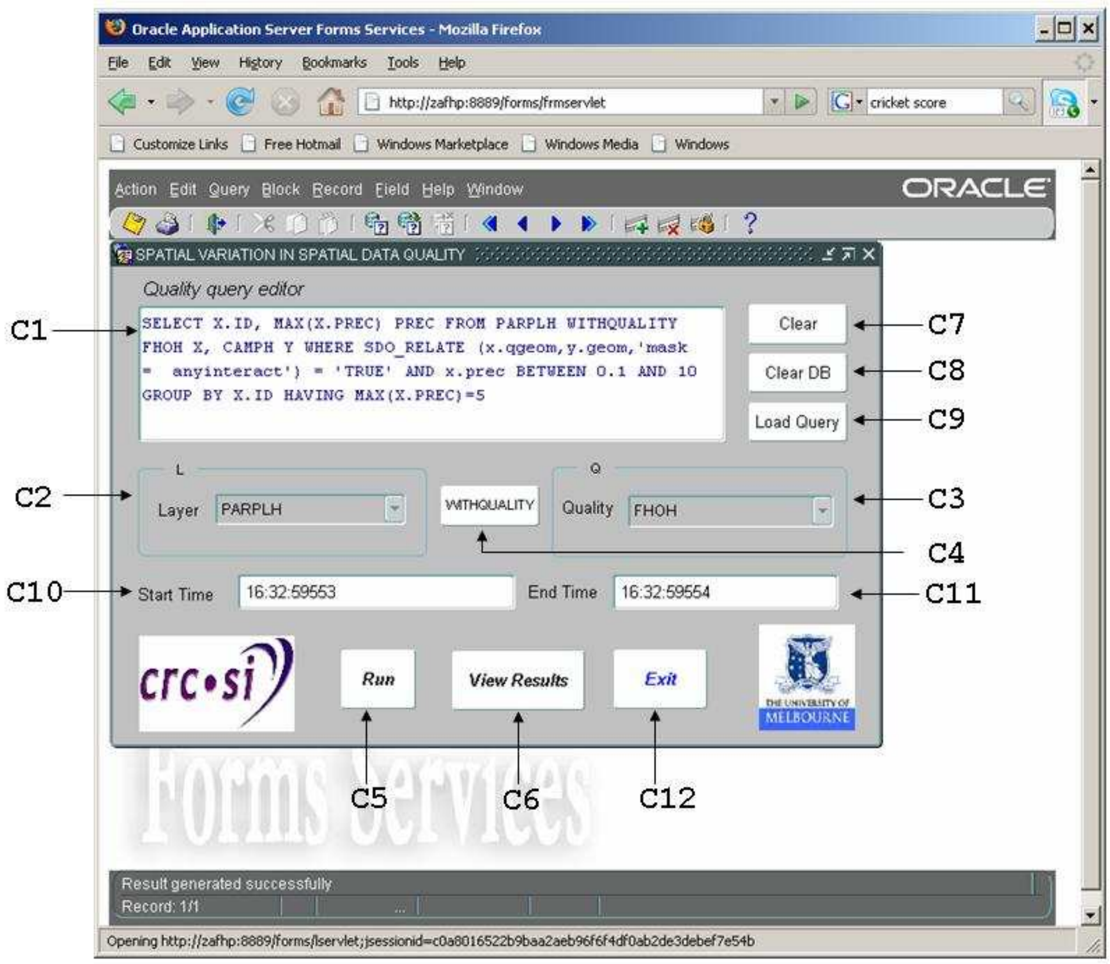
Query 1: What positional accuracy information is associated with region feature with id=5?

```
SELECT * FROM Reg WITHQUALITY PosAcc x WHERE id=5
```

Query 2: What is the precision associated with polygon vertices of parcels with area greater than 500m^2?

```
SELECT x.precision FROM Reg WITHQUALITY Prec x WHERE AREA(geom)>500
```

[htb]


Oracle Application Server Forms Services - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://zafhp:8889/forms/frmservlet
cricket score
Customize Links Free Hotmail Windows Marketplace Windows Media Windows
Action Edit Query Block Record Field Help Window
ORACLE
SPATIAL VARIATION IN SPATIAL DATA QUALITY
Quality query editor
Clear
Clear DB
Load Query
L
Layer
PARPLH
WITHQUALITY
Q
Quality
FHOH
Start Time
16:32:59553
End Time
16:32:59554
crc•si
Run
View Results
Exit
THE UNIVERSITY OF MELBOURNE
Forms Services
Result generated successfully
Record: 1/1
Opening http://zafhp:8889/forms/lservlet;jsessionid=c0a8016522b9baa2aeb96f6f4df0ab2de3debef7e54b
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12

```
SELECT X.ID, MAX(X.PREC) PREC FROM PARPLH WITHQUALITY
FHOH X, CAMPH Y WHERE SDO_RELATE (x.qgeom,y.geom,'mask
= anyinteract') = 'TRUE' AND x.prec BETWEEN 0.1 AND 10
GROUP BY X.ID HAVING MAX(X.PREC)=5
```

Figure 2. User interface implemented for RSVQ system

Query 3: List all the lineage information relating to a rectangular window r (in table r).

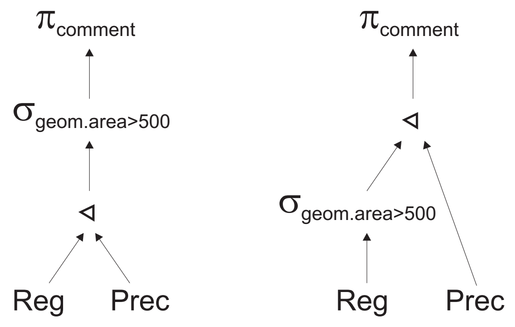
SELECT x.comment FROM r WITHQUALITY Lin x

4.2 Query interface

The WITHQUALITY operator was implemented using PL/SQL, Oracle's procedural language based on SQL Feuerstein (2002). Further, Oracle Forms was used to develop a simple user interface for accessing the WITHQUALITY keyword. The user interface shown in Figure 2, and includes the following controls:

C1: A plain-text query editor window;
C2–3: A combo box for selecting the feature table or quality table from the list of tables stored in the database;
C4: A button for inserting the WITHQUALITY keyword;
C5: A Run button to execute the query currently in the query editor;
C6: A View Results button to invokes external software to display the results of the query as a map (Geometry Spatial Console);
C7–8: Clear and Clear DB buttons clear the contents of the query editor and any temporary tables created by queries;
C9: A Load Query button to load previously saved queries;
C10–11: Debugging information about the time for query execution; and
C12: A button to close the form.

[htb]



a. Non-optimized b. Optimized

Figure 3. Query tree optimization for example query $\pi_{\mathsf{precision}}(\sigma_{\mathsf{geom.area}>500}(\mathsf{Reg} \lhd \mathsf{Prec}))$

4.3 Optimization

Conventional SQL statements can be optimized by decomposing the query into atomic relational algebra operations, and reorganizing these atomic operations to maximize efficiency. For example, joins ($\bowtie$) are amongst the most computationally expensive relational algebra operations. As a result, database optimization routines will usually ensure that when executing an SQL statement, any restrictions (σ) are performed before joining tables (since restrictions will typically reduce the number of tuples in the tables to be joined, decreasing the total computational cost of the combined operation). Similarly, because the WITHQUALITY keyword results in a type of join, the RSVQ implementation included optimization routines to ensure that any restrictions were performed before evaluating any WITHQUALITY keywords. Figure 3 illustrates the optimization process for the extended SQL query given as example query 2 above (SELECT x.precision FROM Reg WITHQUALITY Prec x WHERE AREA(geom)>500 or equivalently $\pi_{\mathsf{precision}}(\sigma_{\mathsf{geom.area}>500}(\mathsf{Reg} \lhd \mathsf{Prec}))$).

Potentially, such optimization functionality could be supported natively by Oracle (alongside existing similar query optimizations already built into Oracle). However, without access to commercial Oracle source code, our RSVQ implementation used customized PL/SQL code developed for this project to detect and perform these query optimizations.

5 CASE STUDY

The RSVQ model was tested in a realistic scenario using cadastral data from the Victorian Department of Sustainability and Environment (DSE, the Victorian Government department responsible for maintaining and updating spatial data for the state of Victoria, Australia). The key feature of the case study was to examine how scalable the RSVQ system is (i.e., how the efficiency of storage and querying in the RSVQ system changed as a function of the input dataset size).

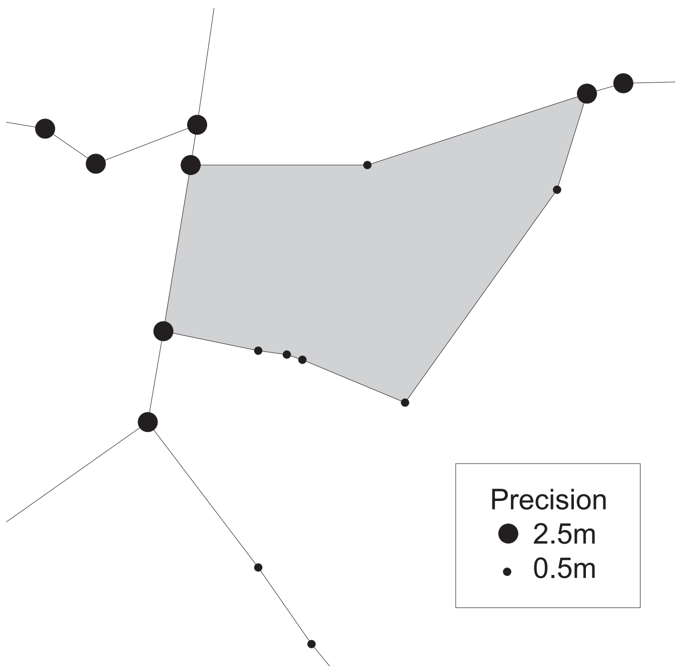
5.1 Data

The data for the case study was supplied by DSE and comprised cadastral land parcel data for the Hume local government area (LGA). This data is known to have coordinate precision that varies within individual land parcels (see Figure 4). Current spatial data quality storage mechanisms are not capable of storing such sub-feature variation. As a result, the actual positional accuracy of the data in some locations can be *higher* than the reported accuracy, since the *lowest* quality is often used where sub-feature variations cannot be be represented.

The case study data only contains information about coordinate precision, but not other data quality elements like lineage or attribute accuracy. This is a limitation of the available data, but not a limitation of the RSVQ model. As discussed in previous sections, RSVQ is capable of supporting a wide range of data quality information (specifically any "representative" data quality, see section 2.5).

The data was collated into four different sized data sets, in order to investigate the scalability of the RSVQ system. The small data set contained approximately 1000 records (Jacana suburb of Melbourne); the medium data set comprised $\sim$4000 records (Cambellfield suburb); the large data set comprised $\sim$13000 records (the western part of the Hume LGA, comprising Sunbury, Wildwood, and Bulla suburbs); with the very large data set including all data from Hume LGA ($\sim$75000 records).

The data quality information about spatially varying coordinate precision was also compiled into each of the three quality models, PF, FI, and FH, for each of the different data set sizes. It would normally be redundant to store the *same* quality information as different quality models in the same database. However, doing so in this case study provided a basis for more direct comparison of the RSVQ performance with the different quality models. The FI and FH quality relations for each of the four data sets contained exactly the same information, although structured somewhat differently (see section 3.1). However, the PF quality relations necessarily contained slightly different information, because the PF quality model cannot represent



[htb]

Figure 4. Example sub-feature variation in data quality within a land parcel geometry in Hume LGA, Victoria (data courtesy DSE)

sub-feature variation. Consequently, in the PF quality relations each parcel was associated with only one coordinate precision record, arbitrarily chosen to be the lowest (worst) precision of all recorded for that parcel.

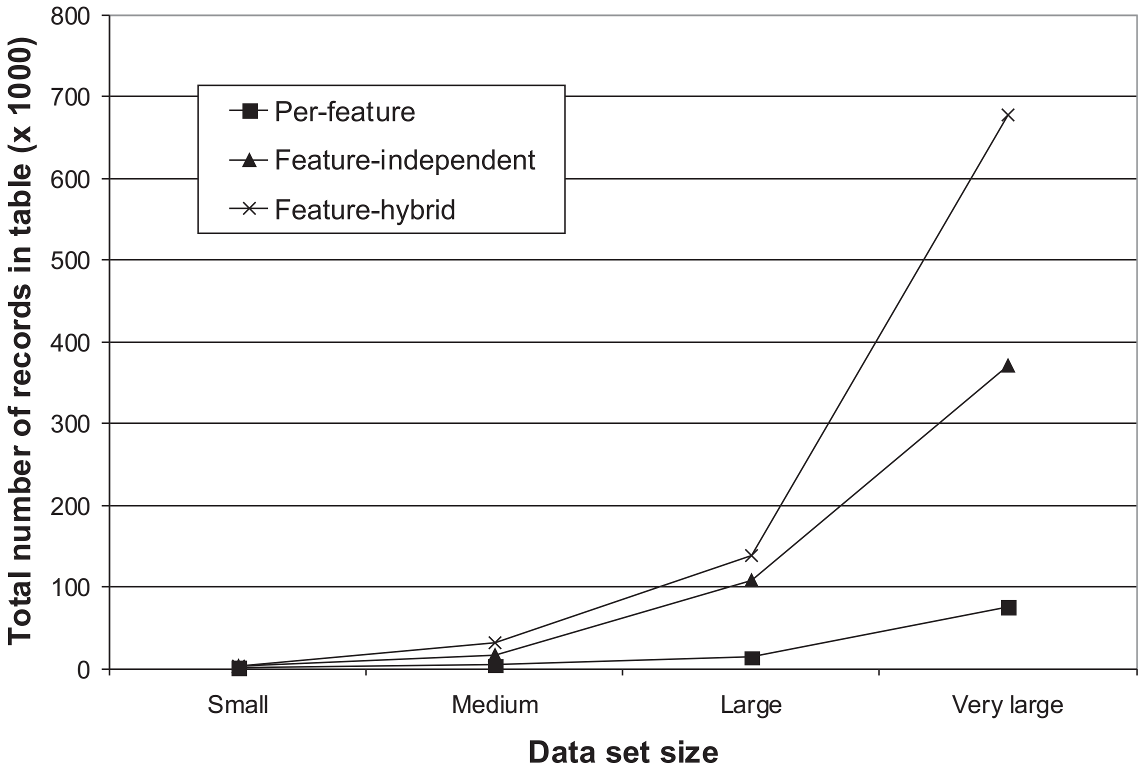
5.2 Storage performance

To investigate the comparative storage performance of the three different quality models, the total number of records was calculated for each of the quality relations and for each data set size. The results are summarized in Figure 5. As expected, per-feature quality relations require the least storage space, only one quality record for each geographic feature. Feature-independent quality relations require substantially more storage space, but are able to store information about sub-feature variation ignored by the per-feature model. Finally, feature-hybrid quality relations require the largest number of records. Although the feature-hybrid quality relations store exactly the same underlying data as the feature-independent quality relations, the feature-hybrid data structure introduces some redundancy into the quality relation, where some quality records appear more than once associated with different geographic features. A degree of redundancy in feature-hybrid quality is inevitable, especially if tables in the database are to remain normalized.

5.3 Query performance

Experiments across a wide range of different query types highlighted the potential inefficiency associated with querying the feature-independent model. For example Figure 6 shows the time taken to compute a response to a basic quality query SELECT * FROM I WITHQUALITY q x WHERE x.id $>$ 15000 AND x.prec $=$ 2.5 (where I is the appropriate feature relation, Small, Medium, Large, or Very Large, and q is the corresponding quality relation) for per-feature and feature-hybrid quality. Because data stored as PF and FH quality is linked with non spatial indexes to individual land parcels, it can be retrieved in a fraction of a second. More important than the absolute time taken to retrieve the answer to the query (which might vary depending on the precise hardware and software used) is the shape of the response curves. The curves exhibit gradual and linear increases in response times with increasing data set size. This indicates that queries on PF and FH quality are highly scalable, requiring little additional time to compute as data set size increases.

Figure 7 shows the response times for the same query, SELECT * FROM I WITHQUALITY q x WHERE x.id $>$ 15000 AND x.prec $=$ 2.5, for data stored as FI quality. This query can take more than 40 minutes in the case of the very large data set. More significantly, because of the spatial join required for queries to FI quality, the response curves exhibit much more rapid increases in computation time with data set size when compared with PF or FH quality. Optimization of the query does lead to a significant reduction in response time (significant at the 1% level for a population of five experiments using a t-test), from about 40 minutes to 20 minutes in the very large data set. However, while query optimization does result in improvements in absolute response times, the overall scalability of the optimized queries is the same, both curves exhibiting similar growth



[htb]

Figure 5. Storage efficiency of per-feature, feature-independent, and feature-hybrid quality

shapres. In other words, optimization can improve the absolute time taken to respond to a query, but has no effect on the overall scalability of the query, which is less efficient that the scalability of queries to PF and FH quality.

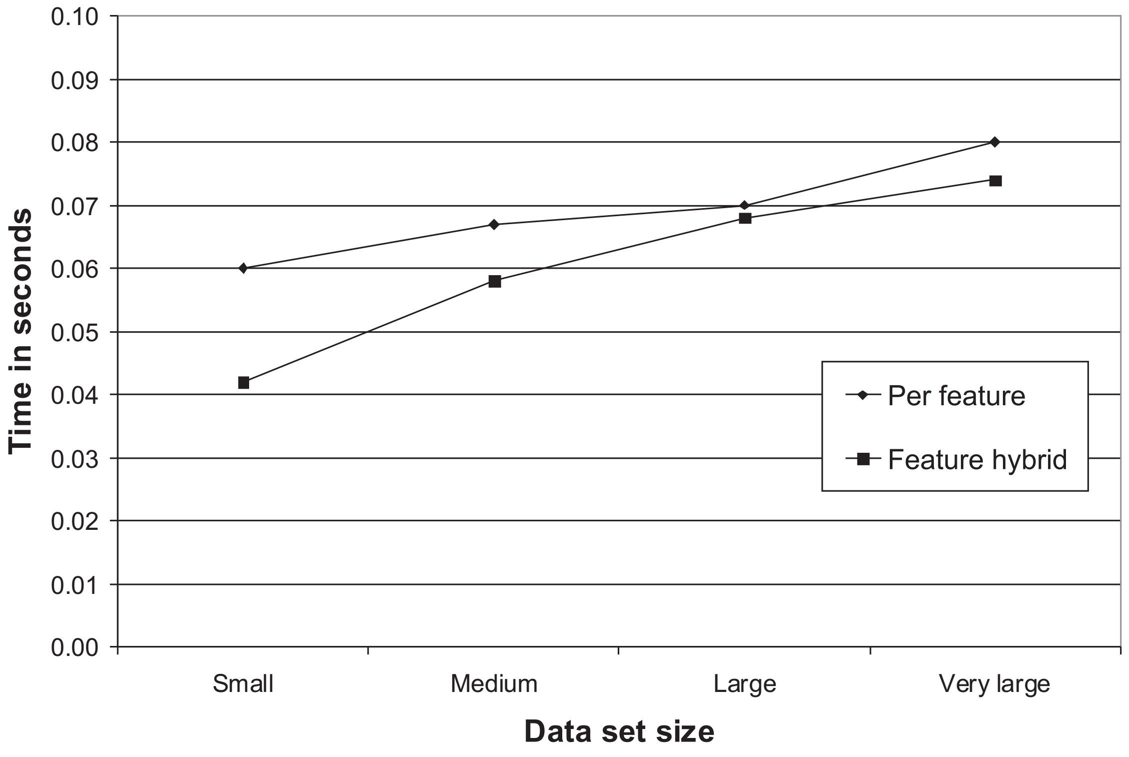
6 DISCUSSION AND CONCLUSIONS

This research has demonstrated that it is possible to develop an integrated spatial data quality storage system, on top of conventional relational database technology, that can support the range of different approaches to spatial data quality management found in the literature. The RSVQ system can allow a single, consistent, and standardized query interface, using SQL, to spatially varying data quality information referenced to individual geographic features, parts of features, or directly to a spatial framework independently of geographic features in the database.

The results in section 5 confirm and reinforce the expectation in section 2 that none of the different models is entirely superior to any other. The per-feature model is simple and efficient for both storing and querying information, but unable to represent variation in spatial data quality below the level of individual features (sub-feature variation). The feature-independent model is also simple, relatively efficient for storing quality information, and able to represent sub-feature variation, but computationally expensive to query. The feature-hybrid model is also able to represent sub-feature variation and computationally efficient to query, but relatively inefficient in terms of storage and inbuilt redundancy.

The balance of these different advantages and disadvantages can be related directly to which model would be most appropriate in different applications. In applications where no sub-feature variation exists, the per-feature model should be preferred. In applications where sub-feature variation does exist, where storage efficiency and data consistency is paramount, but query efficiency is less important, the feature-independent model is most suited. Conversely, in applications where sub-feature variation exists, but query efficiency is more important that storage efficiency, the feature-hybrid model should be preferred. The integrated RSVQ system enables all three types of quality information to be stored and queried together using a single consistent interface.

It is also possible to convert between different models. Indeed, the RSVQ system includes a module for converting between different data models (something that can be achieved relatively simply using SQL statements extended using the WITHQUALITY operator). Direct conversion between all models is possible, although conversion from FH or FI to PF may result in data loss, where any sub-feature variation in FH or FI data will be destroyed in translation into the PF quality



[htb]

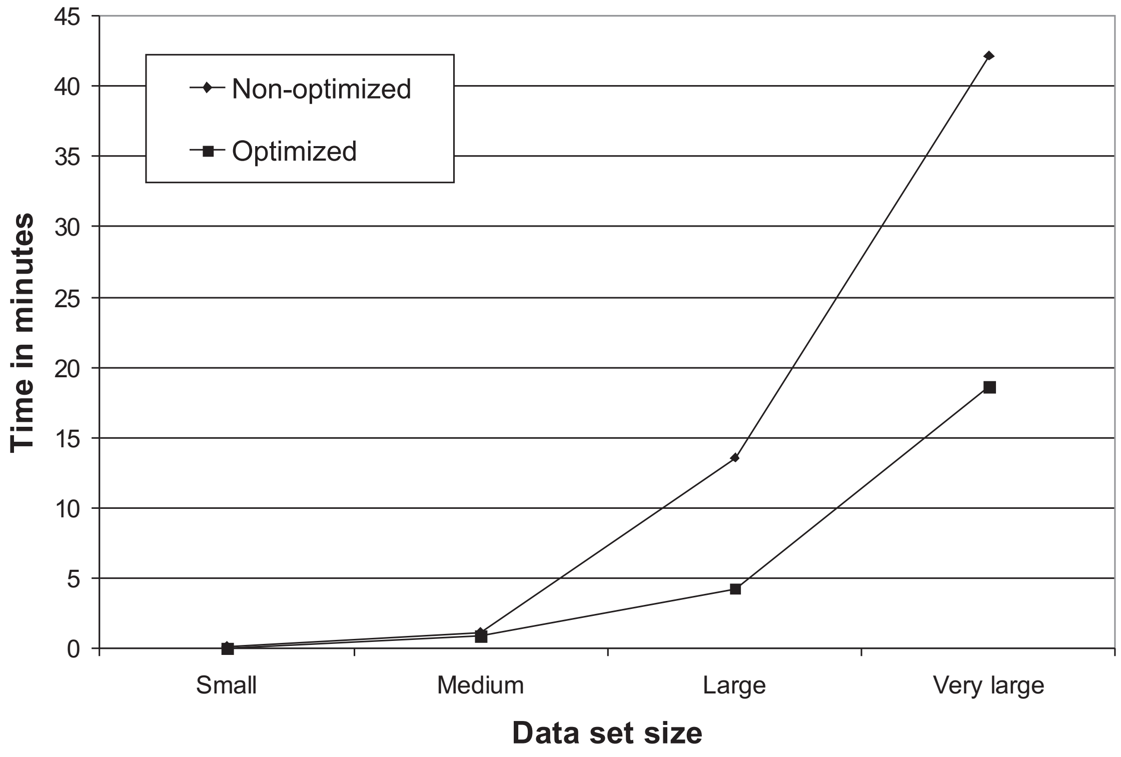
Figure 6. Efficiency of example query for per-feature and feature-hybrid quality

model. Thus, applications adopting one quality model or the other are not "locked in" to that model, and can potentially change to another quality model as requirements and data change.

A range of future work is suggested by this research. A key long-term motivation for this work is to develop an integrated database engine capable of supporting the wide range of different spatial data quality management tools and techniques proposed or currently in existence, or even higher-level quality management functionality, such as error propagation systems (e.g., the data uncertainty engine, DUE, Heuvelink 2007). In particular, tools to assist with updating and revision of data quality information could be supported by the RSVQ system, and could help users to ensure a consistent database is maintained (for example, ensuring the entries in the metadata table are automatically updated when new tables are added or removed).

Although some queries are relatively efficient, response times for others (where spatial joins are involved) can be slow. One possible approach to this would be to combine RSVQ with OLAP tools, such as those already explored by Devillers et al. (2005). Using OLAP essentially materializes ("caches") the results of a range of carefully chosen useful queries, which can be pre-computed and then subsequently dynamically explored by a user requiring rapid responses. Other alternatives would be to investigate non-relational database models for data quality storage (such as object-oriented models). While some work exists examining this possibility (e.g., Duckham 2001), none have yet systematically or experimentally investigated the query or storage efficiency of these approaches.

Finally, the RSVQ model can claim to store a very wide range of quality information, effectively any quality data that can be stored as a data a database (see section 2.5). However, some elements of data quality, like logical consistency, may be better represented as procedures or processes (e.g., the process by which consistency is determined), rather than stored data (e.g., the result of a consistency assessment, whether the data is consistent or not). The RSVQ model does not deal with storage of procedure-based data quality, although other common RDBMS tools and techniques (such as constraints and triggers) may still be able to support such procedures. Extending the RSVQ model to also include the capability to store and query process-based data quality would be a valid topic for future research, although designing and implementing such a capability is not expected to present substantial problems (as basic consistency checking, for example, is already a basic and widely used function of today's RDBMS).



[htb]

Figure 7. Efficiency of example query for feature independent quality (optimized and non-optimized)

REFERENCES

Aalders, H. J. G. L. (2002). Registration of quality in a GIS. In Shi, W., Fisher, P. F., and Goodchild, M. F., editors, *Spatial Data Quality*, pages 186–199. Taylor & Francis, London.

Blakemore, M. (1984). Generalization and error in spatial databases. *Cartographica*, 21:131–139.

Buyong, T. B., Kuhn, W., and Frank, A. U. (1991). A conceptual model of measurement-based multipurpose cadastral systems. *URISA Journal*, 3(2):35–49.

Chrisman, N. (1984). The role of quality information in the long-term functioning of a geographic information system. *Cartographica*, 21:79–87.

Chrisman, N. (1991). The error component in spatial data. In Maguire, D. J., Goodchild, M. F., and Rhind, D. W., editors, *Geographical Information Systems: Principles & Applications*, volume 1, pages 179–189. Wiley.

Devillers, R., Bédard, Y., and Jeansoulin, R. (2005). Multidimensional Management of Geospatial Data Quality Information for its Dynamic Use Within GIS. *Photogrammetric Engineering & Remote Sensing*, 71(2):205–215.

Duckham, M. (2001). Object calculus and the object-oriented analysis and design of an error-sensitive GIS. *Geoinformatica*, 5(3):261–289.

Egenhofer, M. J. (1994). Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86–95.

Elmasri, R. and Navathe, S. B. (1994). *Fundamentals of Database Systems.* Benjamim/Cumings Publishing Company, San Francisco, second edition.

Feuerstein, S. (2002). *Oracle PL/SQL Programming.* O'Reilly.

Fisher, P. (1989). Knowledge-based approaches to determining and correcting areas of unreliability in geographic databases. In Goodchild, M. F. and Gopal, S., editors, *Accuracy of Spatial Databases*, pages 45–54. Taylor and Francis, London.

Fisher, P. F. (1991). Spatial data sources and data problems. In Maguire, D. J., Goodchild, M. F., and Rhind, D. W., editors, *Geographical Information Systems: Principles & Applications*, volume 1, pages 179–189. Wiley, New York.

Gan, E. and Shi, W. (2002). Error metadata management system. In Shi, W., Fisher, P. F., and Goodchild, M. F., editors, *Spatial Data Quality*, pages 251–266. Taylor & Francis, London.

Goodchild, M. F. (1995). Sharing imperfect data. In Onsrud, H. and Rushton, G., editors, *Sharing Geographic Information*, pages 413–425. Rutgers, New Jersey.

Goodchild, M. F. (2002). Measurement-based GIS. In Shi, W., Fisher, P. F., and Goodchild, M. F., editors, *Spatial Data Quality*, pages 5–17. Taylor & Francis, London.

Guptill, S. C. and Morrison, J. L., editors (1995). *Element of Spatial Data Quality*. Elsevier, New York.

Heuvelink, G. B. M. (1996). Identification of field attribute error under different models of spatial variation. *International Journal Geographical Information Systems*, 10(8):921–935.

Heuvelink, G. B. M. (1998). *Error propagation in environmental modeling with GIS*. Taylor & Francis, London.

Heuvelink, G. B. M. (2007). Error-aware GIS at work: Real-world applications of the data uncertainty engine. In *Proc. Fifth International Symposium on Spatial Data Quality*.

International Organization for Standardization (1999). *ISO/IEC 9075-2:1999: Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*. International Organization for Standardization.

Kainz, W. (1995). Logical consistency. In Guptill, S. C. and Morrison, J. L., editors, *Elements of Spatial Data Quality*, pages 109–138. Elsevier, New York.

Moellering, H. (1997). *Spatial database transfer standards 2: Characteristics for assessing standards and full descriptions of the national and international standards in the world*. Elsevier, New York.

National Committee for Digital Cartographic Data Standards (1988). The proposed standards for digital cartographic data. *The American Cartographer*, 15(1):131–135.

Navratil, G., Franz, M., and Pontikakis, E. (2004). Measurement-based GIS revisited. In *7th AGILE Conference on Geographic Information Science*, pages 771–775, Heraklion, Greece.

Qiu, J. and Hunter, G. J. (2002). A GIS with the capacity for managing data quality information. In Shi, W., Fisher, P. F., and Goodchild, M. F., editors, *Spatial Data Quality*, pages 230–250. Taylor & Francis, London.

Sadiq, Z. and Duckham, M. (2007). Storing and querying spatially varying data quality information using an integrated spatial RDBMS. In *ISSDQ, 5th International Symposium on Spatial Data Quality*.

Usery, E. L. (1996). A feature-based geographic information system model. *Photogrammetric Engineering and Remote Sensing*, 62(7):833–838.

Wong, D. W. S. and Wu, C. V. (1996). Spatial metadata and GIS for decision support. In *Proc. 29th Annual Hawaii International Conference on System Sciences*, Hawaii. IEEE.