MODELING SPATIAL VARIATION
OF DATA QUALITY
IN DATABASES

PhD THESIS

SHAIK MOHAMED ZAFFAR SADIQ MOHAMED GHOUSE

Submitted in total fulfillment of the requirements
of the degree of Doctor of Philosophy

May, 2008

Produced on archival quality paper
Abstract

The spatial data community relies on the quality of its data. This research investigates new ways of storing and retrieving spatial data quality information in databases. Given the importance of features and sub-feature variation, three different data quality models of spatial variation in quality have been identified and defined: per-feature, feature-independent and feature-hybrid. Quality information is stored against each feature in the per-feature model. In the feature-independent model, quality information is independent of the feature. The feature-hybrid is derived from a combination of the other two models. In general, each model of spatial variation is different in its representational and querying capabilities. However, no model is entirely superior in storing and retrieving spatially varying quality. Hence, an integrated data model called as RDBMS for Spatial Variation in Quality (RSVQ) was developed by integrating per-feature, feature-independent and feature-hybrid data quality models. The RSVQ data model provides flexible representation of SDQ, which can be stored alongside individual features or parts of features in the database, or as an independent spatial data layer.

The thesis reports on how Oracle 10g spatial RDBMS was used to implement this model. An investigation into the different querying mechanisms resulted in the development of a new WITHQUALITY keyword as an extension to SQL. The WITHQUALITY keyword has been designed in such a way that it can perform automatic query optimization, which leads to faster retrieval of quality when compared to existing query mechanism. A user interface was built using Oracle Forms 10g which enables the user to perform single and multiple queries in addition to conversion between models (example, per-feature to feature-independent). The evaluation, which includes an industry case study, shows how these techniques can improve the spatial data community’s ability to represent and record data quality information.

Keywords

quality, spatial variation, metadata, sub-feature variation, per-feature, feature-independent, feature-hybrid
Declaration

This is to certify that:

(i) the thesis comprises of only my original work towards the PhD except where indicated,

(ii) due acknowledgment has been made in the text to all other material used,

(iii) the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies, appendices and footnotes.

SHAIK MOHAMED ZAFFAR SADIQ MOHAMED GHOUSE
Preface

This work has been supported by the Cooperative Research Centre for Spatial Information, whose activities are funded by the Australian Commonwealth’s Cooperative Research Centers Program.

The data for Hume Local Government Area is part of cadastral data produced by the Department of Sustainability and Environment, Victoria, Australia.
Publications

During the course of this project, a number of public presentations have been made which are based on the work presented in this thesis. They are listed here for reference.

Journal article


Reviewed Conference Papers

Sadiq, Z., Duckham, M., 2007. Storing and querying spatially varying data quality information using an integrated spatial RDBMS. In: 5th International Symposium on Spatial Data Quality, ITC Enschede, The Netherlands

Conference Papers


Acknowledgments

First and foremost, I would like to express my gratitude to my supervisor, Dr Matt Duckham. As a true guru, Matt led me through the obstacles of scientific thinking and writing on the way to become a researcher. Without his encouragement, guidance, support and patience, this thesis would not exist. I would like to thank my co-supervisors Associate Professor Gary J Hunter and Dr Allison Kealy for their constant support, technical advice and encouragement.

I would also like to thank my industry mentors Dr Geoff Lawford, Geoscience Australia for useful discussions and Robert Morrison from the Department of Sustainability and Environment (Victorian state government department) for providing Vicmap data for the research and feedback.

I also thank the Department of Geomatics and the Cooperative Research Centre for Spatial Information (CRCSI) at the University of Melbourne for the pleasant working environment and its generosity in the provision of resources and funding. My thanks are extended to my colleagues, Anna Boin, Martin Tomko, Jochen Wilneff, Joanne Poon, Stephan Hansen, Sue Hope, Jane Inall, Dr Graeme Kernich, Subash Sharma for the discussions, support, encouragement, fun and table tennis matches. And special thanks to Dr Patrik Laube for reading through the draft of this thesis.

Deep gratitude goes to my family and friends who have never failed to help and support me. Finally, I would like to express my thanks to my parents, Dr Shaik M Ghouse and Saleemunissa Ghouse, and my mother-in-law, Shaistha Banu for the years of love and encouragement. Last, but not least, I would like to thank my wife, Sumera Naaz and my son, Taha M Shaik for their patience, support and love during the entire period of the research work. Thank you for everything!

M.G.S.M. Zaffar Sadiq
# Contents

1 Introduction  
1.1 Overview .................................................. 1  
1.2 Spatial data quality ......................................... 1  
1.2.1 Variation in spatial data quality ......................... 3  
1.2.2 Spatial variation in data quality ....................... 3  
1.3 Motivation .................................................. 4  
1.3.1 Need to represent spatial variation in databases ........ 4  
1.3.2 Current approaches and their limitations .............. 5  
1.4 Scope and Objectives ....................................... 6  
1.4.1 Sub-feature variation ...................................... 6  
1.4.2 Research application ...................................... 7  
1.4.3 Problem definition ......................................... 8  
1.4.4 Research hypothesis and objectives .................... 8  
1.5 Approach .................................................. 9  
1.6 Expected outcomes ......................................... 10  
1.7 Thesis outline .............................................. 11  
1.8 Summary .................................................. 11  

2 Background .................................................. 13  
2.1 Introduction ................................................. 13  
2.2 Representation ............................................. 13  
2.2.1 Elements of spatial data quality ....................... 14  
2.2.2 Other representations of SDQ ............................ 15  
2.2.3 Standard organisations on data quality ............... 16  
2.3 Architecture ............................................... 17  
2.3.1 Hybrid quality system .................................... 17  
2.3.2 Integrated quality system ............................... 18  
2.4 Database concepts ......................................... 18
## 2.4.1 Relational algebra ............................................. 20
## 2.5 Storage ................................................................ 22
## 2.6 Querying spatial variation ..................................... 25
  2.6.1 Non-spatial query ............................................... 25
  2.6.2 Spatial query .................................................... 25
  2.6.3 Query optimization ............................................ 26
## 2.7 Spatial variation data models ................................. 27
## 2.8 Summary .............................................................. 27

### 3 Models of spatial variation in data quality 29

#### 3.1 Introduction ..................................................... 29

#### 3.2 Related data models .......................................... 29
  3.2.1 Per-feature model .............................................. 30
  3.2.2 Feature-independent model ................................. 30
  3.2.3 Feature-hybrid model ......................................... 31

#### 3.3 Prototype I ....................................................... 33
  3.3.1 Scenario .......................................................... 33
  3.3.2 Per-feature implementation ................................. 33
  3.3.3 Feature-independent implementation ..................... 35
  3.3.4 Feature-hybrid implementation ............................ 35
  3.3.5 Prototype I summary ........................................... 38

#### 3.4 Prototype II ..................................................... 39
  3.4.1 Storage .......................................................... 39
  3.4.2 Querying .......................................................... 43
  3.4.3 Prototype-II summary .......................................... 54

#### 3.5 Summary .......................................................... 55

### 4 RSVQ: RDBMS for Spatial Variation in quality 57

#### 4.1 Introduction ..................................................... 57

#### 4.2 Formal model .................................................. 57
  4.2.1 Preliminaries .................................................... 58
  4.2.2 Quality operator ............................................... 60
  4.2.3 Quality operator behaviour ................................ 60

#### 4.3 Efficiency considerations ................................... 62

#### 4.4 Example queries ............................................... 63

#### 4.5 Summary .......................................................... 64
5 Formal model implementation

5.1 Introduction ................................................................. 65
5.2 Study area ................................................................. 65
  5.2.1 Location ............................................................... 65
  5.2.2 SDQ of Hume LGA .................................................. 65
5.3 Database design .......................................................... 67
  5.3.1 RSVQ relationship ................................................... 67
  5.3.2 Data format and database scheme ............................... 67
5.4 WITHQUALITY keyword ................................................ 71
  5.4.1 Existing query mechanism .......................................... 71
  5.4.2 New query mechanism .............................................. 71
  5.4.3 Working of WITHQUALITY keyword .......................... 73
  5.4.4 Optimization of WITHQUALITY keyword ..................... 80
  5.4.5 Multiple table query .............................................. 89
  5.4.6 Working of multiple table query ................................. 90
5.5 Conversion between models ............................................. 92
5.6 User interface ............................................................ 96
5.7 Summary ...................................................................... 98

6 Experimentation and Results .............................................. 99

6.1 Introduction .................................................................. 99
6.2 Experimental data ........................................................ 99
  6.2.1 Storage analysis ...................................................... 101
6.3 Experiment design ........................................................ 103
6.4 Query evaluation ......................................................... 106
  6.4.1 Per-feature model evaluation ..................................... 108
  6.4.2 Feature-independent model evaluation ......................... 112
  6.4.3 Feature-hybrid model evaluation ............................... 115
  6.4.4 Query evaluation discussion ..................................... 117
  6.4.5 Re-engineering the model ......................................... 118
6.5 Summary and conclusions .............................................. 118

7 Conclusions and recommendations ..................................... 121

7.1 Research contributions ................................................ 121
7.2 Recommendations ........................................................ 123
  7.2.1 How to update quality in RSVQ? ............................... 123
List of Tables

2.1 Location .............................................. 19
2.2 Output of sample query 1 .............................. 19
2.3 Output of sample query 2 .............................. 20
2.4 Digital reliability diagram legend ................... 23

3.1 PFQ (Per-Feature Quality table for sample parcels in Panton Hill Locality) ...................... 40
3.2 FIQ (Feature-Independent Quality table for sample parcels in Panton Hill Locality) ................. 41
3.3 FHQ (Feature-Hybrid Quality table for sample parcels in Panton Hill Locality) ...................... 42

4.1 Summary of behaviours for polymorphic quality operator behaviour 61

5.1 Oracle table-parplh implementation (sample record) .................. 69
5.2 Oracle table-pfoh implementation (sample record) .................. 69
5.3 Oracle table-fioh implementation (sample record) .................. 70
5.4 Oracle table-floh implementation (sample record) .................. 70
5.5 Metadata table ....................................... 70

6.1 Experiment design .................................... 105
6.2 Results of the experiments. NS: Not Significantly different, Non-opt: Non-optimised approach faster, Opt: Optimised approach faster, NS* indicates 3 out of 4 data sets were NS. .......... 108
List of Figures

1.1 Sub-feature variation across a parcel, Hume, Local Government Area, Victoria, Australia ....................................................... 2

2.1 Digital reliability diagram to indicate reliable features ............. 23

3.1 Example of Per-feature model .............................................. 31

3.2 Example of Feature-independent model .................................. 32

3.3 Example of Feature-hybrid model ........................................ 32

3.4 Per-feature model for linear features .................................... 34

3.5 Per-feature model for polygon features ................................... 34

3.6 Feature-independent model for linear features ......................... 36

3.7 Feature-independent model for polygon features ...................... 36

3.8 Feature-Hybrid model for linear feature ............................... 37

3.9 Feature-Hybrid model for cyclic linear feature. ....................... 38

3.10 Base layer depicting sample parcels in Panton Hill Locality .... 40

3.11 FIQ (Feature-Independent Quality model for sample parcels in Panton Hill Locality ) .......................................................... 41

3.12 FHQ (Feature-Hybrid Quality model for sample parcels in Panton Hill Locality ) .......................................................... 42

3.13 Spatial overlay depicting base layer and region intersection .... 48

3.14 Results of query 3.1 ......................................................... 48

3.15 Intersect operation on results of base intersect fiq over region.geom 49

3.16 Results of query 3.2 ......................................................... 50

3.17 Results of query 4.1 ......................................................... 52

3.18 Results of query 4.2 ......................................................... 52

3.19 Results of query 4.3 ......................................................... 53

4.1 Integrated quality data model .............................................. 58

5.1 Hume LGA Parcels ......................................................... 66
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>Hume LGA positional accuracy</td>
<td>66</td>
</tr>
<tr>
<td>5.3</td>
<td>UML diagram depicting the relationship between the qualities and layer relation</td>
<td>68</td>
</tr>
<tr>
<td>5.4</td>
<td>Oracle spatial SDO GEOMETRY structure</td>
<td>68</td>
</tr>
<tr>
<td>5.5</td>
<td>Flowchart 1: steps involved in sample query 1 execution</td>
<td>74</td>
</tr>
<tr>
<td>5.6</td>
<td>Flowchart 2: the steps involved in the function dia</td>
<td>76</td>
</tr>
<tr>
<td>5.7</td>
<td>Query tree for sample query 2</td>
<td>82</td>
</tr>
<tr>
<td>5.8</td>
<td>Optimized query tree for sample query 2</td>
<td>83</td>
</tr>
<tr>
<td>5.9</td>
<td>Flowchart part-1: optimized query tree implementation</td>
<td>85</td>
</tr>
<tr>
<td>5.10</td>
<td>Flowchart part-2: optimized query tree implementation</td>
<td>88</td>
</tr>
<tr>
<td>5.11</td>
<td>Flowchart-Multiple table query-Part1</td>
<td>91</td>
</tr>
<tr>
<td>5.12</td>
<td>Flowchart-Multiple table query-Part2</td>
<td>93</td>
</tr>
<tr>
<td>5.13</td>
<td>Flowchart-Multiple table query-Part3</td>
<td>94</td>
</tr>
<tr>
<td>5.14</td>
<td>User Interface</td>
<td>97</td>
</tr>
<tr>
<td>6.1</td>
<td>Hume LGA-locality boundary</td>
<td>100</td>
</tr>
<tr>
<td>6.2</td>
<td>Storage evaluation</td>
<td>101</td>
</tr>
<tr>
<td>6.3</td>
<td>Example of feature-hybrid quality generation</td>
<td>103</td>
</tr>
<tr>
<td>6.4</td>
<td>Variables for experiment</td>
<td>104</td>
</tr>
<tr>
<td>6.5</td>
<td>Per-feature model query evaluation</td>
<td>109</td>
</tr>
<tr>
<td>6.6</td>
<td>Subset table check flowchart</td>
<td>111</td>
</tr>
<tr>
<td>6.7</td>
<td>Feature-independent model query evaluation</td>
<td>113</td>
</tr>
<tr>
<td>6.8</td>
<td>Feature-hybrid model query evaluation</td>
<td>116</td>
</tr>
<tr>
<td>6.9</td>
<td>Re-engineered RSVQ flowchart with new function FOB</td>
<td>119</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Overview

Data quality is crucial to assess the fitness for use of spatial data and is therefore the key to achieve interoperability (Frank, 2004). Standards organizations have recognized the importance of data quality and have made provisions to report the quality in the form of data quality statements (Qiu, 2002). In reality, spatial data quality (SDQ) varies according to location. A real world example of spatially varying quality is the cadastral data produced by different cadastral systems in the European Union (Navratil, 2004). A more specific example is the cadastral data produced by the Department of Sustainability and Environment (DSE), Victoria, Australia (the government department that is responsible for maintaining and updating spatial data for the state of Victoria). DSE reports that this data can have positional accuracy that varies within individual land parcels (see figure 1.1). Current spatial data quality storage mechanisms are inadequate for representing such sub-feature variation. As a result, the actual positional accuracy of DSE data in some locations can be higher than the reported accuracy (Ramm, 2005). To represent spatially varying data quality, SDQ may need to be stored for every feature in the database or even for parts of features.

1.2 Spatial data quality

Before discussing variation in SDQ, let us understand what is meant by this term SDQ. There is a variety of definitions in the literature. According to Jakobsson (2002), spatial data quality is the difference between the universe of discourse and the data set, where the data set is defined as an identifiable collection of related data and the universe of discourse as the view of the real world that includes everything of interest. Korte (2001) defines spatial data quality as the degree to
which GIS data accurately represents the real world, the suitability of the data for a certain purpose and the degree to which the data meet a specific accuracy standard. Korte’s definition (2001) is close to the definition of accuracy which is one component of SDQ. Worboys and Duckham (2004) associate uncertainty of data with decision making and refer to data quality as the characteristics of a data set that may influence the uncertainty associated with decisions based upon that data set. Juran et al. (1974) proposed a definition based on the usability of the data as “fitness for use”, which was also mentioned in an article by Chrisman (1984). Hunter (1996) extends (Juran et al., 1974) and (Chrisman, 1984) definition by stating, “From a client’s perspective a data set may be fit for use even if its quality is low, as long as it suits the client’s purpose”.

The definitions of Jakobsson (2002) and Korte (2001) can be classified as a producer’s view of SDQ. The definitions of Worboys and Duckham (2004), Chrisman (1984) and Hunter (1996) can be classified as a user’s perspective on SDQ. In this thesis we adopt a definition of spatial data quality that combines the definition from both the producers’ and users’ perspective as:

“Spatial data quality is the set of characteristics of a spatial data set that influence the level of confidence a data user or producer has in that data”
1.2.1 Variation in spatial data quality

The quality of spatial data may vary between data sets, termed *inter-source* variation. However, the quality of spatial data may also vary within a data set, termed *intra-source* variation. In general, at least three different types of intra-source variation in data quality for spatial data can be identified:

1. thematic variation in data quality;
2. temporal variation in data quality; and
3. spatial variation in data quality.

*Thematic variation* in data quality occurs between different feature classes or individual features within a spatial data set, for example, where one feature class is captured at higher attribute accuracy than another. *Temporal variation* in data quality is variation that occurs over time, for example, where new quality information is received following revision or update of spatial data. *Spatial variation* in data quality occurs when the quality of spatial data varies as a function of spatial location, for example, where data quality is known to be higher in one region of space than another. Each type of variation needs to be considered in managing spatial data quality. However, the key focus of the research will be on spatial variation.

1.2.2 Spatial variation in data quality

Spatial variation is a characteristic of data quality that is unique to spatial data sets. The quality of spatial data for one area may not be applicable to spatial data describing other regions (Wong and Wu, 1996). Clarke and Clark (1995) state that different data capturing techniques, like field surveys, global positioning systems, aerial photography, satellite imagery, existing maps and other documentation, are amongst the key factors affecting spatial variation in SDQ. According to Wong and Wu (1996), the quality of data varies spatially due to problems in data collection (for example, when socio-economic data is gathered by sampling and not all regions are sampled to the same extent), data capture (for example, if cloud cover exists in a remotely sensed image, the data for certain areas covered by clouds in the database may be less accurate than the data describing other regions), compilation, analysis, and representation.
1.3 Motivation

1.3.1 Need to represent spatial variation in databases

Spatial variation in data quality is a common feature of spatial data sets. Hunter (2001) uses the example of metadata (data about a data) that reports positional accuracy as $\pm 1.5m$ in urban areas and $\pm 250m$ in rural areas. Mostly, metadata do not report where this variation occurs. The information would be more meaningful if the location of the different positional accuracies were reported. Knowing the locations where different data qualities exist is clearly important to decision making using a data set. With the advent of new positioning technologies used for cadastral upgrading, updated data often has better accuracy than the data it replaces (Hunter et al., 2005).

Spatial data quality is conventionally presented in the form of a report. The data quality statements in the report typically refer to the entire data set. The data users determine the usability of the data based on these statements. The users of a data set need to use information about spatial data quality in order to be able to assess the “fitness for use” of the data set. In its action plan (2003-2004) ANZLIC (Australia, New Zealand Land Information Council) has stated that the spatial industry in the region has to overcome the problem of incomplete knowledge about the availability and quality of existing spatially referenced data (ASDI standing committee, 2003). Therefore, the producers of a data set need to effectively communicate the quality of their data for a variety of reasons, including competitiveness, data warehousing, liability, and litigation. Graham (1997) states that:

“\textit{It becomes imperative from a liability perspective to check the positional and attribute accuracy, logical consistency, resolution, completeness, timeliness and lineage of data.}”

Consequently, conventional approaches to SDQ that ignore variation in quality within a data set impair the producer’s ability to correctly communicate knowledge about data quality and jeopardize the user’s ability to assess fitness for use.

Furthermore, Hunter et al. (2005) consider that “better accuracy” is often due to new technologies that are used for cadastral upgrading, or when DEM accuracy is improved by the use of new imaging sources such as LIDAR. Therefore, the clients will be interested to know where more accurate data lies within the data set. As a result, there is a need to communicate spatial variation of SDQ. To communicate spatial variation it is first necessary to represent it in the database.
1.3.2 Current approaches and their limitations

SDQ storage in spatial databases can be broadly classified into two representations: external and integrated. In the external representation, the SDQ information is stored separately from the spatial database. Burrough (1987) had discussed simple and complex multi-scale models to handle multiple sources of spatial variation, which can be cited as an example of external representation. The advantage of the external representation is that it is easier to create and manage. The disadvantage is that it is harder to maintain the link between spatial data and externally stored SDQ information and harder to update and query that information. Moreover, SDQ information is often aggregated for the entire spatial database, ignoring spatial variation in the external representation. Devillers et al. (2005) have criticized the linked metadata approach (external representation) as: (1) there are limitations in the type of metadata that can be stored and (2) the level of detail of the metadata information.

In the integrated representation SDQ is stored along with the spatial database. By adopting an integrated representation the user can more easily represent and query spatially varying quality. The issue with the integrated approach is the management of voluminous data. Chrisman (1984) also raised the concern that the amount of quality information might take up as much space in data storage as geometric coordinates. Another issue with the integrated approach is the time taken to compute the voluminous data. Navratil et al. (2004) implemented measurement-based GIS (integrated representation) for a test area of 7km². The implementation concludes that the computers cannot provide the results of a global adjustment (process to improve positional accuracy) in a reasonable time. However, the time taken to compute can vary based on the methodology adopted for implementing the measurement-based GIS and the processing power of the computer.

Several existing studies have adopted an integrated approach to storing SDQ. Duckham (2001) has developed a formal model of object-based variation in spatial data quality, using object calculus. Gan and Shi (2002) have developed an Error Metadata Management System (EMMS) to represent quality at feature level. Similarly, Qiu and Hunter (2002) have developed a model to represent spatial variation of quality in a database up to the feature-level. Devillers et al. (2005) have designed a Quality Information Management Model (QIMM), which has access to different levels of detail on spatial data quality. QIMM is an extension of Qiu’s (2002) work, based on multidimensional spatial database analysis tools (SOLAP).
The existing studies discussed above have two important drawbacks. By storing quality information for each object in the database, some spatial variation can be represented across the layer. However, these approaches are limited in their representational capabilities, since they cannot adequately represent variation in quality within an object termed *sub-feature variation* (see section 1.4.1) in a database environment.

Secondly, the approaches do not address the efficiency of storage and retrieval mechanisms. Efficient storage and querying mechanisms are important factor when representing SDQ for each object or sub-object in the database, since such an approach can potentially result in voluminous SDQ information. One might argue that data storage and processor costs are continually decreasing. Although future advances in technology will undoubtedly make computers faster and cheaper, the focus of the efficiency analysis is on the scalability of processing, i.e., how storage and processing change as a function of input data set size. Scalability is a fundamental characteristic of data structures and algorithms, and is independent of future technological changes. For example, a computational process that scales increases linearly in computing time with increasing input data set size (e.g., double data set size, double processing time) will always scale linearly regardless of future technological advances. However, irrespective of storage cost, efficient and effective storage and querying are essential for handling large volumes of SDQ data.

### 1.4 Scope and Objectives

#### 1.4.1 Sub-feature variation

Goodchild (1988) states that the attributes are assumed to be spatially invariant or homogeneous over the spatial object; alternatively the system may infer systematic spatial variation of attributes over or between objects (sub-feature variation). *Sub-feature variation* can be defined as a *spatial variation within a geographic feature*. Standard survey adjustments have supported variation of quality within a feature ((Buyong et al., 1991); (Tobler, 1995)). One of the critical questions that is not adequately addressed by the models discussed in section 1.3.2 is, how to efficiently represent sub-feature variation of quality information in a database?

Researchers in the late 1980s and early 1990s have addressed the issue of representing sub-feature variation. However, each approach has its own advantages and disadvantages with respect to storage and querying efficiency. The multiscale
spatial database introduced by (Jones, 1991) and extended by (Kinder and Jones, 1993) and (Jones et al., 1994) enable subsets of spatial objects that are part of a topographic surface to be retrieved at variable levels of detail determined by the scale of the required output. The database represents the multiresolution topographic surface by a sequence of levels where the top level is the coarsest resolution and the bottom is the finest resolution. Although the multiscale database provides a platform to represent sub-feature variation, the limitation of the database is the adoption of a quadtree method. In the quadtree method, each time a feature is updated the corresponding structure of the quadtree also has to be modified (see section 2.5 for further discussions on quadtree). Similarly, Kleiner (1989) has used a cell method which subdivides space into a regular grid. By using cell method, sub-feature variation is represented but the querying cost is high as the cells have to be composed to retrieve a spatial object.

Active research is carried out in the field of remote sensing to represent sub-pixel and mixed pixel classification (Foschi, 1994) in raster-based data structure. Sub-feature variation is not a problem for points in vector-based spatial data. However, lines and polygons in vector data sets can exhibit sub-feature variation in data quality. Hence, the thesis mainly focuses on representing and retrieving sub-feature variation in vector-based spatial data.

1.4.2 Research application

Buyong et al. (1991) discuss a measurement-based multipurpose cadastral system using least square adjustment to process the measurements and store them in a measurement database. The measurements include measurements between higher-quality points (in coordinate-based systems, such measurements are known as control point measurements), parcel boundary measurements and the object of interest measurement (Buyong et al., 1991). This research’s querying functionality can be useful in updating new measurement information in the measurement based system, especially updating measurements within an object (sub-feature).

The work of Hope and Gielsdorf (2007) addresses ways to upgrade low positional accuracy (low positional accuracy due to generation of spatial data sets from existing paper maps for the Victorian state government department DSE and high positional accuracy data obtained from high resolution imagery and positioning devices based on Global Navigation Satellite Systems (GNSS)). As a result, a cadastral land parcel may have different positional accuracy information associated with the different vertices (sub-features) which make up that parcel. The parcel (data obtained from DSE) in figure 1.1 (a real world example of sub-feature variation) has quality (positional accuracy) values of 0.5m and 2.5m
precision associated with it. This research work on storing and querying spatially varying data in a database will support Hope’s work to first query low and high positional accuracy data, especially where sub-feature variation is located. Based on the query results, procedures can be run to integrate low positional accuracy data with high positional accuracy data.

The representation of data quality components in a data structure will not only have requirements to facilitate their visual display, but also must be implemented with efficient pointers and links to facilitate update operations (Buttenfield and Beard, 1991). The data model of the research will provide a platform to visualise data quality information, which will enable the user to first query the spatial data quality before starting to use the spatial data. Likewise as the research’s data model is aimed at integrated model updating of quality information is aimed to be straight forward.

1.4.3 Problem definition

Referring to the parcel in figure 1.1, which has varying positional accuracies of 0.5m and 2.5m precision associated within the parcel, the issue here is that to represent this sub-feature variation the parcel has to be segmented into smaller features with homogeneous quality values associated with each. In other words, one record is split into many records (segmentation) for the sake of storing metadata (quality) information. As a result of segmentation, the data structure of the parcel is altered. In order to avoid segmentation, we can store spatially varying quality as a separate layer in the database. By performing overlay (spatial join) between parcel and quality (positional accuracy) layer, we can retrieve spatially varying quality. Spatial joins involve retrieving tuples from two or more relations based on a spatial join condition. Spatial joins are amongst the most computationally expensive operations in a spatial database. There are still issues involved in handling sub-feature variation as discussed in this section, which emphasise the need for new data models that could report spatially varying quality up to sub-feature level irrespective of the spatial data type.

1.4.4 Research hypothesis and objectives

Based on the discussions in the problem definition section, the research question was framed as:

“How do we efficiently and effectively model spatial variation of data quality in spatial databases?”
Research hypothesis: The above research question leads to the following research hypothesis:

“Existing relational spatial database technology can be used to model and evaluate spatial variation including sub-feature variation of spatial data quality.”

Research objectives: To enable us to answer the research question, this thesis adopts the following research objectives:

1. to identify and classify data models to represent spatial variation of quality;
2. to structure and analyse the related data models by their representation and querying capabilities;
3. to develop an integrated model by integrating the identified spatial variation models;
4. to develop an efficient and effective querying mechanism; and
5. to evaluate the integrated model using an analytical and experimental approach.

1.5 Approach

The research method is subdivided into three components: storage models; querying; and evaluation. The storage models are classified based on the object- and field-based models. 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 et al., 2006). Quality information is stored against each feature in the per-feature model. In the feature-independent model quality information is independent of the feature and stored as a separate layer. The feature-hybrid model is derived from a combination of the other two models.

The mechanism to retrieve spatial variation in data quality is discussed in the querying component. The querying structure depends on each model’s data organisation. Thus, each model will have a different behaviour whilst retrieving spatially varying quality. A detailed study was carried on querying behaviour of
each model. The results of the query behaviour study paved way to the development of a single quality query operator. The quality operator was developed with an automatic query optimization technique for faster retrieval of quality when compared to existing query mechanism. Benchmark queries were developed to test the quality query operator.

Finally, the evaluation component deals with the analytical and experimental evaluation of storage and querying functionalities. It is proposed to evaluate the different models of spatial variation in SDQ based on both analytical and experimental approaches. Analytical approach (formal model discussed in chapter 4) helps to measure efficiency independent of processing power of the computer. There are limitations to the analytical evaluation as highly complex and stochastic procedures cannot be easily modeled. Therefore, the experimental approach is also important for evaluation. Each model has its own characteristics with respect to storage and querying functionalities. Thus, it is proposed to compare the models by developing an evaluation matrix.

1.6 Expected outcomes

The expected outcomes of the research are as follows:

1. Data model: The first outcome is an integrated data quality model. The data model provides a framework to model per-feature, feature-independent and feature-hybrid data for a data set. Irrespective of which data model is used to store cadastral data’s quality elements, the integrated data model aims to provide a single framework to model all quality elements in a data set. 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 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.

2. Query mechanism: The second outcome of the research is the quality query operator. A derived polymorphic relational query operator is developed to retrieve spatially varying data quality irrespective of which data model is used. The quality query operator has the capability to retrieve more than one quality information at a time. For example, a parcel’s lineage, positional accuracy and precision information can be retrieved in a single query.
using the quality query operator. Furthermore, the quality query operator is capable of optimizing the queries which results to faster data retrieval.

3. User interface: Although the development of an user interface was not one of the main objectives of the research, the user interface was developed to interact with the data model.

1.7 Thesis outline

This thesis is structured as follows: the next chapter provides an overview of existing literature in spatial data quality, spatial variation data models, storing and querying data quality in databases. Chapter 3 introduces related data models to represent variation in data quality. Apart from prototype implementation of the related data models, a detailed analysis of the implementation based on storage and querying capabilities of the prototype have been discussed in chapter 3. Chapter 4 introduces a formal data model using relational algebra to overcome the limitations of the prototype discussed in chapter 3. The implementation of the formal model in Oracle spatial RDBMS along with an industry case study is discussed in Chapter 5. Chapter 6 provides an analysis of the results of the case study. Conclusions and recommendations of the research are highlighted in chapter 7.

1.8 Summary

The chapter introduces spatial variation in data quality citing examples in real world. Later sections of the chapter outline the motivation for the research with current approaches and their limitations. The scope and objectives of the research were discussed elaborately along with expected outcomes of the research. The chapter concludes with a briefing on the thesis structure.
Chapter 2

Background

2.1 Introduction

Conventionally, SDQ is often represented in the form of a report. In reality the quality of spatial data will vary at different geographic locations. Spatial variation is the main category of intra-source variation which is not adequately addressed by conventional approaches to data quality. Monmonier (1993) states that the quality of data will affect users’ confidence in decisions (and ultimately success of policies) based on that data. Hence, it is necessary to consider and report spatial variation of data quality.

This chapter mainly focuses on analysing conventional and existing approaches to reporting data quality. The chapter begins with a review of the literature. The existing approaches to representing SDQ are discussed in section 2.2. Building on section 2.2, section 2.3 discusses the architecture to represent SDQ. Section 2.4 reviews different storage types to store SDQ. Querying spatially varying quality including efficient retrieval mechanism is elaborated in section 2.5. Spatial variation models are discussed in section 2.6. The chapter is summarized in section 2.7.

2.2 Representation

Spatial data quality has been an important area of research in the topic of geographic information science (GISc) for many years. This section reviews the main GISc approaches to representing SDQ.
CHAPTER 2. BACKGROUND

2.2.1 Elements of spatial data quality

The overriding majority of work into SDQ takes as its starting point the five elements of spatial data: lineage, positional accuracy, attribute accuracy, logical consistency and completeness.

1. Lineage describes the history of a data set (Clarke and Clark, 1995).

2. Positional accuracy concerns the closeness of a feature’s “true” position to that stored in the database (Drummond, 1995).

3. Attribute accuracy is about the closeness of the “true” value for a qualitative or quantitative attribute to that observed or stored in the database (Goodchild, 1995).

4. Completeness describes whether the objects within a data set represent all entity instances of the abstract universe (Brassel, 1995).

5. Logical consistency deals with the logical rules of structure and attribute rules for spatial data and describes the compatibility of a data item with other data items in a data set (Kainz, 1995). Logical consistency is often separated into a) database consistency and b) topological consistency.

Although the five elements of SDQ form the foundations for data quality representations, one criticism of this representation is that it is not exhaustive. There are many other data quality elements that have been suggested in the literature.

1. Semantic accuracy refers to the pertinence of the meaning of the geographical object, rather than to the geometrical representation (Salge, 1995). Semantic accuracy encompasses concepts such as completeness (errors of omission and commission), consistency (validation of semantic constraints), currency (changes through time), and attribute accuracy.

2. Source and usage are often preferred to lineage as two distinct elements of data quality describing where the data came from, and what previous usages it has been put to (Aalders, 2002).

3. Temporal accuracy is the agreement between the encoded and “actual” temporal coordinates (Veregin, 1999). A land use data set can be temporally accurate but still out of date.
4. Currency is how “up-to-date” data is. Currency may depend on the application and is distinct from temporal accuracy (Worboys and Duckham, 2004).

5. Precision concerns the level of detail of data (Worboys and Duckham, 2004). Similarly, Goodchild and Proctor (1997) suggest using detail as an element of SDQ. Precision is closely related to granularity, the existence of clumps or grains within data. The individual elements in the grain cannot be distinguished or discerned apart (Worboys and Duckham, 2004). For example, imprecision in the satellite imagery of a forest would lead to granularity, where individual trees could not be discerned apart.

6. Bias is the existence of systematic distortions within data, which might have been introduced deliberately (as a water mark), or as an unforeseen consequence of observational, data collection, or analysis technique (Worboys and Duckham, 2004).

7. Reliability is the trustworthiness of degree of confidence a user may have in data (Worboys and Duckham, 2004).

A further criticism is that the definition of some of the elements of SDQ may not be mutually exclusive, in the sense that some elements overlap, and that it is not always possible to distinguish between different elements. For example, for some data it may be unclear whether an attribute has been recorded incorrectly at the correct location (attribute accuracy) or correctly recorded at the wrong location (positional accuracy) (Chrisman, 1987).

2.2.2 Other representations of SDQ

Not all representations of data quality are founded on the different SDQ elements. Fisher (1999) introduction to uncertain viewshed (probabilistic and fuzzy viewshed) is one example of other representations of SDQ. Other examples may include the literature of (Worboys and Duckham, 2004) which defines a typology of imperfection that comprises inaccuracy, which concerns a lack of correctness in information; imprecision, which concerns a lack of specificity or detail in information; and vagueness, which concerns the existence of boundary cases in information. Inaccuracy and imprecision are orthogonal: the statement “Melbourne is in Queensland” is not accurate. At the same time the statement “Melbourne is in Australia” is less precise (provides less detail about Melbourne’s location), but is still entirely accurate. Vagueness is a special type of imprecision that occurs in many spatial predicates and relations that exhibit indeterminate boundaries or
CHAPTER 2. BACKGROUND

borderline cases, such as “mountain” and “near”. Many of the elements of SDQ can be mapped to categories in the typology of imperfection. However, some cannot (e.g., lineage), and conversely vagueness has no direct correspondence in the elements of SDQ.

Price and Shanks (2004) have defined another representation of data quality based on semiotics. Here, data quality elements are classified into three levels: syntactic (deals with the structure of data, e.g., confirming to data integrity rules), semantic (deals with the meaning of quality, e.g., fidelity of data to the external world), and pragmatic (deals with suitability of use, e.g., pertaining to the delivery or importance of data). This approach is appealing since unlike the elements of SDQ it has a theoretical basis founded in semiotics. However, this work has been developed for data quality in general, rather than SDQ specifically. Although it appears that all elements of SDQ can be mapped to the semiotic framework, to date no work has been done to clearly study and define this relationship.

2.2.3 Standard organisations on data quality

Despite their limitations, the elements of SDQ have gained widespread acceptance over the years (Harold, 1997). The standard organisations including the US Federal Geographic Data Committee (FGDC), spatial data transfer specification (SDTS) (National Committee for Digital Cartographic Data Standards, 1988) and the ISO geographic information metadata standard (ISO 19115, 2003) are the basis for standardising data quality elements. In general, standard organisations specify that the data quality to be reported at various levels as: (1) database; (2) coverage; (3) tile; and (4) individual features. Chan (1999) has criticized the standards for being very restrictive when dealing with geographic primitives (sub-features) to store data quality. The ISO 19115 (2003) recommends quality to be stored as additional attributes for each feature, ignoring spatial variation within a feature (sub-feature variation). The issue facing most SDQ standards is that they are primarily aimed at data producers (data quality specifications) rather than data users (assessment of fitness for use) (Frank, 1998).

Irrespective of the standard five elements (lineage, positional accuracy, attribute accuracy, completeness and logical consistency) or other SDQ elements (semantic accuracy, source and usage, temporal accuracy, currency, precision, bias and reliability) or other representations of SDQ (inaccuracy, imprecision and vagueness), the thesis provides a platform to model any SDQ element or representation. Moreover, the thesis focuses on Chan’s (1999) criticism of representing sub-feature variation which was ignored by standards organisations.
2.3 Architecture

Having reviewed the main representations of SDQ, we now turn to computing with these representations. Although, on the fly calculation is one of the processes to generate data quality information, the thesis focuses on development of a data model to support spatial variation of data quality rather than investigating processes used to generate data quality information. From a spatial database perspective, we can think of computation with SDQ as comprising three key elements: storage, updating, and querying. Note that querying is required for other computational processes, such as visualization, since the data upon which a visualization is based must be efficiently retrieved from the spatial data store. As discussed in the first chapter, the thesis considers modification (updating) as an integral part of storage mechanism. Hence, there is no separate discussion made on the modification mechanism in the thesis. There are two schools of thought in storage of SDQ: one is to represent it as metadata (data about data) that resides outside the spatial data set to which the data quality refers, termed here a hybrid quality system; the other is to represent SDQ in with the spatial data set itself, termed an integrated quality system.

2.3.1 Hybrid quality system

Today, most spatial data providers document the quality of their data sets using a hybrid quality system, with metadata stored outside the spatial data set. Frank et al. (2004) discuss a computational model for the use of metadata describing the quality of geographic data to assess usability or fitness for use. Aalders (2002) proposes a design of quality registration in GIS using user-defined data sets in a relational database. Devillers et al. (2005) proposes a hybrid Quality Information Management Model (QIMM) to visualise and manage data quality. Devillers et al. (2005) adapted a non-spatial database technique called OLAP (online analytical processing) to store and visualise SDQ (SOLAP: spatial OLAP). The advantage of the hybrid approach is that it is easier to create and manage metadata. For example to host the metadata for a spatial data set on the web, all that we have to do is to upload the XML file containing metadata information. Updating of metadata is done by modifying the contents of the XML, which is relatively easy. Although, the QIMM, may represent spatial variation as the model is independent of the data set level. Most of the hybrid approaches discussed here do not fully represent spatial variation of quality as most of the quality data are reported at the global level. Moreover, management of links between the metadata and
CHAPTER 2. BACKGROUND

spatial data becomes more complex when more spatial data are added to the database.

2.3.2 Integrated quality system

In contrast, Buttenfield (1993) suggests that quality information should be stored in the data set itself (integrated approach) and should be linked to data elements at a level of specificity that allows user access and compensation. Guptill (1989) argues that data quality information derived during the process of data processing can be stored as part of the spatial database. Qiu and Hunter (2002) take advantage of an integrated approach in order to store data quality information at multiple-levels within a hierarchical structure, using object oriented techniques. The advantage of this approach is that, when an element in a data set is modified, the quality information is also automatically modified. Using a hybrid approach, the metadata has to be updated separately when changes occur on the data set. Similarly, Gan and Shi (2002) have developed an Error Metadata Management System (EMMS) to represent quality at feature level. Duckham (2001) provides a formal model of an integrated data quality system based on object-orientation. All these approaches provide capabilities to store, update, and query spatial variation in databases.

Overall, the hybrid and integrated approaches provide different storage, querying, and updating capabilities, but more research needs to be done to rigorously categorize the respective advantages and disadvantages. None of the existing research outlined in this section can claim to be able to efficiently represent spatially varying SDQ across a range of spatial data types. One of the critical questions that is not adequately addressed by these models is the representation of sub-feature variation (variation in quality within a geographic feature).

2.4 Database concepts

The thesis uses SQL (Structured Query Language) to create and query tables in a database. SQL is the leading international standard database language also recognised by International Organization for Standardization (1999). Connolly et al. (1999) have defined the following database terminologies, which will be used in the rest of the thesis.

- A relation is a table with columns and row.
- An attribute is a named column of a relation.
A domain is the set of allowable values for one or more attributes.

A tuple is a row of a relation.

The degree of a relation is the number of attributes it contains.

The cardinality of a relation is the number of tuples it contains.

A database is a collection of relations (tables).

For example, a relation scheme for a location relation is given in the form as:

\[
\text{location (id: number, x: number, y: number).}
\]

The location is the name of a relation. The id, x and y are attributes of the relation location. The attribute id is an unique identifier (primary key) and has a number data type. The attribute x and y also have a number data type. The conceptual schema for the relation location is shown in the table 2.1.

<table>
<thead>
<tr>
<th>ID</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31.25</td>
<td>141.25</td>
</tr>
<tr>
<td>2</td>
<td>31.45</td>
<td>141.50</td>
</tr>
<tr>
<td>3</td>
<td>31.85</td>
<td>141.75</td>
</tr>
<tr>
<td>4</td>
<td>31.90</td>
<td>141.93</td>
</tr>
</tbody>
</table>

Table 2.1: Location

To retrieve the x and y coordinates of location id = “3” the following SQL query structure is adopted:

Sample query 1

\[
\begin{align*}
\text{SELECT } x, y \\
\text{FROM location} \\
\text{WHERE id = 3}
\end{align*}
\]

Table 2.2: Output of sample query 1

To retrieve all information associated with location id = “3” the following SQL query structure is adopted:
Sample query 2

```
SELECT *
    FROM location
    WHERE id = 3
```

Table 2.3: Output of sample query 2

```
<table>
<thead>
<tr>
<th>ID</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>31.85</td>
<td>141.75</td>
</tr>
</tbody>
</table>
```

The **SELECT** keyword specifies which columns are to appear in the output like sample query 1. The asterisk * refers to all columns in the table like sample query 2. The **FROM** keyword specifies the table to be used. The **WHERE** clause filters the result subject to some condition. The sample outputs of the **SELECT** statement are shown in the table 2.2 and 2.3. The other optional keywords used after the **WHERE** clause are **GROUP BY** (forms group of rows with the same column value), **HAVING** (filters the group subject to some condition) and **ORDER BY** (specifies the order of the output) (Connolly et al., 1999).

### 2.4.1 Relational algebra

The relational algebra is a formal system for manipulating relations, and provides the formal basis for SQL. Thus, relational algebra is used in the development of the formal model in chapter 4. There are five fundamental relational operators: **union**, **difference**, **product**, **project** and **restrict**. Three further relational operators **intersection**, **divide** and **join** termed **derived** relational operators, can be expressed using different combinations of the fundamental five operators. All operators take one or two relations as input and return a single relation as a result.

**Project:** The project operator is a unary operator and is represented by the symbol $\pi$. The results of the $\pi$ operation contain a new relation that has a subset of attributes of the input relation. The syntax is $\pi_{\text{attributes}}(\text{relation})$.

**Restrict:** The restrict operator is a unary operator and is represented by the symbol $\sigma$. The results of the $\sigma$ operation returns a new relation consisting of all tuples from a specified relation that meet a specified condition. The syntax is $\sigma_{\text{condition}}(\text{relation})$. 
Renaming: It is sometimes necessary to rename attributes using the renaming relational operator \( \rho \). The rename operator takes as an input a relation and returns the same relation. The expression \( \rho_{y/x}(a) \) produces an identical relation to \( a \), except with attribute \( x \) renamed as \( y \).

Product: The product operator is a binary operator and is represented by the symbol \( \times \). The product operation multiplies two relations to define another relation consisting of all possible pairs of tuples (see section 3.3) from the two relation. Consider two relations \( R \) and \( S \). The relational expression \( R \times S \) produces one tuple for each combination of tuples in \( R \) and \( S \).

Union: The union operator is a binary operator and is represented by the symbol \( \cup \). Union is possible only if the schema of the two relations match, that is, if they have the same number of attributes with matching domain in other words, the relations must be union compatible. Projection operation may be used in some cases to make two relations union compatible. Consider two relations \( R \) and \( S \). The relational expression, \( \pi_{id}(R) \cup \pi_{id}(S) \) retrieves the set of tuples that are in \( R \) or \( S \).

Intersection: The binary operator intersection \( \cap \) is a derived operator. Consider two relations \( R \) and \( S \). The expression \( R \cap S \) produces tuples that are in both \( R \) and \( S \). Like the union operator the intersection operator requires the relations \( R \) and \( S \) to be union compatible.

Join: The join is a binary operator and is represented by the symbol \( \bowtie \). The joins are classified as natural and spatial join.

Natural join: The product of two relations \( R \) and \( S \) is obtained by pairing every tuple from \( R \) relation with every tuple from \( S \). Intuitively, the join operator \( \bowtie \) restricts such pairings to those that satisfy some predicate (\( R.id = S.id \)), called join predicate. The join operator with its join predicate is represented as: \( \bowtie_{\text{att1} = \text{att2}} \). If the join predicates are non-geometric, then the join is called a natural join.

Spatial join: An object of one theme is joined with an object of the other theme if their geometries intersect. The resulting object has the descriptive attributes of both participating objects and its geometry is the intersection of the participating objects’ geometry (Rigaux et al., 2002b). Spatial join is used to combine two or more data sets with respect to a spatial predicate. Predicate can
be a combination of directional, distance, and topological spatial relations as: inside; contains; touch; covered by; covers; equal; on; overlap boundary intersect; and overlap boundary disjoint (9 intersection model) (Egenhofer and Franzosa, 1991). The spatial join is represented as: $R_{\text{geometry}} \cap S_{\text{geometry}}$.

**Example of mapping relational algebra expression to SQL:** To retrieve the $x$ and $y$ coordinates of location which has id < “2”, the relational algebra expression is:

$$\pi_{x,y} (\sigma_{\text{id}<2} (\text{location})).$$

The SQL query structure for the above relational expression is:

```sql
SELECT x, y
FROM location
WHERE id < 2
```

### 2.5 Storage

Storage models can be classified according to whether they are based primarily on object-based or field-based representations. An object-based model treats the space as populated by discrete, identifiable entities with a geospatial reference (Worboys and Duckham, 2004). A field-based model treats geographic information as a collection of spatial distributions. Each distribution may be formalized as a function from a spatial framework to an attribute domain (Worboys and Duckham, 2004). A few authors have carried out work aiming to store spatial variation in SDQ, including: reliability diagrams (object-based), object-oriented techniques (object-based), co-ordinate based quality (object-based), variability diagram (field-based), data quality matrix (field-based) and quadtree (field-based).

**Reliability diagrams (object-based):** Reliability diagrams can be classified as hardcopy and digital diagram.

**Hardcopy reliability diagram:** Representing spatial variation in SDQ is not a new concept. Cartographers and geologists have used reliability diagrams as quality indicators to communicate spatial variation in SDQ (Hunter, 2005). Cartographers have used symbols to depict uncertain contours as reliability diagrams (Fisher, 1991). Geologists have also used symbols to represent variation in rock fault with the help of reliability diagrams (Hunter, 2005). However, the standard reliability diagrams have limitations such as the ability to reflect the
complexities of error prorogation nor the associated probabilities of an overlay surface (National Center for Geographic Information and Analysis, 1989).

**Digital reliability diagram:** National Committee for Digital Cartographic Data Standards (1988) had proposed that the variation in positional accuracy shall be reported either as additional attributes of each feature or through a quality overlay (reliability diagram). The United States Department of the Interior Bureau of Land Management have developed a database called the Geographic Coordinate Data Base (GCDB) based on the NCDCDS (1988) recommendations. The main objective of the database was to communicate the reliability of the features by means of reliability diagrams (United States Department of the Interior Bureau of Land Management, 2001). The approach illustrates that spatial variation (see figure 2.1 and table 2.4) can be represented by storing an additional attribute called reliability value in the attribute table. However, the approach lacks the ability to show variation within individual features.

![Digital reliability diagram](image)

Figure 2.1: Digital reliability diagram to indicate reliable features

<table>
<thead>
<tr>
<th>Highest Reliability Value in Feet</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Black or White</td>
</tr>
<tr>
<td>1 - 3</td>
<td>Blue</td>
</tr>
<tr>
<td>4 - 40</td>
<td>Purple</td>
</tr>
<tr>
<td>41 - 200</td>
<td>Green</td>
</tr>
<tr>
<td>Over 200</td>
<td>Red</td>
</tr>
</tbody>
</table>

Table 2.4: Digital reliability diagram legend
(Source including original symbolization: United States Department of the Interior Bureau of Land Management)
Variability diagram (field-based): Soil scientists have used variability diagram to depict the variation of homogeneity in soils (Maclean et al., 1993). The variability diagrams were generated in GIS using Shannon’s measure of entropy (Shannon entropy or information entropy is a measure of the uncertainty associated with a random variable) (Shannon and Weaver, 1949). Variability diagrams coupled with digitized soil maps inform users of the degree of soil map unit variability and the variability of limiting soil properties. As variability diagrams are stored as a separate layer in the database, overlay operations are required each time the information is retrieved.

Object-oriented technique (object-based): Qiu and Hunter (2002) have stored data quality information at multiple-levels within a hierarchical structure, using object-oriented techniques. Qiu’s method of storing quality at feature level represents spatial variation but it is limited in its representational capabilities, since it cannot represent variation in quality within an object. Duckham (2001) addresses the problem of representing variation within an object by using object calculus, which is a theoretical approach. Object calculus does not represent a database in full and fails to address the querying mechanism to retrieve spatial variation of SDQ.

Coordinate based quality (object-based): One could simply think to associate quality information to each geographic coordinate (Faiz and Boursier, 1995). The 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. Navratil et al. (2004) and (Dutton, 1991) after implementation of measurement based GIS have also concluded that storing quality information at coordinate level makes the database voluminous and takes more time to compute. One might argue that storage and computing cost are affordable. However, irrespective of storage and computing time, managing voluminous data quality information alongside the geometric features is another additional overhead associated with managing coordinate based quality.

Data quality matrix (field-based): Hetrick (1991) proposes the development of a data quality matrix for raster data in which information can be coded and stored to represent missing data. The matrix can be visually inspected or automatically queried by analysts. The data quality matrix approach works for raster
data to represent completeness (errors of omission or errors of commission), an element of SDQ. As a result of storing SDQ in each pixel, the approach increases the volume of the stored data significantly.

**Quadtree (field-based):** Beard et al. (1991) have discussed the possibility of storing spatially variable data quality information in a manner similar to the quadtree method. The quadtree works on the principle of recursively subdividing the cells into quarters (Worboys and Duckham, 2004) so that each quadrant has an homogeneous level of quality. The quadtree method is widely used for spatial database indexing. The limitations of the method are that each time a feature is updated the corresponding structure of the quadtree also has to be modified. Thus, data sets which need frequent updating are not suited to using the quadtree method. Moreover, if the features in the raster are non-areal, like points and lines then the quadtree has less efficiency in storage. Any linear feature (including polygon boundary) is also inefficient.

### 2.6 Querying spatial variation

Querying is the process of retrieving stored information from the database. The retrieval process mainly depends upon the data storage mechanism. Generally queries are classified as *non-spatial* and *spatial queries*.

#### 2.6.1 Non-spatial query

A non-spatial query retrieves information based on non-geometric attributes. A point query finds the feature, which is associated with a particular attribute value. A range query searches for features whose attribute values fall within a particular range. Many SDQ queries can be considered non-spatial, for example:

- “What positional accuracy information is associated with parcel id = 5?” (point query)
- “List the precision associated with polygon vertices of parcels between 1m and 2.5m?” (range query)

#### 2.6.2 Spatial query

A query, such as “Show the distance between two points,” deals with geometries (spatial data). A spatial query is defined as one that manipulates location data.
in addition to attribute data (Adam and Gangopadhyay, 1998). In simple terms a spatial query is a search for the stored data to satisfy a given spatial condition. Examples of spatial queries are “Where is Melbourne University”, “Display the departments which are within a distance of 300 meters from the Geomatics department” and “Show the shortest route from the university to the train station”. Shekar and Chawla (2003) have classified spatial queries as: point (find all features that intersect a particular point), range (find all features that intersect a particular region), nearest neighbour (find the geometries residing close to a given feature), and spatial join queries (involves retrieving information based on the spatial relationships between features in two or more relations). Some examples of spatial queries in SDQ include:

- “What is the positional accuracy at coordinate (x, y)?” (spatial point query).
- “List the accuracy of the region within 50m of parcels with accuracy 10m” (spatial range query).
- “List two nearest accuracies to parcels with 0.5m positional accuracy” (nearest neighbour query).
- “List vertical and horizontal positional accuracies of parcels” (query with spatial join).

### 2.6.3 Query optimization

Spatial queries involving spatial joins are computationally expensive. Hence, we need to perform an optimization step to reduce the computation time. The database literature recommends the following filter and refine paradigm for optimization.

- Filter: use spatial index approximations to efficiently exclude objects that definitely do not satisfy query criteria.
- Refine: with filtered object set, perform computationally intensive spatial processing to provide exact answer to query.

Based on efficient data organisation including indexing data and effective query plan, result in faster data retrieval. An efficient and effective query mechanism is essential for retrieving spatially varying data quality. The thesis had adopted the query optimization step in retrieving spatially varying quality.
2.7 Spatial variation data models

Heuvelink (1996) uses the idea of spatial quality fields to generate a range of different models of spatial variation in attribute error. The DMSV (discrete model spatial variation) assumes that major jumps in attribute values occur at the boundaries of homogeneous mapping units; the CMSV (continuous model of spatial variation) assumes that attribute values change gradually in space. A third model, the MMSV (mixed model of spatial variation) is a mixture of both and includes gradual as well as abrupt changes. Heuvelink (1996) concludes that object-oriented vector data structures are often not well suited to deal with continuous spatial variation and raster GISs are more flexible in this respect and can more easily handle all three types of spatial variation.

Davis et al. (2001) use error models to quantify and predict the spatial variability of DEM (digital elevation model) error. Developing efficient models of variation in SDQ is an important and challenging topic. A few researchers have begun to address the issue of storing spatial variation in SDQ. Hunter and Qiu (2003) discuss the problem of recording of local variation in the quality of features and attributes. They suggest including additional attributes to carry this information or to adopt object-oriented techniques. However, the former approach is difficult as it tends to lead to alterations in the structure of the data. The latter approach is conceptually appealing, but suffers practical difficulties given that the relational geodatabase model remains the dominant spatial database architecture. Ramlal and Beard (1996) propose a strategy called mixed variation model to integrate data sets with varying quality. Wong and Wu (1996) suggest that, in addition to metadata for spatial data, it is necessary to develop spatial metadata that can reflect the geographical differences in data quality. Wong and Wu (1996) conclude that problems such as the storage issue, the structure of spatial database, and the dynamic query of SDQ information still need to be addressed.

2.8 Summary

Spatial variation in SDQ is the most important, challenging and under researched type of variation in SDQ (Wong and Wu, 1996). Although a few models and some limited storage capabilities exist for spatial variation in SDQ, in general it is not yet possible to fully represent or compute with spatially varying SDQ. The elements of SDQ are the most important representation of SDQ, but not the only one. Clearly, any computational approach to variation in SDQ must support the elements of SDQ found in common and emerging SDQ standards.
The literature review illustrates the following key points:

- Spatial variation is an important aspect of SDQ that has not yet been adequately researched.
- Standards organisations have considered spatial variation aspect of data quality excluding sub-feature variation in their standards. The new version of standards may consider including sub-feature variation in SDQ.
- From a spatial database perspective, the key issues that require resolution are the storage and querying of spatially varying SDQ.

Ultimately, resolving these issues and developing a model to represent spatial variation in SDQ will help the producers of spatial data to communicate the quality of their data more effectively, so enabling the users of spatial data to more effectively assess fitness for use.
Chapter 3

Models of spatial variation in data quality

3.1 Introduction

This chapter discusses the related data quality models of spatial variation in databases: per-feature, feature-independent and feature-hybrid. The chapter starts with an introduction of the three data quality models. The prototypes I and II were developed to implement the models. Prototype I focuses on implementing the models using conventional GIS software and uses linear referencing (LR) approach. The advantages and disadvantages of using LR approach in implementing the models were analysed. The analysis resulted in the development of the Prototype II. The Prototype II adopts a relational database approach for the implementation. Oracle Spatial RDBMS was used to implement the models. The Prototype II section is sub-divided into storage and query sub-sections. The storage subsections discuss on each model’s storage characteristics whereas the query sub-section illustrates the behaviour of each model’s performance on spatial variation queries. At the end of the chapter, a brief summary and synthesis of the Prototype I and Prototype II implementation is presented.

3.2 Related data models

Spatial variation models in a database can be classified according to whether they are based primarily on object-based or field-based representations. An object-based model treats the space as populated by discrete, identifiable entities with a geospatial reference. A field-based model treats geographic information as collections of fields, each field defines the spatial variation of an attribute as a function from the set of locations to an attribute domain (Worboys and Duckham, 2004).
Given that modeling spatial variation in spatial data quality is an important unsolved problem, it is possible to suggest some initial options based on the literature for solving this problem. Therefore, three following options have been identified as:

1. Per-feature
2. Feature-independent
3. Feature-hybrid

### 3.2.1 Per-feature model

First, models of thematic variation do provide limited capabilities for modeling spatial variation, termed here the *per-feature model*. The spatial data quality of a feature can be stored along with the feature as an additional attribute as for many of the object-based models of spatial variation in data quality discussed in chapter 2 (e.g., Hunter and Qiu, 2003; United States Department of the Interior Bureau of Land Management, 2001). Per-feature (object based) quality can be modeled as a function $f : O \rightarrow Q$ where $O$ is a set of objects (features) and $Q$ is a quality codomain. Features in $O$ can include points, lines and polygons. The quality codomain $Q$ might comprise information about any of the elements of spatial data quality (e.g., $Q$ might represent the set of all positional accuracy values for features in a data set). Because each feature maps to a single quality value, the per-feature model cannot represent variation within a feature (sub-feature variation). Figure 3.1 illustrates an example of a per-feature model.

### 3.2.2 Feature-independent model

The second option is to store spatial variation in spatial data quality as a separate theme or layer in a spatial database, as for many of the field-based models of spatial variation in data quality discussed in chapter 2 (e.g., Maclean et al., 1993; Heuvelink, 1996), termed here the *feature-independent model*. Feature-independent (field-based) quality can be modeled as a function $g : S \rightarrow Q$, where $S$ is the spatial framework and $Q$ is the quality codomain. Spatial objects, such as lines and polygons are not explicitly represented in this model. Instead, data quality is represented as a field, independent of features in the data set. However, while the feature-independent model is conceptually appealing and helps in modeling sub-feature variation more research needs to be directed at the computational implications of the feature-independent model. In particular, querying the feature-independent model is potentially computationally expensive as it re-
3.2. RELATED DATA MODELS

Figure 3.1: Example of Per-feature model

requires repeated spatial joins (overlays) to determine the quality of an individual feature. The model may require additional storage space to store quality information. Figure 3.2 illustrates an example of feature-independent model.

3.2.3 Feature-hybrid model

Finally, the third option, termed the feature-hybrid model, is to store quality information on a per-feature basis, but augment the stored quality with some additional spatial structure. Feature-hybrid (object-and field-based) quality can be modeled as a function \( h : O \rightarrow Q^S \) where \( Q^S \) denotes the set of functions (fields) from \( S \) to \( Q \). Although there might be different ways of designing the feature-hybrid model, the thesis investigates only two options to design the feature-hybrid model. The first option is to use dynamic segmentation (Cadkin, 2002) which applies to linear features only, otherwise called as Linear Referencing (LR) technique. Linear referencing is widely used in transportation field. Another option is to integrate (overlay) the object’s geometry with quality’s geometry. This approach can be termed as overlay approach. Figure 3.3 is an example of a feature-hybrid model created by an overlay approach. By adopting this approach sub-feature variation can be represented across both polygon and linear features.
CHAPTER 3. MODELS OF SPATIAL VARIATION IN DATA QUALITY

Figure 3.2: Example of Feature-independent model

Figure 3.3: Example of Feature-hybrid model
3.3 Prototype I

As discussed earlier in the first chapter (see section 1.4.1), the thesis pays particular attention toward line and polygon data structure in the implementation.

3.3.1 Scenario

Consider a highway that was constructed in stages. Data about the highway in the database may have varying positional accuracy at each stage of its construction. The assumption made here is that the highway has three varying positional accuracies of 1m, 3m and 5m across four segments. In this example, the first and third segments have 1m accuracy, the second segment has 3m accuracy, and the fourth segment has 5m accuracy. To represent the spatial variation of positional accuracy in the highway three models discussed in the above section, per-feature, feature-independent and feature-hybrid were tested.

3.3.2 Per-feature implementation

In the per-feature model the positional accuracy of the highway can be stored as an additional attribute called “Positional” (Figure 3.4) in the highway (sv-pf attribute table) table. As per the scenario, the single record of highway feature has to be segmented into four records to store positional accuracy of each segment which will represent spatial variation. To retrieve all the features with a positional accuracy of between 1 and 4m, for example, the following SQL statement is used:

```
SELECT *
FROM   sv-pf
WHERE  positional ≥ 1 AND positional ≤ 4
```

Spatial variation of quality can be represented in polygon features in the same way. For example, the land parcel in Figure 3.5 has two attribute accuracy values as 1% and 4%. The only way to record this sub-feature variation of attribute accuracy is to again split the feature into two records (see Figure 3.5, attributes of parcel second table). To retrieve the features with attribute accuracy greater than 1%, for example the following SQL statement is used:

```
SELECT *
FROM   parcel
WHERE  attribute > 1
```
CHAPTER 3. MODELS OF SPATIAL VARIATION IN DATA QUALITY

Figure 3.4: Per-feature model for linear features

Figure 3.5: Per-feature model for polygon features
In summary, in the per-feature model the data structure must be altered for the sake of storing sub-feature variation in quality information. For example, the highway is segmented based on the positional accuracy information. In turn one record is split into four records. Changing the data structure in this way causes fragmentation of the data set and leads to increase in the volume and complexity of the database.

3.3.3 Feature-independent implementation

In the feature-independent model, the segmentation of the highway is avoided by storing the data quality as a separate layer (see Figure 3.6). The positional accuracy layer is independent of the highway layer. However, to retrieve the spatial variation of positional accuracy in the highway feature an additional operation of a spatial join (overlay) is required. Spatial joins involve retrieving tuples from two or more relations based on a spatial join condition. Spatial joins are amongst the most computationally expensive operations in a spatial database. In this case, to retrieve the features with a positional accuracy of 3m or less, the corresponding SQL statement will be used:

SELECT highway.id, positional 
FROM highway, sv-fi 
WHERE highway.geom ∩ sv-fi.geom AND positional ≤ 3

The feature-independent model can equally model the quality of polygonal features. For example, consider a land parcel having spatially varying accuracy. The accuracy is stored as a separate layer, which is independent of parcel (see Figure 3.7). In order to retrieve the quality information a spatial join operation is carried out on the parcel and accuracy layers. Querying an individual feature in the feature-independent model requires repeated spatial joins to determine the quality of an individual feature. Additional storage space (space complexity) and increase in processing time (time complexity) are therefore limitations of this model when compared with the per-feature-model.

3.3.4 Feature-hybrid implementation

Feature hybrid model 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
CHAPTER 3. MODELS OF SPATIAL VARIATION IN DATA QUALITY

Figure 3.6: Feature-independent model for linear features

Figure 3.7: Feature-independent model for polygon features
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. One technique for implementing the feature-hybrid model is to use linear referencing (Cadkin, 2002) to enable spatially varying quality information to be stored on the feature without altering the feature. Linear referencing is the method of storing geographic locations by using relative positions along a measured linear feature. Distance measures are used to locate events along the line. Each linear feature is assumed to have a unique identifier. In addition to the feature attribute table, an additional table, called an event table (see Figure 3.8), is used to store the quality values. Using linear referencing, modification of quality information becomes computationally straightforward, since updates only need to alter the event table. To retrieve the spatial variation of positional accuracy across the highway, the event table is queried with a similar SQL structure to that used in the per-feature model. However, using linear referencing is only one example of a feature-hybrid model.

Figure 3.8: Feature-Hybrid model for linear feature

Linear referencing offers limited representational capabilities with respect to polygon features. For example, the technique can be used to represent spatial variation in the quality of the boundaries of polygons (cyclic linear features) (see Figure 3.9), but cannot be used to represent spatial variation across the interior of a polygon. It is proposed to use overlay approach as an alternate to linear referencing to represent spatial variation across the interior of a polygon.
In the feature-hybrid model, if the features are modified the event table has to be restructured accordingly. There are inevitably computational overheads associated with managing the additional event tables.

3.3.5 Prototype I summary

The three models, per-feature model (object-based quality), feature-independent (field-based quality) and feature-hybrid (object- and field-based quality) represent spatial variation in data quality. The per-feature model is easy to implement but has limitations in representing sub-feature variation and alters the data structure for the sake of storing quality information. Additional storage space (space complexity) and increase in processing time (time complexity) are the limitations of the feature-independent model, in spite of the model’s ability to represent sub-feature variation. The feature-hybrid model integrates both per-feature and feature-independent models, and potentially overcomes the limitation of the previous models. However, using linear referencing as a technique for implementing the feature-hybrid model has limitations with respect to representing polygon features.
3.4 Prototype II

To overcome the limitations discussed in Prototype I implementation, the Prototype II examines the comparative processing requirements for data quality management through overlay operations, such as union or intersection of features. The spatial structure adopted in the overlay approach is that the object’s geometries were integrated with quality’s geometries by performing a simple overlay. By adopting this approach sub-feature variation was represented across both polygon and linear features (Sadiq and Duckham, 2007) which was the limitation of Prototype I.

Rather than propose new architectures for storing SDQ (e.g., Duckham 2001), Prototype II adopts a conventional spatial RDBMS (relational database management system) architecture. Oracle Spatial was used to implement the relational scheme for each of the three models of spatial variation in SDQ. To illustrate the different models of spatial variation in SDQ, a few parcels (9) pertaining to the Panton Hill suburb from the DSE’s (the Victorian Department of Sustainability and Environment) Vicmap Property database were taken as sample data for developing the second prototype. The parcels were represented as polygon features. The accuracy of the corresponding parcels were generated hypothetically to test the spatial variation models in section 3.2.1, 3.2.2 and 3.2.3.

3.4.1 Storage

Let the parcel layer be called as base (Figure 3.10) which has nine parcels of the Panton Hill locality. The base table which stores the spatial data has the following relation scheme:

base \((tid: \text{number}, \text{geom}: \text{geometry})\)

where \(tid\) is a primary key and \(\text{geom}\) is the geometry of the parcel features in the base table. The following subsections show how spatially varying SDQ information can be stored in the three models. Note that as discussed in section 3.2.1, the representational limitations of the per-feature model mean that there is no sub-feature variation in the per-feature model.

Per-feature relation scheme

The per-feature quality model (see section 3.2.1) has a relation scheme of the general form:

\(\text{pf} \ (\text{fid}: \text{int}, \text{quality}: \text{int}, \ldots)\)

where \(\text{fid}\) is a quality identifier and \(\text{quality}\) stores the quality information, which
relates qualities to features in a specific feature relation. In this example, the per-feature quality relation is implemented as table \texttt{pfq}(table 3.2), which stores corresponding quality information (accuracy in meters) for each parcel in the base table. The per-feature quality table \texttt{pfq} has the following relation scheme:

$$\texttt{pfq} (\texttt{fid}: \text{number}, \texttt{acc}: \text{number})$$

<table>
<thead>
<tr>
<th>FID</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.75</td>
</tr>
<tr>
<td>2</td>
<td>2.75</td>
</tr>
<tr>
<td>3</td>
<td>2.75</td>
</tr>
<tr>
<td>4</td>
<td>2.5</td>
</tr>
<tr>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td>6</td>
<td>2.5</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.1: PFQ (Per-Feature Quality table for sample parcels in Panton Hill Locality)

**Feature-independent relation scheme**

A feature-independent quality relation \texttt{fi} has the relation scheme of the general form:

$$\texttt{fi} (\texttt{qid}: \text{int}, \texttt{geom}: \text{geometry}, \texttt{quality}: \text{number, \ldots})$$

where \texttt{qid} is a quality identifier, \texttt{geom} is a geometry data type that describes the spatial characteristics of each quality feature and \texttt{quality} stores the quality infor-
mation (section 3.2.2). The feature-independent quality model in this example is implemented as table fiq (table 3.2, Figure 3.11). The quality information is stored along with the quality geometries independent of base’s (parcel) geometries. Thus, for the feature-independent quality table fiq the following relation scheme is used:

\[
\text{fiq (qid: number, geom: geometry, acc: number)}
\]

![Figure 3.11: FIQ (Feature-Independent Quality model for sample parcels in Panton Hill Locality)](image)

<table>
<thead>
<tr>
<th>QID</th>
<th>GEOMETRY</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MDSYS.SDO_GEOMETRY</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>MDSYS.SDO_GEOMETRY</td>
<td>2.5</td>
</tr>
<tr>
<td>3</td>
<td>MDSYS.SDO_GEOMETRY</td>
<td>2.75</td>
</tr>
<tr>
<td>4</td>
<td>MDSYS.SDO_GEOMETRY</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3.2: FIQ (Feature-Independent Quality table for sample parcels in Panton Hill Locality)

**Feature-hybrid relation scheme**

A feature-hybrid quality model (see section 3.2.3) has a relation scheme of the general form:

\[
\text{fh (qid: int, fid: int, geom: geometry, quality: int, ...)}
\]

where qid and fid are composite keys (quality identifiers), which relate qualities to features in a specific feature relation; geom is a geometry data type that describes the sub-feature spatial characteristics of each quality feature; and quality stores the SDQ information. The feature-hybrid quality model is implemented as table
fhq (table 3.3, Figure 3.12). The fhq table stores sub-feature quality information for each parcel in the base table. Thus, for the feature-independent quality table the following relation scheme is used:

```
fhq (qid: number, fid: number, geom: geometry, acc: number)
```

Note that in this case, the (hypothetical) quality information in the fhq table is derived from the fiq table (simply the intersection of the base and fiq tables). It need not always be the case that the fhq be derived from the fiq table: this is purely for illustrative purposes.

![Figure 3.12: FHQ (Feature-Hybrid Quality model for sample parcels in Panton Hill Locality)](image)

<table>
<thead>
<tr>
<th>QID</th>
<th>FID</th>
<th>GEOMETRY</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>MDSYS.SDO_GEOMETRY</td>
<td>2.5</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>MDSYS.SDO_GEOMETRY</td>
<td>2.5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>MDSYS.SDO_GEOMETRY</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>MDSYS.SDO_GEOMETRY</td>
<td>2.0</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>MDSYS.SDO_GEOMETRY</td>
<td>2.75</td>
</tr>
</tbody>
</table>

Table 3.3: FHQ (Feature-Hybrid Quality table for sample parcels in Panton Hill Locality)

Storage summary

To summarize:

- In the per-feature model, the quality is stored in an additional table. For each parcel of the base table a corresponding quality value was stored in
the quality table. The limitation of the model is that sub-feature variation cannot be represented. For example, the model cannot represent the case where different parts of parcel have different quality values.

• In the feature-independent model, the geometry of spatial variation in quality was stored independent of the parcel’s geometry. While this is simple, it can increase the storage and maintenance costs when compared to the per-feature model. The model does allow the representation of sub-feature variation when it is queried in conjunction with the base table.

• In the feature-hybrid model, the qualities represented in the model are associated with geometries that are independent, but contained within, the corresponding parcel’s geometry. To link the quality and parcel information, the feature id of each parcel is stored along with its quality geometries. As a result, each parcel may have more than one quality value, making the quality table capable of representing sub-feature variation.

3.4.2 Querying

The query process has been classified as non-spatial and spatial queries.

Non-spatial point query (Query 1)

“What is the accuracy at parcel 5?”

Per-feature model: To retrieve the quality information of parcel number 5 stored in the per-feature model involves two steps. The first step is to perform a natural join (see chapter 2, section 2.5.3) on relation base and pfq as:

\[ \text{base} \bowtie \text{id} = \text{id} \text{pfq} \]

The second step is to select parcel 5 (\(a.fid = 5\)) from the join results.

Query 1.1

\[
\begin{align*}
\text{SELECT} & \quad a.fid, b.acc \\
\text{FROM} & \quad \text{base a, pfq b} \\
\text{WHERE} & \quad a.fid = b.qid \text{ AND } a.fid = 5;
\end{align*}
\]

Results: The results of query 1.1 retrieves accuracy of 2.5m for the parcel number 5 upon performing a natural join between base and pfq tables.
Feature-independent model: The non-spatial point query on feature-independent model requires a spatial join (see chapter 2, section 2.5.3) to be performed on relation base and fiq as:

\[
\text{base} \bowtie \text{geom} \cap \text{geom fiq}
\]

The selection step \((\text{a.fid} = 5)\) was performed after the join conditioned.

Query 1.2

```
SELECT a.fid, b.acc
FROM base a, fiq b
WHERE a.geom \cap b.geom AND a.fid = 5;
```

Results: The results of query 1.2 denote that the parcel \text{fid} = 5 has two accuracy values (2m and 2.75m) within the feature. Thus, the feature-independent model represents sub-feature variation. The spatial join needed to find the spatial intersection of records in \text{base} and \text{fiq} is computationally expensive when compared to the natural join performed in the query 1.1.

<table>
<thead>
<tr>
<th>FID</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Feature-hybrid model: The non-spatial point query for the feature-hybrid model requires a natural join \(\text{base} \bowtie \text{fid} = \text{fid fhq}\) to retrieve accuracy of parcel number 5.

Query 1.3

```
SELECT a.fid, b.acc
FROM base a, fhq b
WHERE a.fid = b.fid AND a.fid = 5;
```
Results: Note the result of the feature-hybrid query 1.3 and feature-independent query 1.2 are same. The feature-hybrid query is more efficient than the feature-independent query because the feature-hybrid query uses a natural join (computationally inexpensive) when compared to feature-independent query, which uses a spatial join (computationally expensive).

<table>
<thead>
<tr>
<th>FID</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2.75</td>
</tr>
</tbody>
</table>

Spatial point query (Query 2)

The spatial point query is an interactive query which is usually performed by clicking the mouse at a particular point on the display in conventional GIS softwares. In Oracle spatial, the working behind the interactive query process is to first create a separate table poi to store x and y as point geometry. Depending on the quality model, a natural or a spatial join is performed between the base and quality (pfq, fiq and fhq) table to retrieve its corresponding accuracy. Although the output of the query retrieves quality information for an unknown area, the query lacks capability in retrieving spatial variation.

“What is the accuracy at coordinate x, y?”

Spatial point query on per-feature model: In the per-feature model, a natural base \( \times_{\text{fid} = \text{fid}} \text{pfq} \) is performed to retrieve the accuracy of each parcel. In the next step the x, y value is retrieved using a spatial join between the geometries of the point (x, y) and the results of the natural join.

Query 2.1

```
SELECT a.fid, b.acc
FROM
    (SELECT i.geom, j.acc
     FROM base i, pfq j
     WHERE i.fid = j.qid) a, poi b
WHERE a.geom \cap b.geom;
```
Results: The spatial point query performed on the per-feature model requires both spatial join and a natural join to retrieve quality information.

\[
\text{ACC} \quad ------ \quad 2.75
\]

Spatial point query on feature-independent model: Querying quality of a point using the feature-independent model involves two spatial joins as:

\[
(\text{base} \times_{\text{geom} \cap \text{geom} \text{fiq}}) \times_{\text{geom} \cap \text{geom} \text{poi}}
\]

Query 2.2

\begin{verbatim}
SELECT a.acc
FROM
    SELECT i.geom \cap j.geom, j.acc
    FROM base i, fiq j a, poi b
WHERE a.geom \cap b.geom;
\end{verbatim}

Results: The spatial point query performed on the feature-independent model requires two spatial joins to retrieve quality information.

\[
\text{ACC} \quad ------ \quad 3
\]

Spatial point query on feature-hybrid model: The feature-hybrid model’s spatial point query is similar to that of query 2.1. First, a natural join \(\text{base} \times_{\text{fid} = \text{fid}} \text{fhq} \) is performed to retrieve the accuracy of each parcel. In the next step the x, y values are retrieved by performing a spatial join between the geometries of the point \((x,y)\) and the results of the natural join.

Query 2.3

\begin{verbatim}
SELECT a.acc
FROM
    (SELECT i.geom \cap j.acc
     FROM base i, fhq j
     WHERE i.fid = j.fid) a, poi b
WHERE a.geom \cap b.geom;
\end{verbatim}
**Results:** The spatial point query performed on the feature-hybrid model requires both a spatial join and a natural join to retrieve quality information.

### Comparing per-feature, feature-independent and feature-hybrid models on spatial point queries

Comparing per-feature, feature-independent and feature-hybrid models on spatial point queries, the per-feature and feature-hybrid require only a natural join to retrieve accuracy at the x and y coordinates of geometry stored in base table. The feature-independent model retrieves accuracy at coordinates x and y by performing a spatial join. The results of querying the feature-hybrid and feature-independent models are identical and exhibit sub-feature quality (more than one quality for a single feature). In this case, the feature-hybrid query achieves a desirable balance of greater efficiency than the feature-independent model, while still being able to represent sub-feature quality, unlike the per-feature model.

### Spatial range query (Query 3)

The spatial range query is similar to a window query in GIS. The user specifies a window or an area of interest (AOI) to be queried on a spatial data set. A spatial range query over quality information retrieves spatially varying quality over a specific area of interest. The AOI is a polygon covering south eastern part of Panton Hill suburb.

“List the accuracy of parcels based on a region (south eastern part of Panton Hill suburb)”

**Spatial range query on per-feature model:** The spatial range query for the per-feature model was developed by performing a spatial join $base \times_{fid = qid} pfq$ (see Figure 3.13) over the region’s geometries as follows:

#### Query 3.1

```sql
SELECT geom, acc
FROM region a,
     (SELECT c.geom
      FROM base c, pfq d
      WHERE c.fid = d.qid) b
WHERE a.geom \cap b.geom;
```
Results: The query results in fig 3.14 show 5 accuracies across the southern region of Panton Hill locality.

Spatial range query on feature-independent model: Using the feature-independent model for a spatial range query requires two spatial joins when compared to the per-feature model. Figure 3.15 shows the results of the first spatial join performed as:

\[\text{base} \bowtie \text{geom} \cap \text{geom fiq}\]
Query 3.2

SELECT b.geom, b.acc
FROM region a,
     (SELECT (i.geom ∩ j.geom) AS geom, j.acc
         FROM (base i, fiq j) b
     ) AS b
WHERE a.geom ∩ b.geom;

Results: The query results in Figure 3.16 have retrieved 9 accuracies across the southern region of the study area. The results reveal that except for parcel 4 (accuracy value 2.75m) all other parcels have accuracies recorded in sub-features.

Spatial range query on feature-hybrid model: The range query for feature-hybrid model involves one natural and spatial join. Geometries of the fhq table are intersected with the region’s geometries and then a natural join is performed on the results of the intersection and the base table.

Query 3.3

SELECT geom, acc
FROM base a,
     (SELECT c.geom, d.acc
         FROM region c, fhq d
     ) AS c
WHERE a.geom ∩ c.geom;
The results of query 3.3 are the same as the result of feature-independent model query 3.2. The feature-hybrid query retrieves sub-feature quality by performing one natural join and a spatial join when compared to the following:

- Per-feature query 3.1 performs one spatial and one natural join but does not retrieve sub-feature quality.
- Feature-independent query 3.2 performs two spatial joins to retrieve sub-feature quality.

In summary, the spatial range query performed on all three models retrieves spatially varying quality. This type of query enables the user to retrieve quality information on a specific area.

Nearest Neighbour (NN) query (Query 4)

The NN queries are the strength of spatial database that work on the basis of topological relations. A typical NN query fetches information of features from its nearest proximity. In Oracle spatial, a spatial operator SDO_NN is used in the NN query, which requires the geometries to be spatially indexed. The geometries, which result from a query, cannot be used by the SDO_NN operator as the geometries are not spatially indexed. In Oracle, a spatial index cannot be created
on the fly for the geometries whilst querying. Hence, the interim results of the query have to be stored as a table.

“List two nearest accuracies to any region with 1m accuracy?”

Nearest Neighbour (NN) query on Per-Feature model: The per-feature NN query is executed in two steps. The first part is to create a table pfqn based on the relation base $\text{fid} = \text{qid}$ pfq and create a spatial index on the geometries. The second part is to find the accuracies adjacent to the parcel’s region with 2m accuracy.

Query 4.1

Step 1
CREATE TABLE pfqn AS
SELECT a.fid, a.geom, b.acc
FROM base a, pfq b
WHERE a.fid = b.qid;

CREATE INDEX pfqn_idx ON pfqn(geom)
INDEXTYPE IS mdsys.spatial_index

Step 2
SELECT a.acc, b.geom
FROM pfqn b, pfqn a
WHERE b.acc = 1 AND
SDO_NN(a.geom, b.geom) = ‘TRUE’ AND ROWNUM $\leq$ 2;

Results: The query results in Figure 3.17 shows that there are two accuracies of 2m and 2.5m, which are in close proximity to 1m accuracy.

NN query on feature-independent model: The feature-independent query model uses SDO_NN operator directly to find accuracies adjacent to 2.5m accuracy.

Query 4.2

SELECT a.acc, b.geom
CHAPTER 3. MODELS OF SPATIAL VARIATION IN DATA QUALITY

Figure 3.17: Results of query 4.1

```
FROM fiq b, fiq a
WHERE b.acc = 2.5 AND
  SDO_NN(a.geom, b.geom) = 'TRUE' AND ROWNUM ≤ 2;
```

**Results:** The results of the query 4.2 retrieve 2.5 m and 2m accuracies.

Figure 3.18: Results of query 4.2

**NN query on feature-hybrid model:** Unlike the other models on the NN query, the feature-hybrid model does not require an additional table. Thus, there
is no need to create a spatial index as the fhq geometries are already spatially indexed. The SDO\_NN query operator is directly used to retrieve the neighbours.

Query 4.3

\[
\text{SELECT } a.\text{acc}, b.\text{geom} \\
\text{FROM } \text{fhq } b, \text{fhq } a \\
\text{WHERE } b.\text{acc} = 2.5 \text{ AND } \text{SDO\_NN}(a.\text{geom}, b.\text{geom}) = '\text{TRUE}' \text{ AND ROWNUM} \leq 2;
\]

Results: The results of the query 4.3 retrieve 2.5 m and 2m accuracies.

In summary, the NN query on the spatial variation models retrieves the adjacent accuracies of the region requested. The strength of the NN queries is that the user will be able to identify any alarming patterns recorded in the quality within the neighbourhood. The NN query on the per-feature model requires the query to execute in two steps due to index creation when compared to the other two models.

Complex spatial join query (Query 5)

Query 5 is termed complex as it involves more than one quality model. An example of complex query is as follows:

“List the feature ids and accuracies for parcels where the recorded per-feature
and feature-independent accuracy differ by 0.25 m.”

The complex spatial query involves a spatial join between different spatial variation models. The complex query is developed to retrieve the accuracies with a difference of 0.25m in per-feature and feature-independent models. First a natural join is performed on base and pfq relation as base ⊝ fid = qid pfq. Secondly, a spatial join is performed between the geometries of relation base ⊝ fid = qid pfq with geometries of fiq. Lastly, the accuracies are selected from the results of the spatial join, where the accuracies differ by 0.25m.

Query 5

```sql
SELECT a.fid parcelid, a.acc AS pfqacc, b.acc AS fiqacc
FROM
    (SELECT i.fid, i.geometry, j.acc
     FROM base i, pfq j
     WHERE i.fid = j.qid)
    a, fiq b
WHERE (a.geom ∩ b.geom) AND (b.acc - a.acc) = 0.5;
```

Results: The results of the complex spatial query enable comparison between the quality stored in per-feature and feature-independent model.

<table>
<thead>
<tr>
<th>PARCELID</th>
<th>PFQACC</th>
<th>FIQACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>2.5</td>
<td>3</td>
</tr>
</tbody>
</table>

Other queries such as, non-spatial range and spatial range queries (buffer), which are not discussed in the query section, were implemented for all three models. The non-spatial range query’s performance was similar to that of non-spatial point query. The buffer query implementation reveals that the query pattern of region queries were similar to that of buffer query.

3.4.3 Prototype-II summary

Analysis of Prototype II queries

1. Queries 1, 2 and 3 for the per-feature and feature-hybrid models have a similar pattern on performing the joins.
2. The per-feature and feature-hybrid query implementations use a greater number of natural joins than spatial joins, which reduce the computing cost.

3. The feature-independent model query implementation on queries 1, 2 and 3, uses only spatial joins to retrieve quality. Queries 2 and 3 have two spatial joins as the model retrieves quality based on geometries, making the model’s computation cost higher on retrieval when compared to the other two models’ query performance.

4. The commonality between feature-independent and feature-hybrid model are that they store independent quality geometries and have the ability to retrieve sub-feature variation quality.

5. Query 4 (NN query) requires additional processing to query per-feature model when compared to the other two models.

6. The complex spatial query (query 5) provides a platform for the quality models (per-feature, feature-independent and feature-hybrid) to interact between them.

3.5 Summary

In general, each model of spatial variation is different in its representational and querying capabilities. The per-feature model is easiest to implement, simply storing quality as an additional attribute. However, the model is unable to represent sub-feature variation. The feature-independent model represents quality information independent of its spatial features. As a result, the feature-independent model can represent sub-feature variation. As the quality has geometries associated with it, additional storage and an increase in processing time during querying affects the model’s efficiency. The feature-hybrid model is derived from a combination of per-feature and feature-independent quality models. Both non-spatial and spatial joins are required in querying this model. The storage cost is higher for the hybrid model when compared to the per-feature and the feature-independent model. However, no model is entirely superior in storing and retrieving spatially varying quality and none of the models can be ignored. Hence an integrated approach is required, which allows flexible representation of the per-feature, feature-independent, and feature-hybrid quality models. By developing an integrated approach, the user can retrieve quality irrespective of which data model is used to store it. The integrated approach is discussed in the next chapter (Chapter 4).
Chapter 4

RSVQ: RDBMS for Spatial Variation in quality

4.1 Introduction

The results of the implementation of prototype I and II discussed in chapter 3 emphasise the need for a quality system that allows flexible representation of per-feature (PF), feature-independent (FI) and feature-hybrid (FH) data quality models. Apart from flexible representation, the results of the prototypes’ implementation and summary of the literature survey discussed in chapter 3 stress the need for an efficient retrieval mechanism that can retrieve spatially varying quality including sub-feature variation. This chapter focuses on the development of an SDQ management system primarily aiming to overcome the limitations of prototypes I and II discussed in the previous chapter. The system is called RDBMS for Spatial Variation in Quality (RSVQ). Section 4.2 discusses the development of the RSVQ formal model using relational algebra. Section 4.3 explains the working of the formal model using query examples. The chapter concludes with the summary section.

4.2 Formal model

In the following, a formal model for storing and querying spatially varying data quality information in a relational database was developed based on the relational algebra (see section 3.3.1). The model defines the underlying relational structure of the quality information to be stored and defines a derived relational operator for querying quality. Fig 4.1 illustrates the structure of the RSVQ which integrates PF, FI and FH data quality models.
4.2.1 Preliminaries

The formal model is developed based on the three data quality models discussed in Chapter 3. Let $S$ be a spatial framework, $Q$ a set of quality records, and $O$ as set of geospatial objects. Then the per-feature model is defined as a function that maps each geospatial object to a unique quality record, $f : O \rightarrow Q$. The feature independent model is defined as a function that maps each location in space to a unique quality record, $g : S \rightarrow Q$. The feature hybrid model is defined as a function that maps each location in each geospatial object to a unique quality record, $h : O \rightarrow Q^S$ where $Q^S$ denotes the set of functions (fields) from $S$ to $Q$.

The following definitions 1, 2, 3 and 4 are generalised based on the prototype II implementation (see section 3.5):

**Definition 1.** A feature relation (or a “layer”) $l$ has a relation scheme of the form $l(\text{id}: \text{int, geom: GEOMETRY, ...})$ where $\text{id}$ is the identifier for the geospatial objects about which we wish to store quality information, and $\text{geom}$ is the geometry of those objects. Each tuple in a feature relation stores information about an object $o \in O$. $\mathcal{L}$ denotes the set of all feature relations.

**Definition 2.** A per-feature quality relation $qf$ has the relation scheme of the form $qf(\text{oid}: \text{int, ...})$ where $\text{oid}$ 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 \( q \in Q \) along with the identifier of the object to which that quality record refers. \( Q_f \) denotes the set of all per-feature quality relations.

**Definition 3.** A feature independent quality relation \( q_g \) has the relation scheme of the form \( q_g(\text{qid}: \text{int}, \text{qgeom}: \text{GEOMETRY}, ...) \) where \( \text{qid} \) is a quality identifier, and \( \text{qgeom} \) is a geometry attribute that describes the spatial characteristics of each quality feature. 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. \( Q_g \) denotes the set of all feature independent quality relations.

**Definition 4.** A feature hybrid quality relation \( q_h \) has the relation scheme of the form \( q_h(\text{qid}: \text{int}, \text{oid}: \text{int}, \text{qgeom}: \text{GEOMETRY}, ...) \) where \( \text{qid} \) is a quality identifier, \( \text{oid} \) is a foreign key which relates qualities to features in a specific feature relation, and \( \text{qgeom} \) 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 \( q \in Q \) along with the set of locations that map to that quality record and the identifier of the object to which that quality record refers. \( Q_h \) denotes the set of all feature hybrid quality relations.

Quality information stored using either the per-feature or feature-hybrid model relates to a specific feature relation (with matching feature identifiers). For simplicity, we assume here that the quality relation stores information only about a feature relation, not about specific attributes, although the model could be extended to allow the quality of a specific attribute to be stored. Information about which feature relation matches with which per-feature or feature-hybrid quality relation must be stored in the RSVQ system.

**Definition 5.** The metadata function, \( m : Q_f \cup Q_h \rightarrow \mathcal{L} \), identifies the unique feature relation that each per-feature or feature-hybrid quality relation that stores the quality of.

Note that because \( m \) is a function, each per-feature or feature hybrid quality relation must have exactly one associated feature relation. Feature-independent quality relations are excluded from the domain of \( m \) because they do not refer to a specific feature relation.
4.2.2 Quality operator

The querying of the quality data is formalized using a derived relational operator. The quality operator is represented by the symbol $\triangleleft$. The operator $\triangleleft$ is polymorphic, exhibiting different behaviours in the context of the different data quality models (PF, FI and FH). Recalling the queries discussed in prototypeII implementation (see section 3.5) the non-spatial point query (Query1) required a natural join to retrieve quality from the PF and FH model. The FI model required a spatial join to retrieve the quality. Although, PF and FH are structurally different but their query behaviour was same for query 1. The query behaviour of FI model for query 1 was different from the other two models as it had required a spatial join to retrieve quality. Another typical query behaviour noted using the PF model was depicted in the spatial point query (Query 2). The PF model required both spatial and a natural joins to retrieve quality. Thus, two join operations of different type (natural and spatial join) define another behaviour of the PF query. In short, the $\triangleleft$ operator annotates the geospatial objects in a feature relation $l$ with associated quality information from a quality relation $q$.

Definition 6. The relational quality operator $\triangleleft$ takes two tables as arguments: a feature relation $l \in L$ and a quality relation $q \in Q$, where $Q = Q_f \cup Q_g \cup Q_h$. The operator returns a single joined relation $l \triangleleft q$ where the qualities from relation $q$ are associated with related features from $l$. The relation $l \triangleleft q$ is a type of join, $l \triangleleft q \subseteq l \times q$, with attributes $\text{att}(q) \cup \text{att}(l)$ (where $\text{att}(r)$ is the set of attributes of a relation $r$). It need not be the case that $m(q) = l$ (i.e., we may apply per feature quality information to any feature relation $l \in L$, not only a feature relation which has its foreign key in $q$).

4.2.3 Quality operator behaviour

Essentially, quality operator $\triangleleft$ performs natural, spatial, or combined spatial and natural joins depending on the relations used as arguments. For an operation $l \triangleleft q$, the choice of which behavior the quality operator exhibits depends whether the quality relation $q$ is a FI or PF or FH quality relation, and in the case of the latter whether $l = m(q)$. Three different behaviours, B1, B2, and B3, are detailed below. Table 4.1 summarizes the relationship between the choice of behaviours and the different possible arguments.
Behavior of $l \triangleleft q$

<table>
<thead>
<tr>
<th>Behavior</th>
<th>$q \in \mathcal{Q}_f$</th>
<th>$q \in \mathcal{Q}_g$</th>
<th>$q \in \mathcal{Q}_h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $l = m(q)$</td>
<td>B1</td>
<td>n/a</td>
<td>B1</td>
</tr>
<tr>
<td>Otherwise</td>
<td>B3</td>
<td>B2</td>
<td>B2</td>
</tr>
</tbody>
</table>

Table 4.1: Summary of behaviours for polymorphic quality operator behaviour

### B1 behaviour

As shown in Table 4.1, the B1 behaviour is triggered when $q \in \mathcal{Q}_f \cup \mathcal{Q}_h$ and $l = m(q)$. The B1 behaviour of the $\triangleleft$ operator, $l \triangleleft^{B1} q$, implements a natural join of the quality relation $q$ and the feature relation $l$ based on their shared object identifiers, as follows:

$$l \triangleleft^{B1} q \equiv l \bowtie \_id=oid \ q$$

**Example of B1 behaviour:** Recalling query 1.1 of section 3.4.2 of chapter 3, the natural join $a.fid = b.qid$ performed to retrieve pf quality is an example of behaviour B1 mentioned above.

**Query1.1**

```sql
SELECT a.fid, b.acc
FROM base a, pfq b
WHERE a.fid = b.qid AND a.fid = 5;
```

### B2 behaviour

The B2 behaviour is triggered when $q \in \mathcal{Q}_g$, or when $q \in \mathcal{Q}_h$ and $l \neq m(q)$. The B2 behaviour of the $\triangleleft$ operator, $l \triangleleft^{B2} q$, implements a spatial join of the quality relation $q$ and the feature relation $l$ based on their shared geometries, as follows:

$$l \triangleleft^{B2} q \equiv l \bowtie geom \cap qgeom \neq \emptyset q$$

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

**Example of B2 behaviour** Recalling query 1.2 of section 3.4.2 of chapter 3, the spatial join $a.geom \cap b.geom$ performed to retrieve fi quality is an example of behaviour B2 mentioned above.
Query 1.2

```
SELECT a.fid, b.acc
FROM base a, fiq b
WHERE a.geom ∩ b.geom AND a.fid = 5;
```
In order to improve efficiency, the following function $m'$ is introduced.

**Definition 7.** The restriction rewriting function $m'$ is defined as $m' : Q_{\subseteq} \rightarrow Q_f \cup Q_h$, where $Q_{\subseteq} = \{ \sigma_a(x) | x \in Q_f \text{ or } x \in Q_h \}$, and $a$ is any subset of attributes of $x$, and $m'(\sigma_a(x)) = x$.

In effect $m'$ keeps track of the root quality table for any restriction of that table. Now we can replace the condition in Table 4.1 $l \subseteq m$ with $l \subseteq m \circ m'(\sigma_a(q))$, where $a$ is any subset of attributes of $q$.

### 4.4 Example queries

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

- **Lin(qid, comment)**: per-feature quality relation representing lineage;
- **Prec(qid, qgeom, precision)**: feature independent quality relation representing precision;
- **PosAcc(qid, oid, qgeom, acc)**: feature hybrid quality relation representing positional accuracy; and
- **Reg(id, geom, cat)**: a feature table of regions, where $m(\text{Lin}) = \text{Reg}$.

Then we can pose and formalize the following example queries using the $\lhd$ operator in combination with standard relational algebra operators.

**Query 1:** What lineage information is associated with region feature with id=5?

\[ \sigma_{id=5}(\text{Reg} \lhd \text{Lin}) \]

The $\lhd$ operator will execute B1 behaviour (natural join) as the relation Lin is of type PF. The output of the natural join will be restricted by the $\sigma_{id=5}$ expression to give the final output. As per the efficiency consideration section, the restriction operation will be first performed. The result of the restriction will be then joined (natural join) with the relation Lin as follows : $\sigma_{id=5} (\text{Reg} \lhd \text{Lin})$.

**Query 2:** What is the precision of the region feature with id=5?

\[ \pi_{\text{precision}}(\sigma_{id=5}(\text{Reg} \lhd \text{Prec})) \]
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 $\sqcap$ operator will be different (B2 behaviour in query 2 as opposed to B1 behaviour in query 1). The reason for B2 behaviour to execute in query 2 is because the relation Prec is of type FI.

**Query 3:** List the types of roads that have a positional accuracy of less than 5m RMSE.

$$\pi_{\text{type}}(\sigma_{\text{rmse}<5m}(\text{Reg} \sqcap \text{PosAcc}))$$

The expression above will execute B1 behaviour. The B1 behaviour is executed because the relation PosAcc is of type FH.

**Query 4:** Find all the positional accuracy information contained within a rectangular window $R$ (spatial range query).

$$R \sqcap \text{PosAcc}$$

where the relation $R(\text{id}, \text{geom})$ contains one tuple, $(1, w)$. The above expression will execute B2 behaviour as the relation PosAcc is of type FH.

**Query 5:** Find all the positional accuracy information contained within a rectangular window $R$ (spatial range query).

$$R \sqcap \text{Lin}$$

where the relation $R(\text{id}, \text{geom})$ contains one tuple, $(1, w)$. The above expression will execute B3 behaviour as the relation Lin is of type PF.

### 4.5 Summary

The development of the formal model *RDBMS for Spatial Variation Quality in (RSVQ)* was discussed using relational algebra. To understand the working of the formal model few examples were illustrated. The model has also considered efficient ways to retrieve spatially varying quality. The implementation of the formal model is discussed in the next chapter 5.
Chapter 5

Formal model implementation

5.1 Introduction

The implementation of the formal model using Oracle Spatial RDBMS is the main theme of this chapter. Section 5.2 describes the study area chosen for the case study. The database design is discussed in section 5.3. Section 5.4 illustrates the steps involved in the development of the WITHQUALITY keyword. Section 5.5 details the conversion between data quality models. The user interface is explained in section 5.6. The chapter concludes with a summary in section 5.7.

5.2 Study area

5.2.1 Location

Hume is a local government area (LGA) in the state of Victoria, Australia. Hume was selected as the study area based on the recommendations of the DSE (Department of Sustainability and Environment, custodian of spatial data in the state of Victoria). Hume has a very strong industrial base, with motor vehicle manufacturing and heavy engineering as the main industries. The southern parts of the city are well-established urban areas, while the north remains rural in character. The City of Hume covers an area of approximately 503 square km, north west of Melbourne. Hume LGA consists of 75445 parcels (see figure 5.1).

5.2.2 SDQ of Hume LGA

The reason for selecting Hume LGA was because Hume had recorded the maximum range of spatially varying positional accuracy (precision) for its cadastral data. The range of the positional accuracy was recorded as: 0.1m, 0.5m, 2.5m,
10m and 25m. The high precision (0.1m) is derived from the incorporation of new digital subdivisions into the mapbase. All other precisions reflect the capture scale of the original mapping data used for converting the cadastre into a digital format. The 0.5m precision is derived from 1:500 mapping, 2.5m precision is derived from 1:2500 mapping, 10m precision from 1:10,000 mapping and 25m precision from 1:25,000 mapping. The positional accuracies of the cadastral data have been recorded as points over the parcel. The graphic display of the positional accuracies is shown in figure 5.2.
5.3 Database design

5.3.1 RSVQ relationship

The Unified Modeling Language (UML), a general-purpose modeling language that includes a graphical notation was used to design the RSVQ data model. The UML diagram in figure 5.3 depicts the relationship for the relations l (parcel layer), qf (per-feature quality), qg (feature-independent quality) and qh (feature-hybrid quality) discussed in section 4.2.1.

The relation l (layer) has a one-to-one (1 to 1) relationship with the per-feature quality relation qf based on the definition 2 in section 4.2.1. A one-to-one relationship maps the id of relation l with oid of relation qf (per-feature quality).

The relationship between the feature-independent relation qg and the relation l defined in definition 3 in section 4.2.1. is depicted as many-to-many (N to N) relationship in the UML diagram. A many-to-many relationship maps the geom of relation l with one or more qgeom of relation qg and vice versa in the data model.

The relationship defined between the relation l and the feature-hybrid relation qh in definition 4 in section 4.2.1 is depicted as one-to-many (1 to N) relation qh in the UML diagram. A one-to-many relationship maps id of relation l with one or more oid attribute of relation qh.

5.3.2 Data format and database scheme

The data provided by DSE was supplied in ESRI shape file format. Spatial Console visualisation software was used to import the shape files into Oracle Spatial format. The tables parplh (parcels of Hume local government area), pfoh (the quality element precision for Hume stored as per-feature quality table), fioh (the quality element precision for Hume stored as feature-independent quality table) and fhoh (the quality element precision for Hume stored as feature-hybrid quality table) were created by importing the shape files.

Parcel table implementation: The relation scheme defined for layer l in definition 1 of section 4.2.1 was adopted for table parplh. Thus the table parplh has the following relational scheme:

parplh(id: number, geom: GEOMETRY).

Oracle Spatial has a data type called SDO_GEOMETRY (see figure 5.4) to represent point, lines and polygons.
Figure 5.3: UML diagram depicting the relationship between the qualities and layer relation

Figure 5.4: Oracle spatial SDO GEOMETRY structure
The per-feature quality table implementation: The per-feature quality table `pfoh` was generated by aggregating the positional accuracy of each parcel by performing an average built-in-function `AVG`. By aggregating the positional accuracy each parcel is associated with a single quality value. The table `pfoh` follows the relational scheme defined in definition 2 of section 4.2.1. Thus the relation scheme for the table `pfoh` is given as:

\[ pfoh(\text{oid}: \text{number}, \text{prec}: \text{number}) \]

Some sample records are listed in table 5.2 after the implementation of the above relational scheme.

Feature-independent quality table implementation: The table `fioh` was created by importing the positional accuracy point data provided by the DSE. The table has an additional column `geom` when compared to the per-feature table. The column `geom` stores quality as a point geometry which is independent of the `parplh` polygon geometry. Based on the definition 3 of section 4.2.1, the table `fioh` has the following relational scheme:

\[ fioh(\text{qid}: \text{number}, \text{geom}: \text{GEOMETRY}, \text{prec}: \text{number}) \]

A sample record from table `fioh` is shown in table 5.3 after the implementation of the above relation scheme.
CHAPTER 5. FORMAL MODEL IMPLEMENTATION

Table 5.3: Oracle table-fioh implementation (sample record)

<table>
<thead>
<tr>
<th>QID</th>
<th>GEOM</th>
<th>PREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>491</td>
<td>SDO_GEOMETRY(2001, 8311, SDO_POINT_TYPE(144.922009, -37.666654, NULL), NULL, NULL)</td>
<td>.5</td>
</tr>
</tbody>
</table>

Table 5.4: Oracle table-floh implementation (sample record)

<table>
<thead>
<tr>
<th>OID</th>
<th>QID</th>
<th>GEOM</th>
<th>PREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>40704</td>
<td>491</td>
<td>SDO_GEOMETRY(2001, 8311, SDO_POINT_TYPE(144.922009, -37.666654, NULL), NULL, NULL)</td>
<td>.5</td>
</tr>
</tbody>
</table>

Feature-hybrid quality table implementation: The table fhoh was generated by performing a spatial join between the geometries of parph and fioh. The table fhoh follows the relational scheme defined in the definition 4 of section 4.2.1. Thus, the table fhoh has the following relational scheme:

\[
\text{fhoh}(\text{qid: number, oid: number, geom: GEOMETRY, prec: number}).
\]

A sample record of table fhoh is shown in table 5.4 after the implementation of the above relation scheme.

Metadata table implementation: A metadata table metap stores the relation between layer and quality tables. The relation metap has the following relational scheme:

\[
\text{metap(qtab: varchar2(32), ftab: varchar2(32), qtyp: varchar2(32))}.
\]

The column qtab stores the name of the quality table, the column ftab stores the name of the layer table and the column qtyp stores the type of quality (per-feature or feature hybrid). The table 5.5 depicts the metap relation. Recalling definition 5 of section 4.2.1, feature-independent quality relations are excluded because they do not refer to a specific feature relation. The per-feature or feature-hybrid quality relation have exactly one associated feature relation as stated in the definition 5. The usage of metap table will be explained during the discussion on the quality query operator later in this chapter.

<table>
<thead>
<tr>
<th>QTAB</th>
<th>FTAB</th>
<th>QTYP</th>
</tr>
</thead>
<tbody>
<tr>
<td>fioh</td>
<td>parph</td>
<td>pf</td>
</tr>
<tr>
<td>fhoh</td>
<td>parph</td>
<td>fh</td>
</tr>
</tbody>
</table>

Table 5.5: Metadata table
5.4 WITHQUALITY keyword

5.4.1 Existing query mechanism

The next step to address in the database design is the querying mechanism. The observations from implementing Prototype II were considered while developing the query mechanism. The observations were as follows:

- Three behaviours were required while retrieving spatially varying quality from the per-feature, feature-independent and feature-hybrid representations of spatially varying quality. The first behaviour termed B1 requires a natural join to query quality. The second behaviour termed B2 requires a spatial join to query quality. Lastly, the third behaviour B3 requires both natural and spatial join to query quality. The behaviours B1, B2 and B3 relate to the three fundamental categories of queries in a spatial information system defined by Bultzingsloewen (1987) as: queries about non-spatial properties (B1 behaviour); queries exclusively about spatial properties (B2 behaviour); and queries which combine spatial and non-spatial properties (B3 behaviour).

- The query structure in Prototype II implementation had nested spatial and non-spatial joins. The nested queries have a complex SQL structure which is hard to understand and implement.

- To query multiple quality tables at the same time in Prototype II implementation would have been a more complex task. The reason for more complexity is that more nested queries were required in addition to the existing nested structure which had been criticized above.

- The queries tested in Prototype II implementation can be redesigned to improve query execution time.

5.4.2 New query mechanism

Considering all the factors mentioned in section 5.4.1, the quality operator ◄ was developed in the formal model. The formal quality operator was developed as a WITHQUALITY keyword as an extension to standard SQL. Some of the guidelines suggested by Egenhofer (1994) for an SQL extension were adopted in the implementation. The guidelines adopted are:

1. The premises of the design of the SQL extension should retain the concepts of the host language.
2. The characteristic structure of the language with the SELECT-FROM-WHERE should stay untouched.

3. Every query result is a relation.

4. Predicates in the WHERE clause are formulated upon attributes.

In the SQL extension, the WITHQUALITY keyword takes three input parameters: layer, quality and an alias name. The layer and quality are tables. The alias name is an output table name, which stores the results of the WITHQUALITY operation. The syntax for the quality query operator is:

\[
\text{<layer table name> WITHQUALITY <quality table name> <alias name>}
\]

In SQL, for example, to retrieve a quality information from per-feature model for a parcel number 30, the following query (same as Query 1.1, chapter 3) is used:

**SQL structure:**

```sql
SELECT a.fid, b.acc
FROM base a, pfq b
WHERE a.fid = b.qid AND a.fid = 30;
```

The same query (Query 1.1) is restructured using WITHQUALITY keyword as:

**SQL extension structure:**

```sql
SELECT x.fid, x.acc
FROM base WITHQUALITY pfq x
WHERE x.fid = 30;
```

The SQL extension structure mentioned in Query 1.1 follows the same structure of SQL satisfying the guidelines no 1, 2 and 4 mentioned by Egenhofer (1994). The output of the query is a view which satisfies guideline number 3. Like normal SQL, the SQL extension mentioned here allows the user to apply the WHERE clause to perform operations like restriction (\(id < 5\) or \(prec \text{ LIKE } \%0.5\)) which satisfies guideline number 4. The WITHQUALITY keyword uses its polymorphic property to determine the type of join (natural, spatial or both) to be performed. The user does not need any prior knowledge on data quality model (per-feature
or feature-independent or feature-hybrid) adopted by the quality table. All that
the user needs to know is the name of the layer and its corresponding quality
table to retrieve quality information.

**Example of LIKE keyword usage in SQL extension:**

```
SELECT x.prec, x.acc
FROM  base WITHQUALITY pfq x
WHERE x.prec LIKE '0.5';
```

Additional operations like **GROUP BY** and **HAVING** and built-in-functions like
**MAX**, **MIN** and **AVG** are supported by the SQL extension.

**Example of MAX() built-in-function, GROUP BY and HAVING clause in SQL extension:**

```
SELECT x.id, MAX(x.prec)
FROM  base WITHQUALITY pfq x
WHERE x.prec ≤ 10 GROUP BY id HAVING MAX(x.prec) ≥ 0.5;
```

Spatial operations like **BUFFER**, **DISTANCE** and **OVERLAY** are also included in
the SQL extension.

**Example of BUFFER operation using SQL extension:**

```
SELECT x.id, SDO.GEOM.SDO_BUFFER (a.geom,0.25,0.5,'unit = mile') buf
FROM  base WITHQUALITY pfq x
WHERE x.prec = 2.5;
```

In conclusion, the SQL extension provides all spatial and non-spatial query func-
tionalities as normal SQL provides.

### 5.4.3 Working of WITHQUALITY keyword

Oracle Corporation’s proprietary procedural extension to the SQL database lan-
guage PL/SQL was used for programing the **WITHQUALITY** keyword. The fol-
lowing query is taken as an example to explain the working of the **WITHQUALITY**
keyword:
Sample query 1: What is the precision at parcel 30?

```sql
SELECT x.id, x.prec
FROM parplh WITHQUALITY pfoh x
WHERE x.id = 30;
```

**STEP1:** The sample query 1 is assigned to a variable `qs` as shown in figure 5.5.

**Figure 5.5: Flowchart 1: steps involved in sample query 1 execution**

**STEP2:** The string stored in `qs` is subdivided into three parts. The subdivision is achieved by programming the functions `proj()`, `mid()` and `cond()` which are shown in figure 5.5. A function in Oracle PL/SQL is a subprogram which performs a specified task and returns a value. The function `proj(qs)` returns the first part of the query string, `SELECT x.id, x.prec FROM` and it is stored in a variable `p`. The function `mid(qs)` returns the string, `parplh WITHQUALITY pfoh x` and it is stored in a variable `m`. Finally, the function `cond(qs)` returns the string `WHERE x.id = 30` and it is stored in a variable `c`.

**STEP3:** The middle part of the query string `qs`, `parplh WITHQUALITY pfoh x` which had been assigned to the variable `m` is further subdivided into three parts. As shown in figure 5.5, the function `laye(m)` returns the layer table name (parplh)
and it is assigned to the variable \( l \). The function \( \text{quae}(m) \) returns the quality table name (\( \text{pfoh} \)) and it is assigned to the variable \( q \). The function \( \text{res}(m) \) returns the alias name \( x \) and assigns it to the variable \( r \).

**STEP 4:** In this step, the *procedure* \( \text{dia}(l,q,r) \) is called. A procedure in Oracle PL/SQL is a subprogram which performs a specified task. Unlike functions, procedures do not return a value. The procedure \( \text{dia}(l,q,r) \) is an implementation of the \(<\) operator mentioned in the formal model and \( \text{WITHQUALITY} \) keyword. The procedure \( \text{dia} \) takes three parameters \( l, q \) and \( r \) as inputs. The values for the parameters have already been processed in the previous step, where \( l \) is assigned to \( \text{parplh} \), \( q \) is assigned to \( \text{pfoh} \) and \( r \) is assigned to \( x \) as shown in S3 in figure 5.5. The parameter \( r \) is not a part of the \(<\) operator discussed in the formal model. The additional parameter \( r \) was included as an SQL processing requirement. The parameter \( r \) is the name of an *Oracle view* which is created on the fly during the query execution. A view is a virtual or logical table composed of the result set of a query. The results of the \( \text{parplh WITHQUALITY pfoh} \) are stored in the view \( x \).

**STEP 5:** The final step, creates a *view* called \( \text{temp} \) on the fly to store the results of the sample query 1 after the execution of the appropriate behaviour. The variable \( p \) containing the string 'SELECT x.id, x.prec FROM' and \( c \) containing the string 'WHERE x.id = 30' from STEP 2, and the variable \( r \) containing the string 'x' from STEP 3 are substituted in the following dynamic SQL to create the output:

```sql
sql1 := 'CREATE OR REPLACE VIEW temp AS '||p||' '||r||' '|| c;
EXECUTE IMMEDIATE sql1;
```

**Results of sample query 1 (What is the precision at parcel 30 ?):** To view the results of the query the *view* \( \text{temp} \) is queried using the following SQL:

```sql
SELECT *
FROM   temp
```

**OUTPUT:** sample query 1

<table>
<thead>
<tr>
<th>ID</th>
<th>PREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>1.5</td>
</tr>
</tbody>
</table>
The working of procedure `dia(l,q,r)` is explained in the following steps with the help of figure 5.6:

**S1:** The parameters `l,q,r` are read as inputs. `l` has the value `parlh`, `q` has the value `pfoh` and `r` has the value `x`.

**Figure 5.6: Flowchart 2: the steps involved in the function dia**

**S2:** The variable `c1` is assigned to the value which is return by the function `pfc(q)`. The function checks whether the quality table is per-feature type. The value 1 is returned, if the quality type is per-feature else value 0 is returned. The metadata table `metap` is queried by the function `pfc(q)` with the following SQL:

```sql
SELECT count(qtab)
FROM metap
WHERE qtab = 'q' AND qtyp = 'PF'
```

In our case, the function returns the value 1 as `pfoh` is present in the `metap` table (see table 5.1). Thus `c1` is assigned the value 1.

**S3:** The variable `c2` is assigned to the value which is return by the function `fhc(q)`. The function checks whether the quality table is feature-hybrid type. The
value 1 is returned, if the quality type is feature-hybrid else value 0 is returned. The metadata table \texttt{metap} is queried by the function \texttt{fhc(q)} with the following SQL:

\begin{verbatim}
SELECT count(qtab) 
FROM   metap 
WHERE qtab = 'q' AND qtyp = 'FH'
\end{verbatim}

In our case, the function returns the value 0 as \texttt{qtyp} is PF. Thus \texttt{c2} is assigned the value 0.

\textbf{S4:} The variable \texttt{t1} is assigned to the function \texttt{lsq(l,q)}. The function returns the layer \texttt{l} by checking whether the layer \texttt{l} is present in the metadata table \texttt{metap} with its corresponding quality table \texttt{q}. In short, the metadata function \texttt{m} is executed which was discussed in the definition 5 of section 4.2.1.

What happens if the quality \texttt{q} is a \(\sigma(q)\) (subset of a quality table)? A subset quality table has only parts of the records of quality table \texttt{q}. Anticipating that the input parameter quality \texttt{q} can also be a subset of the quality table, the formal model had introduced a function \texttt{m'} (see definition 7, section 4.3). A meta-sub-quality table \texttt{sqmetah} has been populated to support the \texttt{m'} function. The \texttt{sqmetah} table stores the relation between the main quality and sub quality tables. The relation scheme is given as:

\texttt{sqmetah(stab varchar2(20), qtab varchar2(20))}.

The column \texttt{stab} stores the name of the sub quality table and column \texttt{qtab} stores the name of the quality table. The function \texttt{lsq(l,q)} also checks whether the quality parameter \texttt{q} is a sub quality and returns the layer \texttt{l} by executing the following query:

\begin{verbatim}
SELECT a.ftab 
FROM   metap a, sqmetah b 
WHERE a.qtab = b.qtab AND stab = 'q'
\end{verbatim}

Another scenario would be, the table layer \texttt{l} can also be a subset of a layer table given as \(\sigma(l)\). A subset layer table contains parts of the records of \texttt{l}. In such a scenario, the function \texttt{lsq(l,q)} performs a sub-table check (STC) on the input parameter \texttt{l} to find whether \texttt{l} is a subset of \texttt{l} stored in the metadata table \texttt{metap}. If the input \texttt{l} is a subset then the function \texttt{lsq} returns the layer \texttt{l}. Thus, the variable \texttt{t1} is assigned the value \texttt{parplh} which was returned by the function \texttt{lsq(l,q)}
S5: Similar to S4 the function \( qsmq(q) \) checks whether the input parameter \( q \) is a subset of a quality table stored in \( metap \) table. In our case the function returns \( pfoh \) and assigns it to the variable \( t2 \).

S6: The condition, if \( t1 \) and \( t2 \) are \textbf{NOT NULL} and \( t2 \) is a per-feature quality type, is checked.

S7: If the condition mentioned in S6 is \textbf{TRUE} then the procedure \( b1(t1,t2,r) \) is called. The procedure \( b1 \) represents the \textbf{B1} behaviour discussed in section 4.3.3. In our case, the condition in S6 was \textbf{TRUE} hence, the following code for \textbf{B1} behaviour was executed.

```
--PL/SQL code for B1 behaviour
CREATE OR REPLACE PROCEDURE B1 (l VARCHAR2,q VARCHAR2,r VARCHAR2)IS
  sql1 VARCHAR2(2000);
BEGIN
  -- sql1 performs a non-spatial join between
  -- l and q when there layer.id = quality.oid
  sql1 := 'CREATE OR REPLACE VIEW ' || r || ' as
    SELECT *
    FROM ' || l || ' a, ' || q || ' b
    WHERE a.id = b.oid';
  EXECUTE IMMEDIATE sql1;
END B1;
```

S8: If the condition in S6 is \textbf{FALSE} then the condition, if \( t1 \) and \( t2 \) are \textbf{NOT NULL} and \( t2 \) is a feature-hybrid quality type, is checked.

S9: If the condition mentioned in S8 is \textbf{TRUE} then the procedure \( b1(t1,t2,r) \) is called and executed.

S10: If the condition in S8 is \textbf{FALSE} then the condition, if \( t1 \) \textbf{IS NULL}, \( t2 \) is \textbf{NOT NULL} and \( t2 \) is a per-feature quality type, is checked.

S11: If the condition mentioned in S10 is \textbf{TRUE} then the procedure \( b3(l,q,r) \) is called. The procedure \( b3 \) represents the \textbf{B3} behaviour discussed in section 4.3.3. The following code for \textbf{B3} behaviour is executed.
-- PL/SQL code for B3 behaviour
CREATE OR REPLACE PROCEDURE B3 (l VARCHAR2, q VARCHAR2, r VARCHAR2) IS
sql1 VARCHAR2(100); p1 VARCHAR2(100); p2 VARCHAR2(100);
sql2 VARCHAR2(2000); sql3 VARCHAR2(2000); sql4 VARCHAR2(100);
sql5 VARCHAR2(2000); sql6 VARCHAR2(2000);
BEGIN
  -- sql1 calls function m(q) to get 'l' which is related to q
sql1 := m(q);
  IF sql1 IS NOT NULL THEN
    p1 := 'SELECT id as tid, geom as g FROM ' || sql1;
    -- non-spatial join
    sql2 := 'SELECT *
              FROM (' || p1 || ') c,' || q || ' d
              WHERE c.tid = d.oid';
    -- spatial join
    sql3 := 'CREATE OR REPLACE VIEW ' || r || ' as
              SELECT *
              FROM (' || sql2 || ') a, ' || l || ' b
              WHERE SDO_RELATE(a.g,b.geom,
                              'mask = ANYINTERACT') = 'TRUE';
    EXECUTE IMMEDIATE sql3;
  ELSE
    -- sql4 calls function m'(q) to get "l" which is related to q
sql4 := mq(q);
    p2 := 'SELECT id as tid, geom as g FROM ' || sql4;
    -- non-spatial join
    sql5 := 'SELECT *
              FROM (' || p2 || ') c,' || q || ' d
              WHERE c.tid = d.oid';
    -- spatial join
    sql6 := 'CREATE OR REPLACE VIEW ' || r || ' as
              SELECT *
              FROM (' || sql5 || ') a, ' || l || ' b
              WHERE SDO_RELATE(a.g,b.geom,
                              'mask = ANYINTERACT') = 'TRUE';
    EXECUTE IMMEDIATE sql6;
  END IF;
END B3;
S12: If the condition in S10 is FALSE then the condition, if \( t_1 \) IS NULL, \( t_2 \) is NOT NULL and \( t_2 \) is a feature-hybrid quality type, is checked.

S13: If the condition mentioned in S12 is TRUE then the procedure \( b_2(l,q,r) \) is called. The procedure \( b_2 \) represents the B2 behaviour discussed in section 4.3.3. The following code for B2 behaviour is executed.

```sql
--PL/SQL code for B2 behaviour
CREATE OR REPLACE PROCEDURE B2 (l VARCHAR2,q VARCHAR2,r VARCHAR2)
IS
sql1 VARCHAR2(2000);
BEGIN
-- Geometries of \( l \) and \( q \) are intersected to form
-- a new geometry in the output (view)
sql1 := 'CREATE Table ' ||r|| ' as '||
   'SELECT a.id, b.qid, b.qgeom, b.prec
    FROM ' ||l||' a, ' ||q|| ' b' || ' '||
   'WHERE SDO_RELATE (a.geom, b.qgeom,
                    'mask = ANYINTERACT') = 'TRUE';
EXECUTE IMMEDIATE sql1;
END B2;
```

S14: If the condition mentioned in S12 is FALSE then the procedure \( b_2(l,q,r) \) is called and executed.

### 5.4.4 Optimization of WITHQUALITY keyword

The next step to test the WITHQUALITY keyword was by querying the feature-independent table \( fioh \). The same sample query 1 was modified as follows:

**Sample query 2:** What is the precision at parcel 30?

```sql
SELECT x.id, x.prec
FROM parplh WITHQUALITY fioh x
WHERE x.id = 30;
```

**Results of sample query 2:** The result of the query retrieved 6 records as shown below.

OUTPUT: sample query 2
The parcel number 30 exhibits sub-feature variation as it has more than one quality (prec) associated with it. One alarming factor noted was that the time taken to execute the query was thirty-seven minutes and thirty-five seconds (00:37:35), which is very long to retrieve a record. Hence, the query processing steps were analysed. The best way to analyse the steps in query processing was to convert the query into a query tree.

**Query tree**  The sample query 2 was translated to a query tree with relational algebraic expressions as shown in figure 5.7. Four levels L1, L2, L3 and L4 were assigned to the query tree. The query processing steps traverse from L1 to L4.

- **Level L1:** In the first level, the layer l (parph) and the feature-independent quality q (fioh) were taken as input.

- **Level L2:** At the second level, the join operation (WITHQUALITY) was performed. As the quality is feature-independent, B2 behaviour was executed which involves a spatial join. The join was performed between the geometries of parph (75, 445 records) and fioh (370, 407 records). Although, the geometries of both layer and quality were spatially indexed the time taken to perform the join operation was noted as thirty-seven minutes and thirty-four seconds (00:37:34).

- **Level L3:** As shown in figure 5.7, the restriction operation (σ) is done on the results obtained from the join operation performed in the previous level L2.

- **Level L4:** Finally, the projection operation was performed by projecting the attributes x.id and x.prec.

<table>
<thead>
<tr>
<th>ID</th>
<th>PREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>2.5</td>
</tr>
<tr>
<td>30</td>
<td>2.5</td>
</tr>
<tr>
<td>30</td>
<td>2.5</td>
</tr>
<tr>
<td>30</td>
<td>.5</td>
</tr>
<tr>
<td>30</td>
<td>.5</td>
</tr>
<tr>
<td>30</td>
<td>.5</td>
</tr>
</tbody>
</table>
Summary: The observation made from the query tree processing was that at level L2 the time taken to perform the spatial operation is very long. Hence, it was proposed to optimize the query to improve the execution time.

Query tree optimization: The query optimization involves reorganisation of the query structure shown in figure 5.8. In the optimized query tree for sample query 2, the levels L1 and L4 remain unchanged when compared to the query tree shown in figure 5.7. The major changes noted in the query optimization tree are at levels L2 and L3 as shown in the figure 5.8. At L2 the restriction (σ) operation is performed. As a result in L4, the geometry of parcel no.30 and the geometries of quality geometries of fioh table are used to perform a spatial join. The time taken to perform is significantly improved by the optimized query tree. There was a gain of about twenty-five minutes for the optimized query tree’s execution time shown in figure 5.8, when compared to the query tree’s (see figure 5.7) execution time.

Optimized query tree implementation: The implementation of the optimized query tree is explained using the figure 5.9 (Flowchart part-1) and figure 5.10 (Flowchart part-2).
Figure 5.8: Optimized query tree for sample query 2

**Flowchart part1:** The flowchart-part 1 comprises of seventeen steps as following:

**S1:** The first step accepts the input query string $qs$.

**S2:** The query string $qs$ is split into three parts by the functions: $\text{proj}(qs)$ which returns the value 'SELECT x.id, x.prec FROM' and stores the string in variable $p$; $\text{mid}(qs)$ returns the value 'parplh WITHQUALITY fioh x' and stores the string in variable $m$; and $\text{cod}(qs)$ returns the value 'WHERE x.id = 30' and stores the string in variable $c$.

**S3:** The variable $l$ is assigned the value 'parplh' which was returned by the function $\text{laye}(m)$, the variable $q$ is assigned the string 'fioh' which was returned by the function $\text{quae}(m)$ and the variable $r$ is assigned the string 'x' which was returned by the function $\text{res}(m)$.

**S4:** The built-in-function $\text{length}$ returns the total length of the string stored in the variable $\text{codstr}$.

**S5:** This step checks whether the condition part (WHERE clause) exists.
**S6:** If codstr is assigned the value 0 then the WHERE clause does not exists. Hence, there is no need for the optimization step to be performed. The procedure dia(l,q,r) is called to perform the join operation. The join is performed based on the behaviour type which once again depends on the type of quality table q. In our case, B2 behaviour will be executed as the quality table fioh is the feature-independent type.

**S7:** The last step in the query process is done by creating a view named as temp. The view is created by substituting the p and r values which were obtained in S2 and S3.

**S8:** If codstr is assigned the value 1 then the WHERE clause exists. Hence, the optimization step is required.

**S9:** This step looks for the string '.' in the WHERE clause. For example, the string 'WHERE x.id = 30' consists of one '.' in the 8th position. The maximum number of conditions allowed is two, one condition for layer l and another condition for quality q. For example, 'WHERE x.id = 30' AND x.prec ≥ 10 or 'WHERE x.id BETWEEN 30 AND 50 AND x.prec BETWEEN 0.1 AND 2.5 are allowed in the SQL extension. There was a conflict while identifying '.' in the string 'WHERE x.id BETWEEN 30 AND 50 AND x.prec BETWEEN 0.1 AND 2.5 as it contains three '.': at x.prec, 0.1 and 2.5. The conflict for '.' appearing between the numbers was resolved by checking the left hand side of the '.' was a number. If it is a number, it is ignored. Furthermore, conditions like 'WHERE x.id ≤ 30 AND x.id ≥ 50 AND x.prec ≤ 0.1 AND x.prec ≥ 2.5 have to be replaced by the BETWEEN AND operators as 'WHERE x.id BETWEEN 30 AND 50 AND x.prec BETWEEN 0.1 AND 2.5. The reason for replacement is that the string with ≤ and ≥ relational operators has four '.' which will perform optimization for the layer part only ('WHERE x.id ≤ 30 AND x.id ≥ 50) as the maximum '.' allowed is only two.

**S10:** The condition for '.' position exists both in the layer l and the query q, is checked.

**S11:** If the condition in S10 is FALSE then there exists only one condition for either layer or quality. The string processing mentioned in S11 in figure 5.9 is performed. The main function of the string processing is to identify layer l (parplh) and its corresponding condition.
qs := 'SELECT x.id, x.vacc
FROM parplh WITHQUALITY fioh x
WHERE x.id = 30'

p := proj(qs)
m := mid(qs)
c := cond(qs)

l:= laye(m)
q:= quae(m)
r:= res(m)

alpos1 := INSTR(c,al1,1,1)
alpos2 := INSTR(c,al1,1,2)
lcol := SUBSTR (al1,lcol)
la := STRUC(l,r)
lcolc := INSTR (la,lcol)

alstr1 := cod
dotpos := INSTR(alstr1,'.')
lrdotpart := SUBSTR(alstr1,dotpos+1)
ispacpos := INSTR(lrdotpart,' ')
lcol := SUBSTR (lrdotpart,1,ispacpos)
la := STRUC(l,r)
lcolc := INSTR (la,lcol)

sql1 := 'CREATE TABLE '||tl1|| 'AS'||' '||la||'
WHERE' ||' '||alstr1;
EXECUTE IMMEDIATE sql1;
sql2 := 'CREATE TABLE '||tq1|| 'AS'||' '||qa||'
WHERE' ||' '||alstr1
EXECUTE IMMEDIATE sql2;
sql3 := 'CREATE VIEW temp AS
'||p||' '|| r
EXECUTE IMMEDIATE sql3;

Figure 5.9: Flowchart part-1:optimized query tree implementation
**S12:** Likewise, the string processing is performed to identify quality \( fioh \) and its corresponding condition.

**S13:** The condition \( lcolc <> 0 \text{ AND } qcolc = 0 \) checks whether the condition exists for the layer \( parplh \) or quality \( fioh \).

**S14:** If the condition in S13 is **TRUE** then the condition for the layer is re-arranged by executing the dynamic SQL as shown in S14, figure 5.9. This step refers to the level L2 mentioned in the query tree optimization. In our case, S14 will execute the following sql:

```sql
CREATE TABLE tl1 AS
SELECT x.oid
FROM PFOH x
WHERE x.oid = 30;
```

**S15:** The procedure \( dia(tl1,q,r) \) is executed. Then the control transfers to S7 for the final process.

**S16:** If the condition in S13 is **FALSE** then the condition for the quality is re-arranged by executing the dynamic SQL as shown in S16, figure 5.9. This step refers to the level L2 mentioned in the query tree optimization.

**S17:** The procedure \( dia(l,tq1,r) \) is executed. Then the control transfers to S7 for the final process.

**Flowchart part-2:** The flowchart part-2 (figure 5.10) comprises the following twelve steps:

**S18:** If the condition in S10 is **TRUE**, then the control is transferred to step S18 (beginning of flowchart part-2). This step starts by assigning the \( q \) condition to the \( alstr2 \) variable. The \( NOTC \) variable stores the position of the \( NOT \) keyword. If the keyword is present the \( NOTC \) holds the value of 1 else the value is 0.

**S19:** This step checks the presence of \( NOT \) keyword. The check of the presence of the \( NOT \) keyword was done in Flowchart-part1 as there was only one condition (either condition on \( l \) or \( q \) exists). Here, the check is required as there are two conditions for both \( l \) and \( q \). Hence, the string manipulation needs to include
or exclude 5 positions (3 positions for the NOT keyword and 2 positions for space before and after the keyword) based on the presence of the NOT keyword either in the l condition or in the q condition or in both.

**S20:** If the condition in S19 is FALSE then the variable alstr2 which holds the condition of q truncates 5 positions from its current value. Then the control is transferred to S21.

**S21:** If the condition in S19 is TRUE then the control is transferred to this step. String manipulation for both alstr1 and alstr2 is performed in this step.

**S22:** The condition spacpos $<> 0$ checks whether there exists an empty space at the end of the string alstr1.

**S23:** If the condition in S22 is TRUE the empty space in string alstr1 is truncated.

**S24:** If the condition in S22 is FALSE the control is transferred to this step for further process. This step is similar to S11. The main function of the string processing is to identify layer l and its corresponding condition.

**S25:** Identical to step S4, the main function in this step is to perform string processing to identify quality q and its corresponding condition.

**S26:** The condition lcolc $<> 0$ AND qcolc $<> 0$ checks whether there exists conditions for both l and q. If the condition in S25 is FALSE then the control is transferred to S11.

**S27:** If the condition in S25 is TRUE then the control is transferred to this step. A table tl1 is created based on the condition of l.

**S28:** Similar to S25 a table tq1 is created based on the condition of q.

**S29:** The procedure dia is called and executed by providing the parameters tl1 , tq1 and r processed in earlier steps.
Figure 5.10: Flowchart part-2: optimized query tree implementation
5.4.5 Multiple table query

The next step in the implementation of the SQL extension is its capability to retrieve multiple tables at the same time. An example of a multiple-table query is shown in sample query 4. The table camph is queried in addition to table x which stores the result of a join operation performed between l (parplh) and q (fioh).

Sample query 4: List the parcel id and precision for Campbellfield region

```
SELECT x.id, x.prec
FROM   camph, parplh WITHQUALITY fioh x
WHERE  camph.geom ∩ x.qgeom
```

The SQL extension can query up to three tables simultaneously but no more due to the limitation of SQL which allows only two hundred characters in the SQL statement at a time for query processing. For example, the sample query 5 shown below involves two tables, x and y which retrieve prec = 5 for the region camph (Campbellfield).

Sample query 5:

```
SELECT x.id, MAX(x.prec) prec
FROM   parplh WITHQUALITY fioh x, camph y
WHERE  x.qgeom ∩ y.geom AND x.prec BETWEEN 0.5 AND 10
       GROUP BY x.id HAVING MAX(x.prec) = 5
```

OUTPUT: sample query 5

<table>
<thead>
<tr>
<th>ID</th>
<th>PREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>67630</td>
<td>5</td>
</tr>
<tr>
<td>71396</td>
<td>5</td>
</tr>
<tr>
<td>72609</td>
<td>5</td>
</tr>
<tr>
<td>73400</td>
<td>5</td>
</tr>
<tr>
<td>73913</td>
<td>5</td>
</tr>
<tr>
<td>73914</td>
<td>5</td>
</tr>
</tbody>
</table>
5.4.6 Working of multiple table query

Most of the steps in the multiple table processing are identical to the flow charts explained earlier. The new steps introduced are steps S7 and S8 shown in the figure 5.11. The steps S7 and S8 store the position of ‘,’ in the input query string. If there are no ‘,’ present in the input query string then it is a single table query. As a result, the steps in the flowchart shown in figure 5.11 are executed. If there is only one ‘,’ present in the input query string then it is a two-table query, and as a result the steps in the flowchart shown in figure 5.12 are executed. If there are two ‘,’ present in the input query string then it is a three-table query, and as a result the steps in the flowchart shown in figure 5.13 are executed.
qs := 'SELECT x.id, MAX(x.prec) 
FROM parplh WITHQUALITY fhoh x,camph y 
WHERE x.qgeom ∩ y.geom GROUP BY x.id 
HAVING MAX(x.prec) = 5

S2: gbc := rgb(qs);
S3: cajr := snjoin(gbc);
S4: cajr := conjo(gbc);
S5: wherec := LENGTH(cajr);
S6: midp := mid(qs); 
midp := trim(midp);
S7: cl := INSTR(midp,',',1,1);
S8: cl := INSTR(midp,',',1,2);
S9: IS 
cl <> 0 AND 
c2 <> 0
? YES
IS
cl <> 0
? NO
S10: IS
cl <> 0
? NO
S11: tno := 1; 
ml := midp; 
mlwqc := INSTR (ml,'WITHQUALITY');
S12: IS 
mlwqc <> 0
? YES
S13: tabt1 := ml;
S14: pro := PROJ(qs)
gbc := gbc(qs)
sgbc := LENGTH (gbc)
S15: IS 
sgbc <> 0
? NO
S16: gbc := ' ';
S17: IS 
NOT tabt1 = ' ' AND 
NOT tabt2 = ' ' AND 
NOT tabt3 = ' '
? YES
S18: IS 
NOT tabt1 = ' ' AND 
NOT tabt2 = ' '
? NO
S19: tabp := tabt1||','||tabt2||','||tabt3;
S20: IS 
JOCON <> ' '
? YES
S21: OUTWOJ(pro, tabp, jocon, gbc)
S22: OUTWOJ(pro, tabp, gbc)
STOP

Figure 5.11: Flowchart-Multiple table query-Part1
5.5 Conversion between models

The RSVQ model is flexible enough to store data in any one of the three data quality models PF (per-feature), FI (feature-independent) and FH (feature-hybrid). It is a matter of choice for the user to choose which model is most appropriate (see chapter 6 where experimentation with the different models leads to recommendations on which model to use in what circumstances). Therefore it must be possible to convert between different models (because if the decision is dependent on the user, perhaps the user’s requirements will change). Note that conversion between different models may lead to loss of information. In the conversions from PF to FH or FI, or between FH and FI there will be no loss of information, but the conversions from FH or FI to PF may involve loss of information (since any sub-feature variation will be lost).

One of the sponsors of this research, the Department of Sustainability and Environment (Victorian government department) had a requirement to assign the worst quality (maximum value of positional accuracy recorded across a parcel’s vertices) to each parcel. The quality data provided by the DSE (see section 5.2.2) was in the form of a point data and it was independent of the parcel table (parph). Thus, the requirement was to convert the feature-independent quality table precision (fioh) to a per-feature quality table. In standard SQL the following conversion query will be executed:

FI to PF conversion query in standard SQL:

```
SELECT a.id as oid, ROUND(MAX(b.prec),1) as prec
FROM    parph a, fioh b
WHERE    SDO_RELATE (a.geom, b.qgeom, mask = ’anyinteract’) = ’TRUE’
GROUP BY a.id
```

FI to PF conversion query using WITHQUALITY keyword:

```
SELECT x.id as oid, ROUND(MAX(x.prec),1) as prec
FROM    parph WITHQUALITY fioh x
WHERE    x.id = 30 GROUP BY x.id
```

Sample output of feature-independent quality for parcel 30.

<table>
<thead>
<tr>
<th>OID</th>
<th>QID</th>
<th>PREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>92</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.12: Flowchart-Multiple table query-Part2
CHAPTER 5. FORMAL MODEL IMPLEMENTATION

Figure 5.13: Flowchart-Multiple table query-Part3
30 134956 2.5
30 134959 2.5
30 133624 .5
30 239609 .5

Sample output after conversion
From: feature-independent quality
To: per-feature quality for parcel 30.

OID  PREC
------  --------
30    2.5

The conversion from feature-independent to per-feature model by applying the WITHQUALITY keyword was helpful for the DSE department to assign the worst quality (maximum value of positional accuracy assigned to the parcel’s vertices) to each parcel. The conversion between other models are shown below (note all conversion models can also be done with the WITHQUALITY keyword:

**FI to FH conversion query in standard SQL:**

```sql
SELECT a.id as oid, b.qid as qid, b.qgeom AS qgeom, b.prec as prec
FROM parplh a, fioh b
WHERE SDO_RELATE (a.geom, b.qgeom, mask = 'anyinteract') = 'TRUE'
```

**FH to PF conversion query in standard SQL:**

```sql
SELECT a.id as oid, ROUND(MIN(b.prec),1) as prec
FROM parplh a, fhoh b
WHERE a.id = b.oid GROUP BY a.id
```

**FH to FI conversion query in standard SQL:**

```sql
SELECT b.qid, b.qgeom as qgeom, b.prec
FROM parplh a, fhoh b
WHERE a.id = b.oid
```
PF to FI conversion query in standard SQL:

```
SELECT rownum as qid, a.geometry as qgeom, b.prec as prec
FROM   parplh a, pfoh b
WHERE  a.id = b.oid
```

PF to FH conversion query in standard SQL:

```
SELECT b.oid as oid, rownum as qid, b.prec as prec, a.geometry as qgeom
FROM   parplh a, pfoh b
WHERE  a.id = b.oid
```

5.6 User interface

Oracle forms 10g was used to develop a user interface. Although developing a user interface was not an objective of this research, a user interface was advantageous to simplify the process of querying required for the experimentation described in chapter 6. Oracle forms was a natural choice as it could understand PL/SQL better than other platforms (VB.NET, Delphi etc).

The user interface shown in figure 5.14 consists of the following controls:

C1: The query editor provides platform for the user to enter queries. The controls C7, C8 and C9 interact with the query editor. The editor can accept query containing up to two hundred characters at a time. In SQL, to run the sample query 5 the query string needs to be in the following format:

```
'SELECT x.id, MAX(x.prec) prec
  FROM parplh WITHQUALITY fhoh x, camph y
  WHERE SDO_RELATE (x.qgeom,y.geom'
  "'mask = anyinteract"'  = ' "'TRUE"'
  'GROUP BY x.id HAVING MAX(x.prec)=5'
```

The SQL format was very complex as it required pipe symbols for concatenation and special quotations in the query string. Whereas the sample query 5 in
the query editor C1 does not require pipelines and special quotations (see C1 in figure 5.14).

C2: A combo box control is provided to select a table from the list of layer tables stored in the database.

C3: A combo box control is provided to select a table from the list of quality tables stored in the database.

C4: The button WITHQUALITY inserts the keyword between the l and q.

C5: The Run button executes the query string present in the query editor.

C6: The View Results button invokes the Spatial Console software to display the graphical output of the query performed.

C7: The button Clear clears the contents of the query editor C1.
C8: The button **Clear DB** is used to clear all temporary tables created during the query execution.

C9: The **Load Query** button is used to load previously saved queries. This functionality is quite useful for the user to re-run the queries performed earlier.

C10: The text box control displays the start time of the query execution.

C11: The text box control displays the end time of the query execution.

C12: The button is used to close the form.

The interface developed with the controls is user-friendly. Developing queries in the query editor proves to be an easy task when compared to the use of SQL interface. The ability of the interface to provide a graphical display of the query results is another feature which reduces the user’s task of importing the result in another graphic package for visualization. Although the user interface is not particularly intuitive, it is helpful for interacting with the RSVQ model.

### 5.7 Summary

This chapter discusses the implementation of the RSVQ model in Oracle spatial. An industry case study was used to test the implementation. The database section explains the steps involved in the database design. An entity relationship diagram was used to explain the relationship of the data quality models. Existing and new query mechanisms were outlined in the chapter. A new keyword **WITHQUALITY** as an SQL extension was introduced in the new query mechanism. The working of the keyword was explained in detail with the help of flowcharts including query optimization. The conversion between data quality models was discussed by using the **WITHQUALITY** keyword. The interface developed using Oracle forms was to overcome limitations of SQL interface was also discussed. The evaluation of the RSVQ model is discussed in next chapter 6.
Chapter 6

Experimentation and Results

6.1 Introduction

This chapter focuses on the testing of the RSVQ (RDBMS for spatial variation in quality) model developed in chapter 5. The RSVQ model is 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 focus of the evaluation is on the dependence of computation time on input size rather than on the cost of each operation, which is implementation and machine dependent (Rigaux et al., 2002a). Therefore, the key feature of each experiment 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). The chapter briefly discusses experiments conducted to test the query execution time of per-feature, feature-independent and feature-hybrid data quality models under varying data set sizes. Section 6.2 illustrates the data used for the experiments. Section 6.3 describes the design of experiments. Section 6.4 presents the results of the experiments. The last section 6.5 draws conclusions from the experiments conducted.

6.2 Experimental data

The test data was classified into four types as: small (~ 8,000 records in total, ~ 2 MB total storage space), medium (~ 60,000 records in total, ~ 6 MB total storage space), large (~ 274,000 records in total, ~ 48 MB total storage space) and very large size (~ 1,200,000 records in total, ~ 90 MB total storage space).
Small data size: The small size data set covers the Jacana locality, which falls in the southern part of Hume LGA area as shown in figure 6.1. The parcel table for Jacana is `jacaparh` (layer), which has 798 records consuming 0.33 MB storage space. The quality tables are: `jacaph` (per-feature quality) containing 798 records occupying 0.07 MB storage space; `jacaflh` (feature-independent quality) containing 2,839 records occupying 0.52 MB storage space; and `jacaflhh` (feature-hybrid quality) containing 3,350 records occupying 0.59 MB storage space.

Medium data size: The medium size data set covers the Campbellfield locality, which falls in the south eastern part of Hume LGA area as shown in figure 6.1. The parcel table for Campbellfield is `camparh` (layer), which has 4,232 records occupying 2.10 MB storage space. The quality tables are: `campaph` (per-feature quality) containing 4,232 records occupying 0.13 MB storage space; `campafh` (feature-independent quality) containing 16,571 records occupying 0.98 MB storage space; and `campafhh` (feature-hybrid quality) containing 31,505 records occupying 2.10 MB storage space.

Large data size: The large size data set was formed by combining three localities, Sunbury, Wildwood and Bulla as shown in figure 6.1, termed here as western part. The parcels of Sunbury, Wildwood and Bulla were combined to form the
 parcel table *westparh* (layer) which has 13,617 records occupying 6.29 MB storage space. The quality tables are: *westpfh* (per-feature quality) containing 13,617 records occupying 0.26 MB storage space; *westfih* (feature-independent quality) containing 108,294 records occupying 17.83 MB storage space; and *westfhh* (feature-hybrid quality) containing 138,063 records occupying 23.07 MB storage space.

**Very large data size:** The very large size data set covers the whole Hume LGA area. The parcel table is *parph* (layer) which has 75,445 records with 28.31 MB storage space. The quality tables are: *pfoh* (per-feature quality) containing 75,445 records occupying 2.10 MB storage space; *fioh* (feature-independent quality) containing 370,407 records occupying 19.92 MB storage space; and *fhoh* (feature-hybrid quality) containing 678,179 records occupying 39.85 MB storage space.

### 6.2.1 Storage analysis

The number of records in the layer and quality tables for small, medium, large and very large data sets were plotted using the bar graph shown in figure 6.2. The x-axis of the graph is divided into Jacana (small size data), Campbellfield (medium size data), Western part (large size data) and Hume LGA (very large size data).
size data). Each division in the $x$-axis is further subdivided into four categories to represent the number of records in each of the parcel (layer) and the three quality tables: per-feature, feature-independent and feature-hybrid. The $y$-axis is labeled with the number of records. The $y$-axis uses the logarithm of the number of records, because the data covers a range spanning three orders of magnitude, from hundreds to hundreds of thousands of records in each table. The number of records for each table is labeled at the top of each graph.

**Per-feature quality storage:** The tables: jacapfh, camppfh, westpfh and pfoh have been modeled as per-feature quality. The number of records in the per-feature quality tables is equal to the number of records in the parcel (layer) table for all the four data sets. Each record in the per-feature quality table has a unique link to its corresponding record in the parcel table which results in equal record size for both the layer and quality tables in the graph shown in figure 6.2. The number of records for the per-feature quality tables have fewer records when compared to the other quality tables. Similarly, the storage space for per-feature model (2.56 MB) for all the four data sets is less when compared to feature-independent (39.26 MB) and feature-hybrid model (65.60 MB). Thus, per-feature model is more efficient in storage. While the per-feature quality model is efficient in storage mechanism, it does not support sub-feature variation due to its limitation in representational capabilities.

**Feature-independent quality storage:** The tables: jacafih, campfih, westfih and fioh have been modeled as feature-independent quality. The number of records in the feature-independent quality table is greater than the per-feature quality table as shown in figure 6.2. Similarly, the storage space consumed by the feature-independent model for all four data sets (39.26 MB) is greater than per-feature storage (2.56 MB). The increase in storage space is because the feature-independent model represents sub-feature variation, so each feature may be associated with multiple quality records. Thus, the feature-independent model requires more storage space when compared to the per-feature model as it represents sub-feature variation.

**Feature-hybrid quality storage:** The tables: jacafhh, campfhh, westfhh and fioh has been modeled as feature-hybrid quality. The feature-hybrid table has the greatest number of records when compared to the other quality table as shown in figure 6.2. Similarly, the storage space for feature-hybrid model (65.60 MB) is greatest when compared to per-feature model (2.56 MB) and feature-independent (39.26 MB). This is because the feature-hybrid model incorporates redundancy in
storage, where a quality record may be represented more than once. In our specific example, the feature-hybrid quality table has been generated by overlaying the parcel table over feature-independent table (see figure 6.3). So, for example the precision point 2.5m shown in figure 6.3 is stored for parcel 1001 and parcel 1002. Thus, the quality record for precision point 2.5m is stored twice (redundant) which increases the storage size of the model. While the feature-hybrid model consumes most space when compared to other data quality models, it represents sub-feature variation efficiently providing a facility for faster data retrieval.

6.3 Experiment design

There are five variables considered in the design of the experiment on query efficiency: query type, data models, data set size, approach and execution time of each approach (see figure 6.4).

Test queries: The aim of the test queries was to test the WITHQUALITY operator. Although many data quality queries were implemented using the RSVQ model (see chapter 5), the following basic quality queries were selected for the test: a) query with WITHQUALITY operator; b) query with WITHQUALITY operator and restriction on quality table; c) WITHQUALITY operator and restriction on parcel table; d) WITHQUALITY operator, restriction on parcel and quality ta-
CHAPTER 6. EXPERIMENTATION AND RESULTS

Figure 6.4: Variables for experiment

ble; and e) WITHQUALITY operator and restriction on a spatial window. Based on the above classifications, five queries Q1 to Q5 were designed as: Q1) List the precision for all the parcels (a); Q2) What precision information is associated with a parcel id? (b); Q3) List the parcels which have a particular precision value (c); Q4) List the parcels having ids above a particular value and a particular precision value (d); and Q5) What is the precision over a region with a particular precision value? (e).

The test queries Q1 to Q5 have been translated to relational algebra expressions in table 6.1 to highlight the underlying structure of non-optimised and optimised query execution. The corresponding behaviours executed for each query have also been listed alongside of the relational algebra expression in the table 6.1.

Non-optimised query: The non-optimised query is executed as per the SQL statement.

Optimised query: 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 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 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Query type</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF</td>
<td>Non-opt</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B3</td>
</tr>
<tr>
<td></td>
<td>l &lt; q</td>
<td>σ(l &lt; q)</td>
<td>σ(l &lt; q)</td>
<td>σ(l &lt; q)</td>
<td>σ(w &lt; q)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Opt</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B3</td>
</tr>
<tr>
<td></td>
<td>l &lt; q</td>
<td>σ(l &lt; q)</td>
<td>l &lt; σ(q)</td>
<td>σ(l &lt; q)</td>
<td>σ(w &lt; q)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>l &lt; q</td>
<td>σ(l &lt; q)</td>
<td>σ(l &lt; q)</td>
<td>σ(l &lt; q)</td>
<td>σ(w &lt; q)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>l &lt; q</td>
<td>σ(l &lt; q)</td>
<td>l &lt; σ(q)</td>
<td>σ(l &lt; q)</td>
<td>σ(w &lt; q)</td>
<td></td>
</tr>
<tr>
<td>FH</td>
<td>Non-opt</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B2</td>
</tr>
<tr>
<td></td>
<td>l &lt; q</td>
<td>σ(l &lt; q)</td>
<td>σ(l &lt; q)</td>
<td>σ(l &lt; q)</td>
<td>σ(w &lt; q)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Opt</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B2</td>
</tr>
<tr>
<td></td>
<td>l &lt; q</td>
<td>σ(l &lt; q)</td>
<td>l &lt; σ(q)</td>
<td>σ(l &lt; q)</td>
<td>w &lt; σ(q)</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Experiment design

**Query Q1:** The query “List the precision value associated for all the parcels” is a basic quality query which performs a WITHQUALITY operation between the parcel (layer) and the quality tables. The WITHQUALITY operator will execute the B1 behaviour (natural join) for the per-feature and feature-hybrid quality tables and the B2 behaviour (spatial join) for the feature-independent quality table. The very large size data set is shown as an example in the SQL structure for illustration purpose. Similar queries were also implemented for the other three data sets.

```
SELECT *
FROM parplh WITHQUALITY pfoh x;
```

**Query Q2:** The query “What precision information is associated with parcel id?” is based on a condition over the parcel table. The WITHQUALITY operator will execute the B1 behaviour for per-feature and feature-hybrid quality tables and the B2 behaviour for the feature-independent quality table to list the precision value for a given parcel.
CHAPTER 6. EXPERIMENTATION AND RESULTS

SELECT x.prec
FROM parplh WITHQUALITY pfoh x
WHERE x.id = 30;

**Query Q3:** The query “List the parcels which have a particular precision value.” is based on a condition over the quality table. The WITHQUALITY operator will execute the B1 behaviour for per-feature and feature-hybrid quality tables and the B2 behaviour for the feature-independent quality table to list the parcels for a given precision value.

SELECT x.id
FROM parplh WITHQUALITY pfoh x
WHERE x.prec = 0.5;

**Query Q4:** The query “List the parcel ids above a particular value and a particular precision value.” is based on a condition over the layer table and the quality table. The WITHQUALITY operator will execute the B1 behaviour for per-feature and feature-hybrid quality tables and the B2 behaviour for the feature-independent quality table to list the parcels and the precision value.

SELECT x.id,x.prec
FROM parplh WITHQUALITY pfoh x
WHERE x.id > 15000 AND x.prec = 2.5;

**Query Q5:** The query “What is the precision over a region with a particular precision value?” is a complex query. The query involves a spatial region win4h (window query) in addition to the condition on quality table makes this query complex when compared to others. The WITHQUALITY operator will execute the B3 behaviour (natural and spatial join) for the per-feature quality table and the B2 behaviour for feature-hybrid quality and feature-independent quality tables.

SELECT x.prec
FROM win4h WITHQUALITY pfoh x
WHERE x.prec = 2.5;

6.4 Query evaluation

The evaluation consisted of executing the non-optimised and optimised queries mentioned in the table 6.1. Since query execution times vary due to processor
(CPU) tasks at the particular time, each query was repeated five times and the statistical distribution of query times was recorded. Only five execution times were recorded, because some queries were very slow and took about up to an hour to complete. All the five queries (Q1 to Q5) were repeated the same number of times for all experiments to provide better comparison. The graphs shown in figure 6.5, figure 6.7 and figure 6.8 depict the results of query execution using non-optimised and optimised approach. The $x$-axis of the graphs is labeled with the data set size. The $y$-axis of the graphs is labeled with the time taken to execute the query. The $y$-axis for some graphs is labeled with seconds whereas the $y$-axis of other graphs is labeled with minutes. The average of the five query execution times was used to plot the average non-optimised and optimised graph for each query. The non-optimised query execution time is denoted by a square symbol. The average optimised query execution time is denoted by a triangle symbol. Two types of curve shapes are observed in the above mentioned graphs: linear and non-linear. The linear curve shape (eg., see figure 6.5d optimised curve shape) is more readily scalable (ie., efficiency as a function of input size) than non-linear curve shapes (see figure 6.5d non-optimised curve shape). In addition to measuring the differences in absolute query execution times, the experiments also investigate scalability in terms of linear and non-linear response curves.

Hypothesis testing was used to test whether the differences between the optimised and non-optimised query execution times were statistically significant. The null hypothesis ($H_0$) assumed that there was no difference between the non-optimised and optimised query execution times. The alternative hypothesis ($H_1$) was that there was a significant difference between these query execution times. Because the sample sizes were small (five observations per sample), a t-test for two samples assuming unequal variance was used for the hypothesis test. The following hypotheses were tested (where $\mu_n$ is the mean of the non-optimised and $\mu_o$ is the mean of the optimised query execution times):

- $H_0$: There is no difference between the means of the non-optimised and optimised query execution time ($\mu_n = \mu_o$);
- $H_1$: There is a difference between the means of the non-optimised and optimised query execution time ($\mu_n < \mu_o$ or $\mu_o < \mu_n$).

The table 6.2 summarizes the results of the t-test carried out for all the three data quality models (per-feature (PF), feature-independent (FI) and feature-hybrid (FH)). The column “data quality model” in table 6.2 consists of data quality models. The column headings Q1 to Q5 in the table 6.2 denote the five test queries discussed in the previous section. In the Q1 column of table 6.2 the
value NS (Not Significant) denotes that the difference between the optimised and non-optimised query execution time was not statistically significant \( (\mu_n = \mu_o) \). Similarly, the value Non-opt (Non-optimised) in the table 6.2 denote that the difference between the optimised and non-optimised query execution time was statistically significant \( \mu_n < \mu_o \). Thus, in this case non-optimised query execution time is significantly better than optimised query execution time. The value Opt (Optimised) in the table 6.2 denotes that the difference between the optimised and non-optimised query execution time was statistically significant \( \mu_o < \mu_n \). Thus, in this case optimised query execution time is significantly better than non-optimised query execution time. The reasons for the differences in table 6.2 are discussed in more detail in the following sections.

<table>
<thead>
<tr>
<th>Data quality model</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF</td>
<td>NS*</td>
<td>Non-opt</td>
<td>Non-opt</td>
<td>Non-opt</td>
<td>Non-opt</td>
</tr>
<tr>
<td>FI</td>
<td>NS</td>
<td>Opt</td>
<td>Opt</td>
<td>Opt</td>
<td>Opt</td>
</tr>
<tr>
<td>FH</td>
<td>NS</td>
<td>Non-opt</td>
<td>Non-opt</td>
<td>Non-opt</td>
<td>Opt</td>
</tr>
</tbody>
</table>

Table 6.2: Results of the experiments. NS: Not Significantly different, Non-opt: Non-optimised approach faster, Opt: Optimised approach faster, NS* indicates 3 out of 4 data sets were NS.

### 6.4.1 Per-feature model evaluation

**Per-feature graph (a):** The graph shown in figure 6.5a depicts the results of query Q1 execution using non-optimised and optimised approaches (corresponding to the cells in rows 1 and 2 of the table 6.1). The shape of the non-optimised and optimised curves are approximately linear, with query time increasing slightly with increasing data size. The optimised queries appear to require slightly longer to execute than the corresponding non-optimised queries. However, the total time taken to execute any query is less than 1/10th of a second. The \( B1 \) behaviour is executed for both optimised and non-optimised queries.

A t-test indicated that the difference between \( \mu_n \) and \( \mu_o \) is not significant (see table 6.2 row 1 of Q1 column) at the 99% level for all data set sizes except the medium-sized data set. Consequently the null hypothesis \( (H_0) \) is accepted in those cases, leading to the inference that there is no difference between optimised and non-optimised query execution times. For the medium-sized data set (see NS* in table 6.2), the null hypothesis is rejected indicating that the optimised query execution time is significantly slower than the non-optimised query execution time. Although no actual optimization is performed on this query, the pre-optimization checks require a small additional overhead to process. However, the magnitude of the difference between the optimised and non-optimised queries
6.4. QUERY EVALUATION

Figure 6.5: Per-feature model query evaluation
is very small, a fraction of one second, and so while statistically significant has negligible impact on the efficiency of this query.

**Per-feature graph (b):** The graph shown in figure 6.5b depicts the results of query Q2 execution (Q2 column in table 6.1). The shape of the non-optimised curve is approximately linear, and independent of the data set size. The shape of the optimised curve exhibits shallow exponential growth increasing with data set size, and each query requires more time to execute than the non-optimised query. The B1 behaviour is executed for both optimised and non-optimised queries.

Again, a t-test was used to investigate whether the apparent differences between non-optimised and optimised query times were statistically significant. The null hypothesis ($H_0$) was that there is no difference between the mean of the non-optimised and optimised query execution time ($\mu_n = \mu_o$); the alternative hypothesis ($H_1$) is that there is a difference between the mean of the non-optimised and optimised query execution time ($\mu_n < \mu_o$ or $\mu_o < \mu_n$). The t-test indicated that $\mu_n < \mu_o$ significant at the 99% level for all data set sizes, leading to the conclusion that the non-optimised query execution was faster than optimised query execution (see table 6.2 row 1 of Q2 column).

The increase in time for optimised query execution can be ascribed to a *subset table check (STC)* required by the WITHQUALITY operator. The STC step checks whether the layer table is a subset of the layer table stored in the metadata table (see table 5.5). In other words, STC performs the cardinality check so that the cardinality of the join cannot be any greater than the cardinality of the cartesian product (Connolly et al., 1999). The flowchart in figure 6.6 shows the steps involved in STC. The step S6 in the flowchart uses an UNION operator as a first step. The step S8 uses a spatial join based on CONTAIN or EQUAL mask to accomplish the second step. The step S8 consumes more time to execute as it involves a spatial join. This step must be performed because the Oracle UNION operator does not support spatial geometries. Step S9 performs the cardinality check based on the inputs from step S6 and S8. Since the optimised query execution must perform the STC, the time taken to execute the query is slower when compared to non-optimised query execution time.

The time taken to perform the STC takes about a minute for the large data set in addition to the query execution time which can be clearly seen in figure 6.5b. Thus, if the number of features is increased, then the time taken to perform STC will also be more. Further, the increase in time is non-linear, leading to much lower scalability for optimised queries than non-optimised queries in this case.
START

\[ l, q \]

\[ \text{tab} := \text{mq}(q) \]

IS 
\[ \text{tab Not NULL} \]
YES 
\[ \text{Return Null} \]
NO

\[ \text{sql1} := \text{'SELECT count (ID) FROM ' || l;} \]
\[ \text{EXECUTE IMMEDIATE sql1 INTO c1;} \]

\[ \text{--Check for union compatible for non geometric attributes} \]
\[ \text{sql2 := 'SELECT ID FROM ' || tab} \]
\[ \text{UNION SELECT ID FROM ' || l;} \]
\[ \text{EXECUTE IMMEDIATE sql2;} \]

\[ \text{sql3 := 'SELECT count (ID) FROM (' || sql2 || ')} \]
\[ \text{EXECUTE IMMEDIATE sql3 INTO c2;} \]

-- sql4 checks for union compatible and geometries and stores the count value in c3
\[ \text{sql4 := 'SELECT COUNT (b.ID) FROM tab a, l b} \]
\[ \text{WHERE SDO_RELATE (a.geom,b.geom,} \]
\[ \text{'mask = CONTAIN' = 'TRUE'} \]
\[ \text{OR} \]
\[ \text{SDO_RELATE (a.geom,b.geom,} \]
\[ \text{'mask = EQUAL' = 'TRUE'} ; \]
\[ \text{EXECUTE IMMEDIATE sql4 INTO c3;} \]

A

IS 
\[ c1 <= c2 \text{ AND } c3 <> 0 \]
YES 
\[ \text{Return 1} \]
NO

STOP

Figure 6.6: Subset table check flowchart
**Per-feature graph (c):** The graph shown in figure 6.5c depicts the results of query Q3 execution (Q3 column in table 6.1). The shape of the non-optimised curve is linear. The shape of the optimised curve is also approximately linear. The $B_1$ behaviour is executed for both optimised and non-optimised queries.

A t-test indicated that the difference between $\mu_n$ and $\mu_o$ is significant at the 99% level for all data set sizes, leading to the conclusion that the non-optimised query execution is faster than optimised query execution (see table 6.2 row 1 of Q3 column). Similar to the per-feature graph (b), the increase in time for optimised query execution can be ascribed to the STC step required by the $B_1$ behavior for this type of query.

**Per-feature graph (d):** The graph shown in figure 6.5d depicts the results of query Q4 execution (Q4 column in table 6.1). The shape of the non-optimised curve is approximately linear. The shape of the optimised curve is non-linear.

A t-test indicated that the difference between $\mu_n$ and $\mu_o$ is significant at the 99% level for all data set sizes, leading to the conclusion that the non-optimised query execution time is better than optimised (see table 6.2 row 1 of Q4 column). It is noted the optimised query execution times are strongly non-linear, exhibiting much slower execution times than comparable non-optimised queries for large and very large data sets. This difference is again due to the STC step performed for optimised queries. The pre-selection steps S16 and S17 of flowchart1, section 5.4.4 of the optimised query create a large number of features to be checked by the STC step. Since the time taken to perform the STC depends on the number of features, the difference in query execution time is ascribed to the STC step.

**Per-feature graph (e):** The graph shown in figure 6.5e depicts the results of query Q5 execution (Q5 column in table 6.1). The shapes of the non-optimised and optimised curves are non-linear due to the $B_3$ behaviour execution.

A t-test indicated that the difference between the optimised and non-optimised query execution time is significant at the 99% level. Consequently the null hypothesis ($H_0$) is rejected in this case, leading to the conclusion that the non-optimised query execution is faster (see table 6.2 row 1 of Q5 column). Again the optimised query execution is slower due to the STC step.

### 6.4.2 Feature-independent model evaluation

**Feature-independent graph (a):** The graph shown in figure 6.7a depicts the results of query Q1 execution using non-optimised and optimised approaches.
Figure 6.7: Feature-independent model query evaluation
(Q1 column in table 6.1). The shapes of the non-optimised graph and optimised graphs are non-linear. The B2 behaviour is executed for both non-optimised and optimised queries. A t-test indicated that the difference between $\mu_n$ and $\mu_o$ is not significant at the 99% level for all data set sizes. Consequently the null hypothesis ($H_0$) is accepted, leading to the inference that there is no difference between optimised and non-optimised query execution times.

**Feature-independent graph (b):** The graph shown in figure 6.7b depicts the results of query Q2 execution using non-optimised and optimised approaches (Q2 column in table 6.1). The shapes of the non-optimised and optimised graphs are non-linear. The B2 behaviour is executed for both non-optimised and optimised queries. A t-test indicated that $\mu_o < \mu_n$, significant at the 99% level for all data set sizes. Thus optimised query execution is faster than the non-optimised query execution.

Pre-selecting the geometries (few records) of the parcel table (see section 5.4.4, flowchart1 step S14 and S15) before applying the WITHQUALITY operator with the quality table makes the optimised query execute faster than the non-optimised query, since this reduces the number of geometries involved in the spatial join.

**Feature-independent graph (c):** The graph shown in figure 6.7c depicts the results of query Q3 execution using non-optimised and optimised approaches (Q3 column in table 6.1). The shape of the graph is similar to the graph in figure 6.7b including the t-test results and the B2 behaviour executed.

The pre-selection steps S16 and S17 of flowchart1, section 5.4.4 makes optimised query execution time faster when compared to the non-optimised query execution time.

**Feature-independent graph (d):** The graph shown in figure 6.7d depicts the results of query Q4 execution using non-optimised and optimised approaches (Q4 column in table 6.1). The shape of the graph, the B2 behaviour executed and the t-test results are similar to graph b and c.

The pre-selection steps S27, S28 and S29 of flowchart2, section 5.4.4 make the optimised query execution time faster when compared to the non-optimised query execution time.

**Feature-independent graph (e):** The graph shown in figure 6.7e depicts the results of query Q5 execution using non-optimised and optimised approaches (Q5
column in table 6.1). Similar to the other graphs of figure 6.7, the shape of the curves for optimised and non-optimised query execution time, the B2 behaviour and t-test results are same.

The optimised query execution time is faster when compared to the non-optimised query execution time as a result of performing a join operation after pre-selecting the geometries (few records) of the quality table with the parcel table (see section 5.4.4, flowchart 1 step S16 and S17).

6.4.3 Feature-hybrid model evaluation

Feature-hybrid graph (a): The graph shown in figure 6.8a depicts the results of query Q1 execution using non-optimised and optimised approaches. The shape of the non-optimised and optimised curves are linear, with query execution time increasing with increasing data size. The B1 behaviour is executed for both non-optimised and optimised queries.

A t-test indicated that the difference between $\mu_n$ and $\mu_o$ is not significant (see table 6.2 row 3 of Q1 column, as the time difference is very small, hence not significant) at the 99% level for all data set sizes. The pre-optimization check discussed for figure 6.5 of per-feature graph (a) is the reason for optimised query execution time to be slower than non-optimised timing even though no optimisation was performed.

Feature-hybrid graph (b) and (d): The graphs (b) and (d) are similar to 6.5b and d. Hence, the shape of the graph, results of the t-test and the reason for non-optimised query execution time to be faster than optimised timing are also similar to figure 6.5b and d of per-feature graph. The B1 behaviour is executed for both non-optimised and optimised queries.

Feature-hybrid graph (c): The shape of the non-optimised curve in the graph is approximately linear. The shape of the optimised curve in the graph is non-linear. The B1 behaviour is executed for both non-optimised and optimised queries.

The t-test indicated that $\mu_n < \mu_o$, which is significant at the 99% level for all data set sizes, leading to the conclusion that the non-optimised query execution time was faster than optimised query execution time (see table 6.2 row 3 of Q3 column). The reasons for optimised query being slower than non-optimised and the substantial difference in execution time between large and very large data
CHAPTER 6. EXPERIMENTATION AND RESULTS

Figure 6.8: Feature-hybrid model query evaluation

Legend
- Non-optimised
- Optimised

c: Q5 (B2 behaviour)
sets are ascribed to STC step, which was discussed for figure 6.5b of per-feature graph.

**Feature-hybrid (e):** The graph shown in figure 6.8e depicts the results of query Q5 execution using non-optimised and optimised approaches. The shape of the graph is non-linear for both the optimised and non-optimised query execution time, due to the $B_2$ behaviour execution as in figure 6.7e. Similarly, the results of the t-test are as for figure 6.7e which leads to the conclusion that the optimised query execution time was faster. The pre-selection of the geometries of the quality table performed in the optimisation query makes the query execution faster when compared to the non-optimised query.

### 6.4.4 Query evaluation discussion

**Per-feature query evaluation summary:** The overall assessment of the per-feature data model indicates that the non-optimised query execution time is faster than the optimised query execution time. There are two reasons for this undesirable result. Firstly, the slow query execution of the optimised query can be ascribed to the *subset table check* (STC), which performs a cardinality check for the input tables. Secondly, the queries Q1 to Q4 for the per-feature model execute the $B_1$ behaviour, which constitutes a natural join. Oracle uses efficient indexes to handle natural joins that automatically make non-optimised queries faster to execute than optimised queries.

**Feature-independent query evaluation summary:** The evaluation led to the conclusion that the optimised query execution time was faster than the non-optimised query execution. The optimised approach is especially efficient at handling large and very large data set, for example in figure 6.7b the non-optimised query takes about 40 minutes to execute very large data set whereas the optimised query takes about 9 minutes. The optimised query is expected to be even more beneficial if the quality for the entire state of Victoria is queried (note, the very large data set represents a local city council only). Thus, optimised query execution will have increasing benefit when large size data sets are queried. However, a query time of minutes still makes the optimised queries not suitable for interactive user queries.

**Feature-hybrid query evaluation summary:** The evaluation concludes that the non-optimised query execution time was faster than optimised query execution for queries Q1 to Q4, which executed the $B_1$ behaviour. The behaviour $B_1$
executed by the queries Q1 to Q4 is similar to the behaviour executed by the per-feature queries Q1 to Q4. Although the same behaviour was executed by the per-feature and the feature-hybrid queries Q1 to Q4, the time difference between the optimised and non-optimised query execution is higher for the feature-hybrid queries when compared to the per-feature queries. The reason for feature-hybrid queries taking more time to execute when compared to the per-feature queries is that the feature-hybrid queries retrieve sub-feature variation (more than one quality associated with a feature). Unlike other queries Q1 to Q4, query Q5 executed the \textit{B2} behaviour. As a result the optimised query execution was faster than non-optimised query. It is also noted that the feature-hybrid query Q5 takes about seven minutes longer to execute than the feature-hybrid query Q5 because the feature-hybrid model has more features when compared to the feature-independent model.

### 6.4.5 Re-engineering the model

The query evaluation revealed that the queries which executed the \textit{B2} behaviour (spatial join) proved to be efficient for optimisation, whereas the queries which executed the \textit{B1} behaviour (natural join) and the \textit{B3} behaviour (spatial and natural join) proved to be more efficient using non-optimised queries. Comparing tables 6.1 and 6.2 it is clear that only those queries that execute the \textit{B2} behaviour benefit from optimisation.

In order to make the RSVQ model handle queries efficiently, the optimised algorithm was modified by incorporating a new function called \textit{FOB} see figure 6.9. The function \textit{FOB} (see step S4 in the flowchart) returns the behaviour of the queries based on the inputs \textit{l} (layer table) and the quality table \textit{q}. If the function \textit{FOB} returns \textit{B2} (see step S5) the query is optimised. If the function \textit{FOB} returns either \textit{B3} (see step S9) or \textit{B1} (see step S12) then the query is executed without optimisation. Thus, the newly incorporated function \textit{FOB} by pre-checking the behaviours helps the RSVQ model to decide whether to perform query optimisation or not. Thus, re-engineered RSVQ model can achieve the fastest query execution for all queries, whether \textit{B2} or \textit{B1} or \textit{B3} behaviours.

### 6.5 Summary and conclusions

The storage evaluation indicates that the per-feature model consumes less storage space when compared to feature-independent and feature-hybrid models as it does not support sub-feature variation. The feature-independent model con-
Figure 6.9: Re-engineered RSVQ flowchart with new function FOB
sumes more storage space when compared to per-feature and less storage space when compared to feature-hybrid model and represents sub-feature variation. The feature-hybrid consumes the most storage space when compared to the other models and represents sub-feature variation.

The RSVQ model allows all three data quality models to be represented in a single database (integrated approach). The integrated approach provides flexibility to model any spatial data quality elements irrespective of the element’s having spatial variation within a feature (sub-feature variation) or within a layer.

The query evaluation indicates that the WITHQUALITY operator introduced in this research as an extension to standard SQL can be effective in handling queries efficiently irrespective of natural join, spatial join, and combination of both spatial and non-spatial joins. Thus, the RSVQ model has good scalability efficiency characteristics, allowing a balance between storage and query efficiency. It also provides flexible representation to model spatial variation of quality in any of the related models (per-feature, feature-independent and feature-hybrid). This flexibility is the basis for allowing data producers and users effective control over data quality storage. The combination of efficiency and effectiveness indicates the model can answer the research question stated in the first chapter.
Chapter 7

Conclusions and recommendations

7.1 Research contributions

The motivation for the research was a real world example of how to represent spatially varying quality including sub-feature variation existing in the cadastral data produced by the Department of Sustainability and Environment, Victoria, Australia (DSE, the government department that is responsible for maintaining and updating spatial data for the state of Victoria). Although there were related studies carried out to support spatial variation in quality none of them have adequately researched to address the efficiency of storage and retrieval mechanisms to represent variation in quality within an object. The fundamental contributions of this work is the development of an integrated model for storage and querying of spatial data quality, founded on widely used relational database theory and technology, in addition to empirical evaluations of the efficiency of the approach. With these key points the research question and hypothesis were framed as:

**Research question:** “How do we efficiently and effectively model spatial variation of data quality in spatial databases?”

**Research hypothesis:** “Existing relational spatial database technology can be used to model and evaluate spatial variation including sub-feature variation of spatial data quality.”
The original contributions of the research are listed below, which address the research question and the hypothesis, and meet the objectives set in the first chapter:

**Identified three data quality models:** The research identified three data quality models: per-feature, feature-independent and feature-hybrid model to represent spatial variation of quality (objective 1 met, see chapter 3). The sub-feature variation of data quality is represented by the feature-independent and feature-hybrid model excluding the per-feature model.

**Developed RSVQ data model:** The research developed a formal data model using relational algebra called, “RDBMS for spatial variation in quality (RSVQ)” to integrate the above data quality models into one database. The RSVQ is flexible enough to represent existing elements of spatial data quality, such as accuracy, precision, lineage, etc., (objective 2 and 3 met, see chapter 4).

**Developed a new WITHQUALITY keyword to query quality:** The research developed a new keyword WITHQUALITY as an extension to standard SQL to query spatially varying quality based on three behaviours: B1 (spatial join), B2 (natural join) and B3 (combination of spatial and natural join) (Objective 4 met, see chapter 5). The WITHQUALITY keyword:

- supports all functionality of standard SQL, hence no need to learn a new syntax to use the SQL extension;
- retrieves spatial variation of data quality irrespective of which data model is used to store it;
- can be optimised in some cases to reduce query execution time;
- provides the functionality to query multiple quality information tables (for example, retrieve lineage and positional accuracy at the same time);
- provides the functionality to convert between data quality models (for example, feature-independent to per-feature data model).

**Evaluated the RSVQ model:** The research evaluated real world data provided by DSE. The storage and the query efficiency of the RSVQ data model were tested by analytical (relational algebra) and experimental (Oracle Spatial) methods (objective 5 met, see chapter 6).

From the above mentioned research contributions it can be concluded that
• By implementing the RSVQ (RDBMS for spatial variation in quality) model, spatial variation in data quality can be represented efficiently and effectively. Thus, the RSVQ data model developed in this research is the answer to the research question.

• Existing relational spatial database technology can be used to model and evaluate spatial variation, including sub-feature variation of spatial data quality confirms the research hypothesis.

7.2 Recommendations

7.2.1 How to update quality in RSVQ?

One question that has not been directly addressed by this research is the updating of data in the RSVQ. To update data quality information in the per-feature model, a row containing data quality can simply be inserted, deleted or modified based on the parcel identifier ID which maps to the quality identified OID of the per-feature quality table. In case of feature-independent quality updating, the row containing data quality information can be updated based on the quality identifier QID in the feature-independent quality table. However, while updates to per-feature and feature independent are straightforward, updating data stored in the feature-hybrid model is more complex. The updating of the feature-hybrid model will be based on QID and OID. The OID maps to the parcel identifier ID. In other words, updating of the feature-hybrid quality will involve a spatial join performed with the new parcel’s information.

Furthermore, the frequency of updates required also has an impact on the data quality models. If the quality information is updated less frequently then performing a spatial join is a one time cost for the feature-hybrid model. Conversely, if the quality information requires frequent updates then the per-feature and the feature-independent models are efficient for updating data quality.

7.2.2 Which data quality model to use?

Deciding which data quality model to use will be application specific. There are four parameters to be considered when choosing a data quality model:

• sub-feature variation;

• storage space;
• querying; and

• updating

Choosing per-feature model  The per-feature model can be selected if the quality information requires no sub-feature variation representation, less storage space, interactive querying and frequent updates.

Choosing feature-independent model  The feature-independent model can be selected if the quality information requires sub-feature variation representation and frequent updates. With regards to querying, the model is less efficient when compared to per-feature and feature-hybrid. The consumption of storage space by the feature-independent is more when compared to per-feature model and less when compared to feature-hybrid model.

Choosing feature-hybrid model  The feature-hybrid model can be selected if the quality information requires sub-feature variation representation, rapid querying and the frequency of updates is less. With regards to storage space, the feature-hybrid model uses most space when compared to the per-feature and feature-independent model.

Thus no model is efficient in satisfying all the four parameters listed above. The RSVQ model is flexible enough to allow users to balance between sub-feature variation representation, storage space, querying and updating.

7.3 Directions for further research

Investigate other ways to implement the feature-hybrid model:  The research identified two options for designing the feature-hybrid model. The first option was to use the linear referencing approach. Each linear feature was assumed to have a unique identifier. In addition to the feature attribute table, an additional table, called an event table was used to store the quality values. Using linear referencing, modification of quality information becomes computationally straightforward, since updates only need to alter the event table. Linear referencing was computationally inexpensive as the quality was stored in a non-spatial table and the data retrieval was quick as natural joins were performed. Since, linear referencing offers limited representational capabilities with respect to polygon features the second option was implemented.
The second option was to create a feature-hybrid quality table by performing a spatial join on the parcel layer table and feature-independent quality table. This option had high storage cost but efficient data retrieval mechanism. The updating of quality was not straightforward in this option because, first the geometries of the parcel and feature-independent quality had to be retrieved and then the updating process had to be done. Furthermore, the updating had to be performed for redundant geometries as well. Considering the limitations of the linear referencing and the integrated approach of the current feature hybrid models, the thesis recommends the investigation of other implementations of the feature-hybrid model. For example, raster structures might be used to efficiently store spatially varying quality information associated with a specific feature.

**SDQ elements with the RSVQ model:** RSVQ is capable of supporting a wide range of data quality information. 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. While the outcome of a logical consistency test could be stored using any of the models above (e.g., associate the results of a logical 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.

**Interactive user queries for the feature-independent model:** Although the feature-independent model queries are optimised, the query execution time of minutes still makes the optimised queries unsuitable for interactive user queries. Consumption of more time to execute the queries is because the feature-independent model uses spatial joins (the join performed between two geometries) for querying. One way to achieve faster query execution time of query involving geometries is to choose efficient indexes. The R-Tree index was used for geometries in this research. It is recommended to further investigate better application of spatial indexes on geometries to improve the query execution time.

**Limitations of handling three tables at a time (SQL limitation):** SQL supports only 200 characters at a time in the query string, which in turn limits the SQL extension to query only three tables at one instance. This is a technical limitation of the specific Oracle environment used for this research. However, another programming environment like PERL, could be used to avoid this limitation.
**Enhance query interface:** As mentioned earlier in the research outcomes, developing a user interface was not the main objectives of the research. The current query interface was developed only to support the testing of queries. More enhancements, like syntax checking, for the WITHQUALITY operator and displaying the output of the query within the query interface can be made.

**Improve visualisation:** The current implementation uses Spatial Console software to graphically display the results of the query, but lacks the ability to assign symbology to the quality values for better visualization. Spatial Console has limitations in assigning different symbols to quality values within a table. It is recommended to investigate how to integrate the output generated from the RSVQ to other visualisation softwares like ArcGIS.

**Integrate existing uncertainty or error propagation tools with RSVQ model:** Integrating uncertainty or error propagation tools with the RSVQ model will facilitate analysing of the query results generated by the model. For example, investigate the possibilities to integrate the RSVQ model with Data Uncertainty Engine (DUE) developed by Heuvelink (2007).
References


REFERENCES


Ramlal, B., Beard, K., 1996. An alternate paradigm for representing soils data and data quality information. In: Third International Conference/Workshop on Integrating GIS and Environmental Modeling, Santa Fe, New Mexico, USA.


# Appendix A

## Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANZLIC</td>
<td>Australia New Zealand Land Information Council</td>
</tr>
<tr>
<td>AOI</td>
<td>Area Of Interest</td>
</tr>
<tr>
<td>CMSV</td>
<td>Continuous Model of Spatial Variation</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DEM</td>
<td>Digital Elevation Model</td>
</tr>
<tr>
<td>DMSV</td>
<td>Discrete Model of Spatial Variation</td>
</tr>
<tr>
<td>DSE</td>
<td>Department of Sustainability and Environment</td>
</tr>
<tr>
<td>EMMS</td>
<td>Error Metadata Management System</td>
</tr>
<tr>
<td>ESRI</td>
<td>Environmental Systems Research Institute</td>
</tr>
<tr>
<td>FGDC</td>
<td>Federal Geographic Data Committee</td>
</tr>
<tr>
<td>FH</td>
<td>Feature Hybrid</td>
</tr>
<tr>
<td>FI</td>
<td>Feature Independent</td>
</tr>
<tr>
<td>GCDB</td>
<td>Geographic Coordinate Data Base</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographical Information System</td>
</tr>
<tr>
<td>GISc</td>
<td>Geographical Information Science</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organisation</td>
</tr>
<tr>
<td>LGA</td>
<td>Local Government Area</td>
</tr>
<tr>
<td>LIDAR</td>
<td>Light Detection and Ranging</td>
</tr>
<tr>
<td>LR</td>
<td>Linear Referencing</td>
</tr>
<tr>
<td>MMSV</td>
<td>Mixed Model of Spatial Variation</td>
</tr>
<tr>
<td>OLAP</td>
<td>Online Analytical Processing</td>
</tr>
<tr>
<td>PF</td>
<td>Per Feature</td>
</tr>
<tr>
<td>QIMM</td>
<td>Quality Information Management Model</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Data Base Management System</td>
</tr>
<tr>
<td>RSVQ</td>
<td>RDBMS for Spatial Variation in Quality</td>
</tr>
<tr>
<td>SDQ</td>
<td>Spatial Data Quality</td>
</tr>
<tr>
<td>SDTS</td>
<td>Spatial Data Transfer Standard</td>
</tr>
<tr>
<td>SQL</td>
<td>Standard Query Language</td>
</tr>
<tr>
<td>STC</td>
<td>Sub Table Check</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Appendix B

PL/SQL code

--NON-OPTIMISED-RSVQ code
DECLARE
l VARCHAR2 (100);
q VARCHAR2 (100);
r varchar2 (100);
l1 VARCHAR2 (100);
q1 VARCHAR2 (100);
r1 varchar2 (100);
l2 VARCHAR2 (100);
q2 VARCHAR2 (100);
r2 varchar2 (100);
pro varchar2(2000);
cod varchar2(2000);
midp varchar2 (1000);
tabp varchar2 (1000);
sq11 varchar2 (4000);
t1 varchar2(1000);
t2 varchar2 (1000);
t3 varchar2 (1000);
qs VARCHAR2 (2000);
str NUMBER;
rs varchar2 (1000);
com1 NUMBER ;
com2 NUMBER ;
wqc NUMBER;
tabt1 varchar2 (100);
tabt2 varchar2 (100);
tabt3 varchar2 (100);
BEGIN
-- Function mid returns the mid part of the string,
-- ie string between FROM and WHERE clause
qs := :layer.SQ;
:layer.TIME1 := CT('CURRENT_TIMESTAMP');
midp := mid(qs);
midp := trim(midp);
DBMS_OUTPUT.PUT_LINE(midp);
-- From the string obtained the position of ',' is checked
com1:= INSTR(midp,',');
DBMS_OUTPUT.PUT_LINE('COM1 POSITION:'||' '||com1);
IF com1 <> 0 THEN -- one
t1 := SUBSTR (midp,1,com1-1);
t1 := trim (t1);
DBMS_OUTPUT.PUT_LINE('t1:'||' '||t1);
wqc := INSTR (t1,'WITHQUALITY');
DBMS_OUTPUT.PUT_LINE('WC:'||wqc);
IF wqc <> 0 THEN
    DBMS_OUTPUT.PUT_LINE('PROCESS t1');
    DBMS_OUTPUT.PUT_LINE('EXECUTING DIAMOND #1');
l1:= laye(t1);
q1:= quae(t1);
r1:= res(t1);
dia(l1,q1,r1);
tabt1 := tab(t1);
tabt1 := trim(tabt1);
DBMS_OUTPUT.PUT_LINE('tabt1 is:'||tabt1);
ELSE
tabt1:= t1;
tabt1 := trim(tabt1);
DBMS_OUTPUT.PUT_LINE('tabt1 is:'||tabt1);
END IF;
rs := SUBSTR (midp,com1+1);
DBMS_OUTPUT.PUT_LINE('REMAINING STRING is:'||rs);
com2:= INSTR(rs,',');
DBMS_OUTPUT.PUT_LINE('COM2 POSITION:'||' '||com2);
IF com2 <> 0 THEN
t2 := SUBSTR (rs,1,com2-1);
t2 := trim (t2);
DBMS_OUTPUT.PUT_LINE('t2 :'||' '||t2);
wqc := INSTR (t2,'WITHQUALITY');
DBMS_OUTPUT.PUT_LINE('WC:'||wqc);
IF wqc <> 0 THEN
    DBMS_OUTPUT.PUT_LINE('PROCESS t2');
    DBMS_OUTPUT.PUT_LINE('EXECUTING DIAMOND #2');
    l2:= laye(t2);
    q2:= quae(t2);
    r2:= res(t2);
    dia(l2,q2,r2);
    tabt2:= tab(t2);
    tabt2:= trim(tabt2);
    DBMS_OUTPUT.PUT_LINE('tabt2 is:'||tabt2);
    rs := SUBSTR (rs,com2+1);
    t3 := rs;
    DBMS_OUTPUT.PUT_LINE('t3:'||t3);
    wqc := INSTR (t3,'WITHQUALITY');
    DBMS_OUTPUT.PUT_LINE('WC:'||wqc);
    IF wqc <> 0 THEN
        DBMS_OUTPUT.PUT_LINE('PROCESS t3');
        DBMS_OUTPUT.PUT_LINE('EXECUTING DIAMOND #3');
        l2:= laye(t3);
        q2:= quae(t3);
        r2:= res(t3);
        dia(l2,q2,r2);
        tabt3:= tab(t3);
        tabt3 := trim(tabt3);
        DBMS_OUTPUT.PUT_LINE('tabt3:'||tabt3);
    ELSE
        DBMS_OUTPUT.PUT_LINE('t3 is a ordinary table');
        tabt3 := t3;
        DBMS_OUTPUT.PUT_LINE('tabt3:'||tabt3);
    END IF;
ELSE
    DBMS_OUTPUT.PUT_LINE('t2 is a ordinary table');
    t2 := SUBSTR (rs,1,com2-1);
    DBMS_OUTPUT.PUT_LINE('t2 :'||' '||t2);
    t2 := trim (t2);
APPENDIX B. PL/SQL CODE

tabt2 := t2;
DBMS_OUTPUT.PUT_LINE('tabt2 is:'||tabt2);
t3 := rs;
t3 := SUBSTR (rs,com2+1);
DBMS_OUTPUT.PUT_LINE('t3:'||t3);
wqc := INSTR (t3,'WITHQUALITY');
DBMS_OUTPUT.PUT_LINE('WC:'||wqc);
IF wqc <> 0 THEN
   DBMS_OUTPUT.PUT_LINE('PROCESS t3');
   DBMS_OUTPUT.PUT_LINE('EXECUTING DIAMOND #4');
   l2:= laye(t3);
   q2:= quae(t3);
   r2:= res(t3);
   dia(l2,q2,r2);
tabt3 := trim(t3);
tabt3 := tab(t3);
   DBMS_OUTPUT.PUT_LINE('tabt3:'||tabt3);
ELSE
   DBMS_OUTPUT.PUT_LINE('t3 is a ordinary table');
tabt3 := t3;
END IF;
END IF;

ELSE

t2 := rs;
t2 := trim (t2);
DBMS_OUTPUT.PUT_LINE('t2 :'||' '||t2);
wqc := INSTR (t2,'WITHQUALITY');
DBMS_OUTPUT.PUT_LINE('WC:'||wqc);
IF wqc <> 0 THEN
   DBMS_OUTPUT.PUT_LINE('EXECUTING DIAMOND #5');
   l2:= laye(t2);
   q2:= quae(t2);
   r2:= res(t2);
   dia(l2,q2,r2);
tabt2 := tab(t2);
   DBMS_OUTPUT.PUT_LINE('tabt2 :'||' '||tabt2);
ELSE
   tabt2 := t2;

140
DBMS_OUTPUT.PUT_LINE('tabt2 :|| '||tabt2);
END IF;
ELSE -- one
    DBMS_OUTPUT.PUT_LINE('NORMAL L Q R ');
    t1 := midp;
    DBMS_OUTPUT.PUT_LINE('t1 :|| '||t1);
    DBMS_OUTPUT.PUT_LINE('EXECUTING DIAMOND # 6');
    l:= laye(t1);
    q:= quae(t1);
    r:= res(t1);
    dia(l,q,r);
    tabt1 := tab(t1);
    DBMS_OUTPUT.PUT_LINE('tabt1 is:'||tabt1);
END IF; -- end of IF no one
pro := PROJ(qs);
DBMS_OUTPUT.PUT_LINE(pro);
cod := COND(qs);
DBMS_OUTPUT.PUT_LINE(cod);
IF NOT tabt1 = ' ' AND NOT tabt2 = ' ' AND NOT tabt3 = ' ' THEN
    tabp := tabt1||','||tabt2||','||tabt3;
ELSIF NOT tabt1 = ' ' AND NOT tabt2 = ' ' THEN
    tabp := tabt1||','||tabt2;
ELSE
    tabp := tabt1;
END IF;
OUTNO(pro,tabp,cod);
: LAYER.TIME2 := CT('CURRENT_TIMESTAMP');
MESSAGE('Result generated successfully');
--Exception handling
EXCEPTION
WHEN OTHERS THEN
    MESSAGE(SQLCODE|| ' - '||SQLERRM);PAUSE;
END;
--OPTIMISED-RSVQ code
DECLARE
r1 varchar2 (100);
r2 varchar2 (100);
r3 varchar2 (100);
pro varchar2(2000);
cod varchar2(2000);
midp varchar2 (1000);
tabp varchar2 (1000);
qs VARCHAR2 (2000);
wqc NUMBER;
tabt1 varchar2 (100);
tabt2 varchar2 (100);
tabt3 varchar2 (100);
jocon varchar2 (2000);
cajr varchar2 (2000);
dot varchar2 (100);
c1 number;
c2 number;
m1 varchar2 (1000);
m2 varchar2 (1000);
m3 varchar2 (1000);
tno number;
mlwqc number;
m2wqc number;
m3wqc number;
ccstr1 varchar2 (2000);
ccstr2 varchar2 (2000);
ccstr3 varchar2 (2000);
WHEREC NUMBER;
RESTAB VARCHAR2 (2000);
TABCHEC NUMBER;
t1 timestamp;
t2 timestamp;
t3 NUMBER;
gbcr VARCHAR2 (2000);
gbc VARCHAR2 (2000);
sgbc NUMBER;
BEGIN
-- Get Input
qs := :layer.SQ;
:LAYER.TIME1 := CT('CURRENT_TIMESTAMP');
-- Group By Clause removal
gbcr := RRGB (qs);
-- Function snjoin returns the join condition if present in the input string
jocon := snjoin(gbcr);
DBMS_OUTPUT.PUT_LINE('jocon is :'||jocon);
-- Function conjo returns the condition part if any and
-- also removes the join condition in the condition part if any
cajr := conjo (gbcr);
DBMS_OUTPUT.PUT_LINE('cajr is :'||cajr);
-- The variable WHEREC gets the length of cajr
WHEREC := LENGTH (cajr);
DBMS_OUTPUT.PUT_LINE('cajr is :'||WHEREC);
-- Function mid returns the mid part of the string,
-- ie string between FROM and WHERE clause
midp := mid(qs);
midp := trim(midp);
DBMS_OUTPUT.PUT_LINE(midp);
-- c1 stores the position of First comma in the midp
c1 := INSTR(midp,',',1,1);
DBMS_OUTPUT.PUT_LINE('COM1 POSITION:'||' '||c1);
-- c2 stores the position of Second comma in the midp
c2 := INSTR(midp,',',1,2);
DBMS_OUTPUT.PUT_LINE('COM2 POSITION:'||' '||c2);
-- Condition to check two commas are present in the midpart
IF c1 <> 0 AND c2 <> 0 THEN
  tno := 3;
  DBMS_OUTPUT.PUT_LINE('No of tables are :'||' '||tno);
m1 := SUBSTR (midp,1,c1-1);
DBMS_OUTPUT.PUT_LINE('m1 is:'||' '||m1);
m1wqc := INSTR (m1,'WITHQUALITY');
IF m1wqc <> 0 THEN
  IF WHEREC <> 0 THEN
    ccstr1 := m1||''||cajr;
    restab := fopt(ccstr1);
    DBMS_OUTPUT.PUT_LINE('ccstr1 for table 1 is:'||' '||ccstr1);
  END IF;
END IF;
END IF;
APPENDIX B. PL/SQL CODE

ELSE
  ccstr1 := m1;
  DBMS_OUTPUT.PUT_LINE('ccstr1 for table 1 is:'||' '||ccstr1);
  restab := fopt(ccstr1);
  DBMS_OUTPUT.PUT_LINE('restab is '||restab);
END IF;
  r1 := res (m1);
  DBMS_OUTPUT.PUT_LINE('r1 is:'||' '||r1);
  tabt1 := r1;
  DBMS_OUTPUT.PUT_LINE('tabt1 has WITHQUALITY :'||' '||tabt1);
ELSE
  tabt1 := m1;
  DBMS_OUTPUT.PUT_LINE('tabt1 is ORDINARY :'||' '||tabt1);
END IF;
  m2 := SUBSTR (midp,c1+1);
  DBMS_OUTPUT.PUT_LINE('m2 is:'||' '||m2);
  c1 := INSTR(m2,',');
  m2 := SUBSTR (m2,1,c1-1);
  DBMS_OUTPUT.PUT_LINE('m2 is:'||' '||m2);
  m2wqc := INSTR (m2,'WITHQUALITY');
  IF m2wqc <> 0 THEN
    IF WHEREC <> 0 THEN
      ccstr2 := m2||' '||cajr;
      DBMS_OUTPUT.PUT_LINE('ccstr2 for table 2 is:'||' '||ccstr2);
      restab := fopt(ccstr2);
      DBMS_OUTPUT.PUT_LINE('restab is '||restab);
    ELSE
      ccstr2 := m2;
      DBMS_OUTPUT.PUT_LINE('ccstr2 for table 2 is:'||' '||ccstr2);
      restab := fopt(ccstr2);
      DBMS_OUTPUT.PUT_LINE('restab is '||restab);
    END IF;
    r2 := res (m2);
    DBMS_OUTPUT.PUT_LINE('r2 is:'||' '||r2);
    tabt2 := r2;
    DBMS_OUTPUT.PUT_LINE('tabt2 has WITHQUALITY :'||' '||tabt2);
ELSE
    tabt2 := m2;
    DBMS_OUTPUT.PUT_LINE('tabt2 is ORDINARY :'||' '||tabt2);
M.G.S.M. ZAFFAR SADIQ

END IF;
m3 := SUBSTR (midp,c2+1);
DBMS_OUTPUT.PUT_LINE('m3 is:'||' '||m3);
m3wqc := INSTR (m3,'WITHQUALITY');
IF m3wqc <> 0 THEN
  IF WHEREC <> 0 THEN
    ccstr3 := m3||' '||cajr;
    DBMS_OUTPUT.PUT_LINE('ccstr3 for table 3 is:'||' '||ccstr3);
    restab := fopt(ccstr3);
    DBMS_OUTPUT.PUT_LINE('restab is '||restab);
  ELSE
    ccstr3 := m3;
    DBMS_OUTPUT.PUT_LINE('ccstr3 for table 3 is:'||' '||ccstr3);
    restab := fopt(ccstr3);
    DBMS_OUTPUT.PUT_LINE('restab is '||restab);
  END IF;
r3 := res (m3);
DBMS_OUTPUT.PUT_LINE('r3 is:'||' '||r3);
tabt3 := r3;
DBMS_OUTPUT.PUT_LINE('tabt3 has WITHQUALITY :'||' '||tabt3);
ELSE
  tabt3 := m3;
  DBMS_OUTPUT.PUT_LINE('tabt3 is ORDINARY :'||' '||tabt3);
END IF;
-- Condition to check one comma is present in the midpart
ELSIF c1 <> 0 THEN
  tno := 2;
  DBMS_OUTPUT.PUT_LINE('No of tables are :'||' '||tno);
m1 := SUBSTR (midp,1,c1-1);
DBMS_OUTPUT.PUT_LINE('m1 is:'||' '||m1);
m1wqc := INSTR (m1,'WITHQUALITY');
IF m1wqc <> 0 THEN
  IF WHEREC <> 0 THEN
    ccstr1 := m1||' '||cajr;
    DBMS_OUTPUT.PUT_LINE('ccstr1 for table 1 is:'||' '||ccstr1);
    restab := fopt(ccstr1);
    DBMS_OUTPUT.PUT_LINE('restab is '||restab);
  ELSE
    ccstr1 := m1;
  END IF;
DBMS_OUTPUT.PUT_LINE('ccstr1 for table 1 is:'||' '||ccstr1);
restab := fopt(ccstr1);
DBMS_OUTPUT.PUT_LINE('restab is '||restab);
END IF;
r1 := res (m1);
DBMS_OUTPUT.PUT_LINE('r1 is:'||' '||r1);
tabt1 := r1;
DBMS_OUTPUT.PUT_LINE('tabt1 has WITHQUALITY :'||' '||tabt1);
ELSE
    tabt1 := m1;
    DBMS_OUTPUT.PUT_LINE('tabt1 is ORDINARY :'||' '||tabt1);
END IF;
m2 := SUBSTR (midp,c1+1);
DBMS_OUTPUT.PUT_LINE('m2 is:'||' '||m2);
m2wqc := INSTR (m2,'WITHQUALITY');
IF m2wqc <> 0 THEN
    IF WHEREC <> 0 THEN
        ccstr2 := m2||' '||cajr;
        DBMS_OUTPUT.PUT_LINE('ccstr2 for table 2 is:'||' '||ccstr2);
        restab := fopt(ccstr2);
        DBMS_OUTPUT.PUT_LINE('restab is '||restab);
    ELSE
        ccstr2 := m2;
        DBMS_OUTPUT.PUT_LINE('ccstr2 for table 2 is:'||' '||ccstr2);
        restab := fopt(ccstr2);
        DBMS_OUTPUT.PUT_LINE('restab is '||restab);
    END IF;
    r2 := res (m2);
    DBMS_OUTPUT.PUT_LINE('r2 is:'||' '||r2);
    tabt2 := r2;
    DBMS_OUTPUT.PUT_LINE('tabt2 has WITHQUALITY :'||' '||tabt2);
    ELSE
        tabt2 := m2;
        DBMS_OUTPUT.PUT_LINE('tabt2 is ORDINARY :'||' '||tabt2);
    END IF;
    -- NO comma is present in the midp
ELSE
    tno := 1;
    DBMS_OUTPUT.PUT_LINE('No of table is :'||' '||tno);
m1 := midp;
m1wqc := INSTR (m1,'WITHQUALITY');
IF m1wqc <> 0 THEN
IF WHEREC <> 0 THEN
DBMS_OUTPUT.PUT_LINE('cajr is : '||cajr);
ccstr1 := m1'||' '||cajr;
DBMS_OUTPUT.PUT_LINE('ccstr1 for table 1 is:'||' '||ccstr1);
restab := fopt(ccstr1);
DBMS_OUTPUT.PUT_LINE('restab is '||restab);
ELSE
ccstr1 := m1;
DBMS_OUTPUT.PUT_LINE('ccstr1 for table 1 is:'||' '||ccstr1);
restab := fopt(ccstr1);
DBMS_OUTPUT.PUT_LINE('restab is '||restab);
END IF;
r1 := res (m1);
DBMS_OUTPUT.PUT_LINE('r1 is:'||' '||r1);
tabt1 := r1;
DBMS_OUTPUT.PUT_LINE('tabt1 has WITHQUALITY :'||' '||tabt1);
ELSE
tabt1 := m1;
DBMS_OUTPUT.PUT_LINE('tabt1 is ORDINARY :'||' '||tabt1);
END IF;
END IF;
-- Function PROJ gets the projection part (SELECT .. FROM) from qs
pro := PROJ(qs);
-- Function GBCI includes the GROUP BY CLAUSE if present in the SQ
gbc := gbci(qs);
DBMS_OUTPUT.PUT_LINE('GBC is: '||gbc);
sgbc := LENGTH (gbc);
IF sgbc = 0 THEN
gbc := ' ';
END IF;
--tabp concatenation based on 3 or 2 or 1 table
IF NOT tabt1 = ' ' AND NOT tabt2 = ' ' AND NOT tabt3 = ' ' THEN
tabp := tabt1||','||tabt2||','||tabt3;
tabp := trim (tabp);
ELSIF NOT tabt1 = ' ' AND NOT tabt2 = ' ' THEN
tabp := tabt1||','||tabt2;
APPENDIX B. PL/SQL CODE

tabp := trim (tabp);
ELSE
  tabp := tabt1;
  tabp := trim (tabp);
END IF;

-- Final processing of sql tp produce the output table temp
-- Final processing of sql tp produce the output table temp
IF JOCON <> '' THEN
  OUTWJ(pro,tabp,jocon,gbc);
  :LAYER.TIME2 := CT('CURRENT_TIMESTAMP');
  MESSAGE('Result generated successfully');
ELSE
  OUTWOJ(pro,tabp,gbc);
  :LAYER.TIME2 := CT('CURRENT_TIMESTAMP');
  MESSAGE('Result generated successfully');
END IF;

-- Exception handling
EXCEPTION
  WHEN OTHERS THEN
    MESSAGE(SQLCODE||' - '||SQLERRM);PAUSE;
END;