An accuracy model for relational databases

Jing Ding
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Recommended Citation
Ding, Jing, "An accuracy model for relational databases" (2004). Retrospective Theses and Dissertations. 20400.
https://lib.dr.iastate.edu/rtd/20400

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
An accuracy model for relational databases

by

Jing Ding

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Leslie Miller, Major Professor
Shashi K. Gadia
Sarah M. Nusser

Iowa State University
Ames, Iowa
2004

Copyright © Jing Ding, 2004. All rights reserved.
This is to certify that the master’s thesis of

Jing Ding

has met the thesis requirements of Iowa State University.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>vii</td>
</tr>
<tr>
<td>CHAPTER 1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2. PREVIOUS RESEARCH</td>
<td>4</td>
</tr>
<tr>
<td>CHAPTER 3. ACCURACY RELATIONAL DATABASE MODEL</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Basic Issues and the Taxonomy of Accuracy</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Acquiring Accuracy Values</td>
<td>9</td>
</tr>
<tr>
<td>3.2.1 Attribute Accuracy</td>
<td>10</td>
</tr>
<tr>
<td>3.2.2 Acquiring Accuracy Values</td>
<td>11</td>
</tr>
<tr>
<td>CHAPTER 4. RELATIONAL ALGEBRA OPERATIONS AND ACCURACY</td>
<td>15</td>
</tr>
<tr>
<td>4.1 Selection</td>
<td>15</td>
</tr>
<tr>
<td>4.2 Projection</td>
<td>16</td>
</tr>
<tr>
<td>4.3 Natural Join</td>
<td>17</td>
</tr>
<tr>
<td>4.4 Union</td>
<td>19</td>
</tr>
<tr>
<td>4.5 Difference</td>
<td>21</td>
</tr>
<tr>
<td>4.6 Intersection</td>
<td>21</td>
</tr>
<tr>
<td>CHAPTER 5. ACCURACY RELATIONAL ALGEBRA</td>
<td>23</td>
</tr>
<tr>
<td>5.1 Definitions of the Operations</td>
<td>23</td>
</tr>
<tr>
<td>5.2 Properties of the Accuracy Relational Model</td>
<td>24</td>
</tr>
</tbody>
</table>
CHAPTER 6. ACCURACY RELATIONAL QUERY LANGUAGE .......... 26

CHAPTER 7. PROTOTYPE ...................................... 29

  7.1 Query Evaluation .................................. 29

  7.2 An Example ...................................... 32

CHAPTER 8. CONCLUSION .................................... 35

BIBLIOGRAPHY ............................................. 36

ACKNOWLEDGEMENTS ...................................... 37
LIST OF TABLES

Table 3.1  An example of a relation with tuple-level accuracy values . . . . . . 8
Table 3.2  An example of a relation with attribute level accuracy values . . . . . . 9
Table 3.3  Value dependent accuracy and value independent accuracy . . . . . . 10

Table 4.1  An example of the employee table . . . . . . . . . . . . . . . . . . . . 16
Table 4.2  Query result after selection . . . . . . . . . . . . . . . . . . . . . . . 16
Table 4.3  The result after a projection on empNo, fName and Salary . . . . . . 16
Table 4.4  The result relation after a projection on lName and Salary . . . . . . 17
Table 4.5  Relation Dept . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
Table 4.6  Relation formed by joining Emp and Dept . . . . . . . . . . . . . . . 19
Table 4.7  Relation emp1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
Table 4.8  Relation emp2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
Table 4.9  Relation emp1 $\cup$ emp2 . . . . . . . . . . . . . . . . . . . . . . . . 20
Table 4.10 Relation emp $'$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
Table 4.11 Relation emp - emp $'$ . . . . . . . . . . . . . . . . . . . . . . . . . 21
Table 4.12 Relation emp $\cap$ emp $'$ . . . . . . . . . . . . . . . . . . . . . . . . 22
LIST OF FIGURES

Figure 7.1 The function of the query preprocessor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
ABSTRACT

The relational database model has gained a great deal of attention over the past decades for its advantages. However, it suffers from a ubiquitous problem of data imperfection. Based on this situation, several models have been proposed to solve those problems. These models could improve the situation caused by the imperfect data, but they lack the ability to handle the accuracy issue that exists in the data. Therefore, an accuracy model is proposed here for relational databases. The theory of probability is utilized to help define accuracy in databases, and also relational algebra is extended for the use with accuracy model. Some implementation issues are also discussed.
CHAPTER 1. INTRODUCTION

The relational database model has gained a great deal of attention over the past decades for its advantages over other data models for manipulating data. However, it suffers from lack of mechanisms to deal with data imperfection. Many real world data are not perfect so they can not be comprehensively represented in a relational database. For example, in an employee database, the annual salary of a specific employee may not be a precise value but a “high” or “low” instead. Or, if there is an attribute named “the mothers maiden name” in the employee table, then this attribute for a male employee may probably have no corresponding value. The same problem also exits for GIS databases, in which data quality is an important issue but many data are inherently inaccurate. For example, a point/line/area may not be in the right position as specified in the database, or the land cover type of a specific area may not be correctly classified.

Since the relational model can not represent the inherent imperfection of the data items, it just ignores the problem and focuses primarily on the data that are stored-assuming their correctness. Therefore, the relational model is not capable of manipulating the inherently imperfect data and providing convincing query results. Before we proceed on, it is necessary to make a classification of the imperfect data.

Incomplete Data. Sometimes it is not practical to obtain complete information in real world because it may even not exist. If we only have a partial knowledge of a real life situation, then the information we have in hand is incomplete. In the context of relational databases, incomplete information may be represented as missing attribute values. Among the various schemes to handle the incomplete information in the databases, the most common approach is to use null values. A null value is a place holder for an attribute of a relation whose value
is either unknown or inapplicable.

Although using null values is considered a reasonable way of handling incomplete information, there are several interpretations of null values. The most well known ones are the “unknown” interpretation, and the “not applicable” interpretation. When the value is assumed to exist but is not known, such as the salary of an employee, it results in the incompleteness in databases and is represented as a null value for this unknown property. When the value does not exist at all, such as the maiden name of a male, it can also be represented as a null value.

*Imprecise Data.* More frequently, the value of an attribute is available but is not precise. In fact, information stored in a database is often subject to imprecision. Instead of having an exact value, an attribute may have an imprecise value represented as a vague (or fuzzy) value, a range of possible values of the attribute, or a set of discrete values associated with the attribute domain.

Although the attribute value is not precise, it is many times still useful. For example, the attribute value for an employee’s salary as “between $50,000 and $60,000” is a range value which is clearly imprecise, but if we want to get a list of employees whose salaries are below $65,000, then this employee must be considered for the list.

*Uncertainty.* Both incomplete and imprecise data can lead to uncertainty. For instance, a query involving the incomplete or imprecise information will return answers that are uncertain. Dey and Sarkar[4] classified uncertainty into two categories: uncertainty due to vagueness and uncertainty due to ambiguity. Uncertainty due to vagueness is associated with the difficulty of making sharp or precise distinction in the real world and is often modeled with fuzzy set theory. Uncertainty due to ambiguity is associated with situations in which the choices among several precise alternatives are left unspecified, and is often handled with probabilistic relational models.

This thesis primarily focuses on two issues: how to represent the data uncertainty in the relational database model and how to handle it. Our approach is to develop a model to deal with an uncertainty measurement called “accuracy”. This model is closely related to relational database systems that are conventionally used in practice. The term “accuracy” will be defined
in a later chapter, and it will be used to represent the data uncertainty. For example, when we say that “we are 99% confident that this data is correct”, this 99% is one form of accuracy. More examples will be provided later in Chapter 3.

The rest part of the thesis is organized as follows. Previous research handling the imperfect information mentioned above is discussed in Chapter 2. Chapter 3 discusses basic terms and ideas of the proposed accuracy model. The accuracy relational algebra is discussed in Chapter 4 and Chapter 5. The query language for the accuracy-enabled relational model is defined in Chapter 6. Chapter 7 discusses the implementation issues and system design.
CHAPTER 2. PREVIOUS RESEARCH

To handle the imperfect data that exist in the relational databases, a lot of research has been done.

Codd[3] presents a treatment of incomplete information in relational databases by providing the semantics of null values. A null value is stored to represent the missing value or non-applicable value, with the constraint that a null value may not appear as a component of a primary key. Three-valued logic is used for query evaluation, with truth values true, false, and unknown (maybe). Queries evaluate to unknown when attribute values involved in query decisions are unknown. This approach assumes that data values are either known with certainty or are unknown, which is too restrictive to model the accuracy related uncertainty problem.

Lipski[6] extends Codd's work by discussing the semantic issue of null values. He provides two different interpretations of queries, the internal one and the external one. A query with an internal interpretation is under the closed world assumption and refers only to the information known to the system. On the other hand, a query with an external interpretation is under the open world assumption and imprecise information is introduced. Although Lipski's scheme is more expressive than the usual interpretation of null values, the model does not provide any measure of uncertainty or accuracy.

Raju and Majumdar[5] discuss a fuzzy relational database model to deal with the imprecise information. The authors apply fuzzy set theory to relational databases to define the term fuzzy relation. They classify the fuzzy relations into two categories. A type-1 fuzzy relation may be considered as a first-level extension of a classical relation, where the domain of an attribute is a fuzzy set. Each element is a pair consisting of data value and the membership value. The membership value corresponds to the association among the attribute data values.
Although type-1 fuzzy relations enable us to represent imprecision in the association among data values, its role in capturing uncertainty in data values is rather limited. The type-2 fuzzy relations provide further generalization by allowing the attribute domain to be a set of fuzzy sets. Type-2 fuzzy relations also have limitations. Although the type-2 fuzzy relational model deals with data uncertainty, since it allows the domain of an attribute to be a set of fuzzy sets, it violates 1-NF which may pose implementation problems.

Wong[7] proposed a statistical model to handle data incompleteness in relational databases. This approach exploits the prior information and transforms the query posed by the user with a statistical preprocessor. This model assumes that data items are not completely available in the database, because the cost of storing them altogether is very high. For example, it is hard to record the position of a ship at every moment. By applying some statistical knowledge and exploiting prior information, the model handles the incompleteness problem with some predictions on the data item. However, this model ignores the uncertainty of the data items themselves. Instead, it just provides a framework to handle data incompleteness. If the values of the data have an obvious pattern, statistical method and previous information together may be useful to predict values for the incomplete data at some acceptable level. But for inherent stochastic data, such as stock prices, this model seems to be insufficient to represent uncertainty.

As long as probability is regarded as a reasonable measurement of uncertainty, there is a great deal of research on probabilistic relational models. Cavallo and Pittarelli[2] provide a general model of probabilistic databases. What they did is attach a probability measure to every tuple of a relation, indicating the joint probability of all the attribute values in that tuple being correctly assigned. They extend relational algebra operations, such as projections and joins, with the probability measures, and discuss the use of information theoretic measures, particularly with regard to the Shannon entropy of the information. The model requires that the total probability assigned to all the tuples in a relation is exactly one. This requirement is overly restrictive, since for every object that is known to exist with certainty, the probability measure associated with it should be unity, thus a separate relation is required for it.
Barbara et al.[1] also proposed an extension of the relational model using probability theory. Their model specifies a probability distribution for the values of a given attribute, and redefines the relational operations using semantics of probability theory. However, since there may be multiple values, each with a probability measure, of the same attribute in one tuple, this model again introduces the non-1NF implementation issue.

Dey and Sarkar[4] discuss a probabilistic relational model in which each tuple is assigned a probability measure, with a restriction that the probability measures associated with any given key value must add up to no more than one. This model overcomes the problem of Cavello and Pittarellis model, but it shares the same assumption with the other probability models previously mentioned. They all assume that there exists an oracle that could tell the probability measure of each tuple, or attribute, which raises concerns when these approaches are implemented.

These approaches discussed above provide insight into many of the problems of dealing with imperfect data, but do not directly allow database system designers and users with ways to deal with accuracy issues. In the next chapter we look at the question of data accuracy and examine ways to extend the relational database model to support accuracy.
3.1 Basic Issues and the Taxonomy of Accuracy

The term “accuracy” comes from the physics context. All experimental data have associated uncertainties. For example, if one records a height of a person as 5’7”, typically it doesn’t mean the person is exactly 5’7” tall. Instead, the measurer may round the result to the nearest inch. In physics, accuracy is a measure of how close the measured value is to the true or accepted value. For example, if you used a balance to find the mass of a known standard 100.00g mass, and you got a reading of 85.50g, then your measurement would not be very accurate.

Accuracy is different from precision. Precision refers to how close together a group of measurements actually are to each other. Precision has nothing to do with the true or accepted value of a measurement, so it is quite possible to be very precise and totally inaccurate. One important distinction between accuracy and precision is that accuracy can be determined by only one measurement, while precision can only be determined with multiple measurements. In many cases, when precision is high and accuracy is low, the fault can lie with the instrument. If a balance or a thermometer is not working correctly, they might consistently give inaccurate answers, resulting in high precision and low accuracy.

In the database context, accuracy is defined as the degree to which information in a database matches the true or accepted values. Accuracy is an issue pertaining to the quality of data and the number of errors contained in a dataset. The type of accuracy issue that a data item can suffer from is dependent on the nature of the data. For example, for nominal or ordinal data the accuracy issues are typically issues of assignment. In other words they are the values of a nominal or ordinal attribute correctly assigned to the tuples.

Ratio data are more likely to have errors that can be measured by defining a confidence
interval around the values. For example, a ratio value that is stored in a traditional database as 300 may in fact be more accurately stated as $300 \pm 10$.

Such accuracy issues can be considered as several different levels of data granularity, i.e., the definition is level dependent. Therefore, our accuracy model supports several levels of accuracy, namely, the relation level accuracy, the tuple level accuracy, and the attribute level accuracy.

**Relation level accuracy** is defined based on the data in a whole relation. It may be a function or a constant that defines tuple level accuracy for all of the tuples in that relation. This is different from tuple level accuracy whose definition will be introduced immediate below, since relation level accuracy is defined uniformly for all tuples in a relation, whereas each tuple may have an individual tuple level accuracy.

**Tuple level accuracy** is defined as the probability that the attribute values are correctly assigned to the key value of the tuple. For example, given an employee table with the scheme $\text{emp}(\text{empNo}, \text{Name}, \text{Name}, \text{Salary})$, then the accuracy value is the conditional probability that $\text{Name}$, $\text{Name}$ and $\text{Salary}$ are correctly assigned to the key $\text{empNo}$, i.e., the probability $P(\text{Name} = \text{ln}, \text{Name} = \text{fn}, \text{Salary} = \text{s}|\text{empNo} = e)$. Our model assumes that the keys are correct, i.e., $P(\text{empNo} = e) = 1$. Note that in our model, we use the traditional definition of a key (i.e., only one tuple is associated with a key). Therefore, there will not be multiple tuples denoting different possibilities of assignments of the non-key attributes to the same key. Table 3.1 shows an example of tuple level accuracy in the relation $\text{emp}$ of our running example.

Table 3.1 An example of a relation with tuple-level accuracy values

<table>
<thead>
<tr>
<th>empNo</th>
<th>Name</th>
<th>Name</th>
<th>deptNo</th>
<th>Salary</th>
<th>Tuple accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>9527</td>
<td>Chow</td>
<td>Stephen</td>
<td>001</td>
<td>55,000</td>
<td>0.6</td>
</tr>
<tr>
<td>9530</td>
<td>Pitts</td>
<td>Joey</td>
<td>001</td>
<td>60,000</td>
<td>0.7</td>
</tr>
</tbody>
</table>

**Attribute level accuracy** is defined based on the category of the attribute value. Attribute values can be classified into three categories: nominal values, ratio values and ordinal values. Nominal attributes serve only to identify or distinguish one entity from another, such as department names, hobbies, skills and vehicle plate numbers. Ratio attributes identify the
numeric values, such as the temperatures. Ordinal data are categorical data where there is a logical ordering to the categories. A typical example is the set of options we see in many surveys: strongly agree, agree, neutral, disagree, and strongly disagree. For a nominal or ordinal attribute, the major error is misclassification, and accuracy is an issue of understanding the misclassification. We define the accuracy of a nominal or ordinal attribute as the probability that this attribute is correctly assigned to the key in the same tuple. Table 3.2 shows us an example of accuracy for nominal or ordinal attributes. The \textit{deptNo} attribute value of the employee whose \textit{empNo} is 9527 is “001” with an accuracy value of 0.9, denoting the probability that the value of employee 9527’s department number is correctly assigned.

For a ratio attribute, accuracy is not represented as a probability value, but a $\pm$ error value instead. For example, if the attribute value for \textit{temperature} shown on a thermometer is $60 \pm 0.006^\circ F$, then the accuracy value is the error value ($\pm 0.006^\circ F$). Table 3.2 also gives an example of ratio accuracy. The annual salary of employee 9527 is $55,000, with the accuracy of $\pm 10$ and a probability 0.95 associated with the confidence of the error range.

<table>
<thead>
<tr>
<th>empNo</th>
<th>lName</th>
<th>fName</th>
<th>deptNo</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>9527</td>
<td>Chow</td>
<td>Stephen</td>
<td>001</td>
<td>55,000</td>
</tr>
<tr>
<td>9530</td>
<td>Pitts</td>
<td>Joey</td>
<td>002</td>
<td>60,000</td>
</tr>
</tbody>
</table>

3.2 Acquiring Accuracy Values

To be useful it is necessary to be able to supply reasonable accuracy values. While generating accuracy values with an oracle is very appropriate when looking at the theoretical development of a concept, they have to be replaced with machine/human generated accuracy values to be useful in a database management system.

In the next subsection, approaches of automatically acquiring accuracy based on data type and accuracy level are examined.
3.2.1 Attribute Accuracy

3.2.1.1 Accuracy for Nominal or Ordinal Attributes

For a nominal or ordinal attribute, the attribute level accuracy is defined as the probability that the attribute value is correctly assigned or classified given the key value of the same tuple. The accuracy value could be either value dependent or value independent. To say the accuracy is value dependent means that the value does have something to do with the accuracy, i.e., for the same attribute, different attribute values correspond to different accuracy values. On the other hand, value independent accuracy always remains the same while the attribute value changes, thus for every entry in the same column of a relation, there is a uniform accuracy value.

Note that these two ways of defining accuracy bring the issue of level back into the discussion. Value dependent accuracy must be identified as part of the value within the tuple, whereas value independent accuracy is part of the metadata describing the database attributes.

Table 3.3 Value dependent accuracy and value independent accuracy

<table>
<thead>
<tr>
<th>empNo</th>
<th>lName</th>
<th>fName</th>
<th>deptNo</th>
<th>Salary (±10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9527</td>
<td>Chow</td>
<td>Stephen</td>
<td>001(0.9)</td>
<td>55,000</td>
</tr>
<tr>
<td>9530</td>
<td>Pitts</td>
<td>Joey</td>
<td>002(0.8)</td>
<td>60,000</td>
</tr>
</tbody>
</table>

Table 3.3 provides an example of both levels. The metadata describing fName includes the probability of 0.95 indicating that assignment of first names has been done with 0.95 accuracy. The deptNo assignment, on the other hand, is department dependent and must be handled at the attribute value level within a tuple.

The interpretation of Table 3.3 is that first names are assigned correctly with probability 0.95 and that the errors of assignment for deptNo are more related to the individual department that the employee works in. It should be noted here that by looking at accuracy in the context of a single attribute (at either level), it is being assumed that the attributes are independent with respect to accuracy, i.e., the accuracy of one attribute does not influence the accuracy of another. As noted earlier, we assume that accuracy that involves the interaction of several
attributes is handled by tuple accuracy (described later in this chapter).

3.2.1.2 Accuracy for Ratio Attributes

For ordinal or ratio attributes, the accuracy issues are the errors that can be specified as a confidence interval around the values. Table 3.3 also gives us an example of accuracy for ordinal/ratio attributes. The attribute *Salary* has an accuracy value in the form of ±10 error. Also, there is a probability 0.95 associated with the range error denoting the level of confidence we have about the error range.

The accuracy of ordinal/ratio attributes can also be value-dependent or value-independent. The accuracy of attribute *Salary* in Table 3.3 is an example of value-independent accuracy.

3.2.2 Acquiring Accuracy Values

Given those definitions about accuracy values, the question is how to estimate the accuracy values.

3.2.2.1 Nominal/Ordinal Accuracy

For nominal or ordinal attributes, our approach to estimate the accuracy values is based on sampling either the current snapshot or the snapshot plus archival data.

To acquire accuracy that is dependent on the value of a nominal attribute, we can randomly select \( n_1 \) tuples with attribute value \( v_1 \) and count how many are correctly classified for the attribute say \( k_1 \), then for attribute value \( v_1 \) the percentage that it is correctly classified is \( k_1/n_1 \), and we take this percentage as an approximation of the accuracy value for this attribute. Note that in this case each value of attribute has its associated accuracy information.

To get accuracy estimates that are independent of the value of a nominal attribute, we can randomly sample \( n \) values of the attribute and count the number of tuples whose values for this attribute are correctly assigned, say \( k \), then the accuracy estimate is the percentage that they are correctly classified, i.e., \( k/n \), and we assign this percentage value as an estimation of
the accuracy value for this attribute. In this case, all the values for the attribute share the same accuracy information.

3.2.2.2 Ratio Accuracy

Value-dependent accuracy for ratio attributes can be provided by the device which yields the measurement error. The device always gives an error that is dependent on value or it gives a function for generating error. For example, the manufacturer of MiniTemp™FoodSafety (MTFS) infrared(IR) Thermometer[9] announces the accuracy of this product as:

\[
f(t) = \begin{cases} 
\pm 2^\circ F (\pm 1^\circ C), & 32^\circ F \leq t \leq 150^\circ F (0^\circ C \leq t \leq 65^\circ C) \\
\pm 2^\circ F (\pm 1^\circ C), & -25^\circ F \leq t \leq 32^\circ F (-30^\circ C \leq t \leq 0^\circ C) \\
1.5\%t, & 150^\circ F \leq t \leq 400^\circ F (65^\circ C \leq t \leq 200^\circ C)
\end{cases}
\]

with a probability associated with the confidence of the errors as 0.97.

Value-independent accuracy for ratio attributes can be provided by the device or an algorithm which gives a constant error or we test the device and can not find an error dependent on value. For example, an LED(Light Emitting Diodes) digital thermometer[8] has an operating range between 0°C and 150°C, and the manufacturer announces that this thermometer has ±0.4°C accuracy.

Note that while these results imply that the accuracy values for ordinal or ratio values are in some sense absolute, there is always a probability associated with such accuracy estimates. For example, we are confident that any temperature produced by a thermometer falls in the accuracy range with probability \(p\). In a similar fashion, a confidence interval for accuracy is always given with some probability \(p\) indicating the level of confidence that we have that the actual value falls in the interval bounds. In the MTFS IR thermometer example shown above, the confidence level is 0.97, which is also provided by the manufacture of the product.

3.2.2.3 Tuple Accuracy

There are difficulties in getting the values of tuple accuracy. However, we can give an estimation of accuracy values by deriving them from attribute values.
According to our previous discussion, the tuple level accuracy value indicates the probability that the nonprime attribute values are correctly assigned to the key. Therefore, given a relation scheme \( R(K, A_1, \ldots, A_n) \) in which \( K \) is the primary key, and \( A_1, \ldots, A_n \) are non-key attributes, the accuracy value should be
\[
P(A_1 = a_1, \ldots, A_n = a_n|K = k) \text{ where } a_1 \ldots a_n \text{ are the measured values.}
\]

**Case 1.** We ignore the ratio attributes but consider only nominal/ordinal attributes.

1. If each non-key attribute has an accuracy value \( P_i \), \( P_i = P(A_i = a_i|K = k) \), estimated from random sampling, and the attributes are independent of each other, then the tuple accuracy is
\[
P(A_1 = a_1, \ldots, A_n = a_n|K = k) = P_1 \cdot P_2 \cdots P_n
\]

2. If the attributes are not independent of each other, but we know the attribute pairs in which one attribute is dependent on the other one. For example, in a pair \((A_i, A_j)\), \( A_j \) depends on \( A_i \), then we could consider estimating the conditional probability \( P_{ji} = P(A_j = a_j|A_i = a_i, K = k) \) using sampling as well. We can randomly select \( n \) tuples and count how many are correctly classified for attribute \( A_j \) assuming that the attribute value of \( A_i \) is correctly classified, say \( k \), then we take the percentage \( k/n \) as an approximation of the conditional probability \( P(A_j = a_j|A_i = a_i, K = k) \). Then,
\[
P(A_i = a_i, A_j = a_j|K = k) = P(A_i = a_i|K = k) \cdot P(A_j = a_j|A_i = a_i, K = k)
\]

And we can get the tuple accuracy based on this. For example, we have a relation scheme \( R(K, A_1, A_2, A_3, A_4) \). \( A_3 \) is dependent on \( A_2 \), and all the other non-key attributes are independent of each other. Then the tuple accuracy is estimated by \( P(A_1 = a_1, A_2 = a_2, A_3 = a_3, A_4 = a_4|K = k) = P_1 \cdot P_2 \cdot P_{3|2} \cdot P_4 \).

**Case 2.** We don’t ignore the ratio attributes regarding their influence on the tuple level accuracy value.

In case 1, we can estimate tuple accuracy because by ignoring ratio attributes, we estimate each accuracy value for an individual nominal attribute as a “probability” or percentage value,
and we can multiply the values since they look like the same type of values. If we take the ordinal/ratio attributes into consideration, the accuracy value of an ordinal/ratio attribute is in the form of $\pm \varepsilon_i$. However, we still have a "probability" value associated with the error range, denoting the probability with which the attribute value falls in an acceptable error range. Therefore, we choose to use this probability value as the probability accuracy value for that attribute. Then, we can apply the multiplication formula to estimate the tuple accuracy.
CHAPTER 4. RELATIONAL ALGEBRA OPERATIONS AND ACCURACY

In order for an accuracy model to be useful in the database setting, it must extend to the query results whenever possible. In the remainder of this chapter we look at the primary relational operations in the context of how they deal with accuracy values embedded in the database.

4.1 Selection

The selection operation is used to find the tuples that satisfy the selection condition. Two aspects of our accuracy model are of consideration in selection, namely, assignment accuracy (e.g., nominal attributes and tuple accuracy) and range accuracy (e.g., ordinal and ratio attributes).

For assignment error, it is simply a matter of reporting the error in the result. For nominal attributes in the selection criteria, this assignment accuracy gives the user an idea of the likelihood that the correct tuples have been retrieved. There is no change in the tuple assignment accuracy since the entire tuple is retrieved as long as it satisfies the selection condition.

A slightly different issue exists for selection condition attributes that are either of type ordinal or ratio. In this case it is clear what alternative values it makes sense to retrieve. For example, in Table 4.1, any query with the selection condition “where Salary>55000” results in the interpretation “where Salary>54990” being used in the query that is executed against the database. Table 4.2 shows the resulting relation for this condition.
Table 4.1 An example of the employee table

<table>
<thead>
<tr>
<th>Employee</th>
<th>empNo</th>
<th>lName</th>
<th>fName</th>
<th>deptNo</th>
<th>Salary (±10)(0.99)</th>
<th>Tuple accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9527</td>
<td>Chow</td>
<td>Stephen</td>
<td>001 (0.8)</td>
<td>54,995</td>
<td>0.554</td>
</tr>
<tr>
<td></td>
<td>9528</td>
<td>Smith</td>
<td>John</td>
<td>002 (1.0)</td>
<td>40,000</td>
<td>0.693</td>
</tr>
<tr>
<td></td>
<td>9529</td>
<td>Bond</td>
<td>James</td>
<td>001 (0.8)</td>
<td>60,000</td>
<td>0.554</td>
</tr>
<tr>
<td></td>
<td>9530</td>
<td>White</td>
<td>Lucy</td>
<td>001 (0.8)</td>
<td>55,000</td>
<td>0.554</td>
</tr>
</tbody>
</table>

Table 4.2 Query result after selection

<table>
<thead>
<tr>
<th>empNo</th>
<th>lName</th>
<th>fName</th>
<th>deptNo</th>
<th>Salary (±10)(0.99)</th>
<th>Tuple accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>9527</td>
<td>Chow</td>
<td>Stephen</td>
<td>001 (0.8)</td>
<td>54,995</td>
<td>0.554</td>
</tr>
<tr>
<td>9529</td>
<td>Bond</td>
<td>James</td>
<td>001 (0.8)</td>
<td>60,000</td>
<td>0.554</td>
</tr>
<tr>
<td>9530</td>
<td>White</td>
<td>Lucy</td>
<td>001 (0.8)</td>
<td>55,000</td>
<td>0.554</td>
</tr>
</tbody>
</table>

4.2 Projection

The projection operation retrieves a subset of the columns from all of the tuples and then removes any duplicates that have been created.

Since the accuracy model revolves around the tuple key for assignment accuracy, tuple accuracy in the result of the project is set to \( \bot \) (undefined) if any of the attributes in the primary key are dropped. Similarly, assignment accuracy values for nominal attributes are considered to be undefined. Notice that by dropping one or more attributes involved in the key, there is no longer a key that we have assignment accuracy information for.

If the key is retained under the projection, we can estimate the new tuple accuracy by calculating the value of \( P(B_1, \ldots, B_m | \text{key}) \) where \( \{B_1, \ldots, B_m\} \cup \{\text{key}\} \) make up the set of attributes that have been projected over.

Table 4.3 The result after a projection on \( \text{empNo}, \text{fName} \) and \( \text{Salary} \)

<table>
<thead>
<tr>
<th>empNo</th>
<th>fName (0.7)</th>
<th>Salary (±10)(0.99)</th>
<th>Tuple accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>9527</td>
<td>Stephen</td>
<td>54,995</td>
<td>0.693</td>
</tr>
<tr>
<td>9528</td>
<td>John</td>
<td>40,000</td>
<td>0.693</td>
</tr>
<tr>
<td>9529</td>
<td>James</td>
<td>60,000</td>
<td>0.693</td>
</tr>
<tr>
<td>9530</td>
<td>Lucy</td>
<td>55,000</td>
<td>0.693</td>
</tr>
</tbody>
</table>

Table 4.3 gives an example of a result relation after projecting on attributes \( \text{empNo}, \text{fName} \) and \( \text{Salary} \).
Since **fName** and **Salary** are independent attributes, the tuple accuracy is calculated as the product of the attribute accuracy values for **fName** and **Salary**, which is $0.7 \times 0.99 = 0.693$.

Notice that in this example, the accuracy value of each tuple in the result relation is higher than that in the original tuple. This makes sense since each attribute contributes more or less to the overall inaccuracy. The more attributes that are involved in a tuple after projection, the lower the accuracy of a tuple will probably be, unless the attributes are all accurate (with attribute accuracy equal to 1.0).

Generally speaking, the key is not necessarily retained under the projection, and accuracy is set to “undefined” in this case for tuple accuracy and nominal attribute accuracy. Table 4.4 shows an example of the result after projecting on attributes **lName** and **Salary**.

<table>
<thead>
<tr>
<th>lName (⊥)</th>
<th>Salary (±10)(0.99)</th>
<th>Tuple accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chow</td>
<td>54,995</td>
<td>⊥</td>
</tr>
<tr>
<td>Smith</td>
<td>40,000</td>
<td>⊥</td>
</tr>
<tr>
<td>Bond</td>
<td>60,000</td>
<td>⊥</td>
</tr>
<tr>
<td>White</td>
<td>55,000</td>
<td>⊥</td>
</tr>
</tbody>
</table>

The attribute **Salary** is of type ratio, thus the accuracy remains the same in the result relation.

### 4.3 Natural Join

A join links the tuples from two (or more) relations based on two groups of attributes (one from each relation, with equal numbers of elements) from a common domain, by comparing the values of these two groups of attributes and returning rows that have matching values. In the accuracy relational database model, the join operation is performed similarly: every tuple in one relation is checked for a match on the attribute common to both relations with every tuple in the other relation, and if a match is found, the two tuples are combined together to form a new row in the result relation. We assume that the common attribute(s) for a join operation is over a foreign key of one participating relation. Two types of accuracy values have
to be determined for join results, namely, attribute accuracy values and tuple accuracy for the resulting tuple. Range accuracy values (i.e., \(\pm e\)) for ordinal and/or ratio attributes are left unchanged.

The issue for nominal attributes is slightly more complex. Nominal attributes that appear in the anchor relation (contains foreign key) will be assigned the same accuracy values that exist in the anchor relation.

For nonprime (non-key) nominal attributes in the second relation, we calculate a new assignment accuracy that is based on their existing assignment accuracy and the assignment accuracy of the foreign key attributes in the anchor relation. For the case where only one join attribute exists, it is simply a matter multiplying the assignment accuracy of the join attribute from the anchor relation times the assignment accuracy of the nominal attributes in the foreign key relation. In the case where there is more than one nominal attribute in the set of join attributes, we estimate the assignment accuracy of the set of join attributes by evaluating \(P(C_1, \ldots , C_i|\text{key})\) where \(C_1 \cdots C_i\) are the join attributes and \(\text{key}\) is the key of the anchor relation.

As an example, suppose we have two relations, \(\text{director}\{\text{name, age, gender, movie_name, movie_year}\}\) and \(\text{movie}\{\text{name, year, producer, length}\}\). In the relation \(\text{movie}\), \(\text{producer}\) is a nominal attribute whereas \(\text{length}\) is a ratio attribute. We want to join these two relations over the attributes \(\text{movie_name}\) and \(\text{movie_year}\), which together are the key of the relation \(\text{movie}\). In the resulting relation, the accuracy of \(\text{length}\) remains unchanged since \(\text{length}\) is of type ratio. However, the accuracy of \(\text{producer}\) is estimated using the production of \(P(\text{movie_name, movie_year}|\text{name, age})\) and \(P(\text{producer}|\text{name, year})\).

For the tuple assignment accuracy value, we follow the lead of Dey and Sarkar[4] and use the product of the two tuple assignment accuracy values. Since all joins are assumed to be over foreign keys, this approach extends to an arbitrary number of relations.

Table 4.6 is an example of the result relation after joining the relation \(\text{emp}\) in Table 4.1 and \(\text{dept}\) in Table 4.5. The assignment accuracy of each tuple is the product of the tuple assignment accuracy values from the two relations.
Table 4.5  Relation *Dept*

<table>
<thead>
<tr>
<th>deptNo</th>
<th>deptName (1.0)</th>
<th>Location</th>
<th>Tuple Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>IT</td>
<td>A (0.8)</td>
<td>0.8</td>
</tr>
<tr>
<td>002</td>
<td>Sales</td>
<td>B (0.4)</td>
<td>0.4</td>
</tr>
<tr>
<td>003</td>
<td>HR</td>
<td>C (0.9)</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Table 4.6  Relation formed by joining *Emp* and *Dept*

<table>
<thead>
<tr>
<th>empNo</th>
<th>lName (0.9)</th>
<th>fName (0.7)</th>
<th>deptNo</th>
<th>Salary (±10)(0.99)</th>
<th>deptName</th>
<th>Location</th>
<th>TupleAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>9527</td>
<td>Chow</td>
<td>Stephen</td>
<td>001(.8)</td>
<td>55,000</td>
<td>IT(.8)</td>
<td>A(.64)</td>
<td>0.443</td>
</tr>
<tr>
<td>9528</td>
<td>Smith</td>
<td>John</td>
<td>002(1.0)</td>
<td>40,000</td>
<td>Sales(1.0)</td>
<td>B(.4)</td>
<td>0.277</td>
</tr>
<tr>
<td>9529</td>
<td>Bond</td>
<td>James</td>
<td>001(.8)</td>
<td>60,000</td>
<td>IT(.8)</td>
<td>A(.64)</td>
<td>0.443</td>
</tr>
<tr>
<td>9530</td>
<td>White</td>
<td>Lucy</td>
<td>001(.8)</td>
<td>55,000</td>
<td>IT(.8)</td>
<td>B(.64)</td>
<td>0.443</td>
</tr>
</tbody>
</table>

### 4.4 Union

The union operation is useful in inserting new data into a relation. According to the definition in the relational database model, the union of two relations is a new relation containing all the rows which are in either of the two participating relations. In the accuracy relational database model, still we need to consider the issue of accuracy for union.

In general, accuracy for union is undefined, since there does not have to be any semantic correlation between the attributes in the two participating relations of union. For example, we have two relations of union, *Horse( Name, Trainer, Weight)* and *Man(Name, Address, Phone)*. In each relation, the first two attributes are of type *string*, and the third attribute are of type *integer*. These two relations can be unioned although there is no semantic correlation between the attributes. Here accuracy has no meaning in representing the quality of data in the resulting relation.

However, there is a special case when the attributes match semantically and they have the same key. Then accuracy makes sense here. Since in most cases accuracy information is retrieved through sampling, the sample spaces should be the same for both participating relations of union. A very common use of union is to take results on a monthly or seasonly basis and then query all data using union. For example, the employee information is stored in
the database on a seasonly basis. Now we have two relations, \textit{emp1} with information about employees hired in season one of the year 2004, and \textit{emp2} with information about those hired in season two of 2004. For both relations, the samples were taken from the same sample space where all employee information is stored. The possible difference is when they were taken, and in most cases the time of sampling will be unique, then the tuples of the participating relations are likely to have a \textit{date} attribute. In this case, the accuracy in the resulting relation is the most up-to-date corresponding accuracy value in the participating relations.

Table 4.7 is a modification of Table 4.1 with date attribute attached. Table 4.8 is another instance \textit{emp2} of the same relation scheme. The result relation after the union of relation \textit{emp1} and \textit{emp2} is shown in Table 4.9.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{Emp1} & & & & & & \\
\textbf{Emp1} & \textbf{empNo} & \textbf{fName} (0.7) & \textbf{deptNo} & \textbf{Salary (±10)}(0.99) & \textbf{Date} (1.0) & \textbf{TupleAcc} \\
\hline
9527 & Chow & Stephen & 001 (0.8) & 54,995 & 2004.S1 & 0.554 \\
9528 & Smith & John & 002 (1.0) & 40,000 & 2004.S1 & 0.693 \\
9529 & Bond & James & 001 (0.8) & 60,000 & 2004.S1 & 0.554 \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{Emp2} & & & & & & \\
\textbf{Emp2} & \textbf{empNo} & \textbf{fName} (1.0) & \textbf{deptNo} & \textbf{Salary (±20)}(0.99) & \textbf{Date} (1.0) & \textbf{TupleAcc} \\
\hline
9530 & White & Lucy & 001 (0.8) & 55,000 & 2004.S2 & 0.713 \\
9531 & Silver & Rachel & 003 (1.0) & 45,000 & 2004.S2 & 0.891 \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{Table 4.9 Relation emp1 \cup emp2} & & & & & & \\
\textbf{emp1} & \textbf{emp2} & \textbf{empNo} & \textbf{fName} (0.9) & \textbf{deptNo} & \textbf{Salary (±20)}(0.99) & \textbf{Date} (1.0) & \textbf{TupleAcc} \\
\hline
9527 & Chow & Stephen & 001 (0.8) & 54,995 & 2004.S1 & 0.713 \\
9528 & Smith & John & 002 (1.0) & 40,000 & 2004.S1 & 0.891 \\
9529 & Bond & James & 001 (0.8) & 60,000 & 2004.S1 & 0.713 \\
9530 & White & Lucy & 001 (0.8) & 55,000 & 2004.S2 & 0.713 \\
9531 & Silver & Rachel & 003 (1.0) & 45,000 & 2004.S2 & 0.891 \\
\hline
\end{tabular}
\end{table}

In the result relation, the accuracy of attribute \textit{fName} is updated to 0.9, which is the accuracy of the attribute taken in season 2, 2004. Also, the accuracy of \textit{Salary} is also updated,
so is the assignment tuple accuracy.

4.5 Difference

The difference operation is useful in deleting old data from a relation. According to the definition in relational database model, the difference of two relations is a relation containing all rows which are in the first relation but not in the second. In the accuracy database model, we keep this definition unchanged, and if the first relation has a tuple with the same key as that of a tuple in the second relation, this tuple will be deleted from the first tuple, regardless the difference in the accuracy of attribute values in the two tuples. Using relation \( emp \) in Table 4.1 as the first relation and \( emp' \) shown in Table 4.10 as the second, we obtain the relation \( emp - emp' \) shown in Table 4.11.

Table 4.10 Relation \( emp' \)

<table>
<thead>
<tr>
<th>empNo</th>
<th>lName  (1.0)</th>
<th>fName (0.9)</th>
<th>deptNo</th>
<th>Salary (±20)(0.99)</th>
<th>TupleAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>9528</td>
<td>Smith</td>
<td>John</td>
<td>002</td>
<td>40,000</td>
<td>0.802</td>
</tr>
<tr>
<td>9531</td>
<td>Silver</td>
<td>Rachel</td>
<td>003</td>
<td>45,000</td>
<td>0.891</td>
</tr>
</tbody>
</table>

Table 4.11 Relation \( emp - emp' \)

<table>
<thead>
<tr>
<th>empNo</th>
<th>lName  (1.0)</th>
<th>fName (0.7)</th>
<th>deptNo</th>
<th>Salary (±10)(0.99)</th>
<th>TupleAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>9527</td>
<td>Chow</td>
<td>Stephen</td>
<td>001</td>
<td>54,995</td>
<td>0.554</td>
</tr>
<tr>
<td>9529</td>
<td>Bond</td>
<td>James</td>
<td>001</td>
<td>60,000</td>
<td>0.554</td>
</tr>
<tr>
<td>9530</td>
<td>White</td>
<td>Lucy</td>
<td>001</td>
<td>55,000</td>
<td>0.554</td>
</tr>
</tbody>
</table>

4.6 Intersection

According to the definition in relational database model, the intersection of two relations is a relation containing all rows that are in both relations. In the accuracy database model, two tuples from two relations separately may share the same key values, but their non-key values may be different or have different accuracy. Therefore, the intersection of two relations in the
accuracy model is a relation containing all rows whose key values are in both relations. As for the non-key attribute values and accuracy, we follow the following principles:

(1) If one attribute value has accuracy $p$ and the other has not, then we take the value with accuracy as the attribute value in the tuple of the result relation, and the accuracy is $p$.

(2) If both attribute values have accuracies, and both are value-dependent, then we take lower accuracy, say $p$, of the value, say $v$, as the attribute level accuracy of $v$. At the same time, we also need to update the attribute accuracy to $p$ in each of the tuples whose value of this attribute is also $v$.

(3) If both attribute values have accuracies, and both are value-independent, then we take the lower accuracy as the attribute level accuracy in the tuple of the result relation.

We do not consider the case that one attribute accuracy value is value-dependent and the other is value-independent since whether the accuracy is value-dependent or not is associated with the attribute name.

Still using Table 4.1 and Table 4.10 as an example of the two participating relations, the intersection relation is shown in Table 4.12.

<table>
<thead>
<tr>
<th>empNo</th>
<th>lName</th>
<th>fName</th>
<th>deptNo</th>
<th>Salary (±20)</th>
<th>TupleAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>9528</td>
<td>Smith</td>
<td>John</td>
<td>002</td>
<td>40,000</td>
<td>0.624</td>
</tr>
</tbody>
</table>

In Table 4.12, the accuracy value of attribute $fName$ is taken as the lower one of the two in the participating relations. As the result, the tuple accuracy is also recalculated.
CHAPTER 5. ACCURACY RELATIONAL ALGEBRA

5.1 Definitions of the Operations

In this section, we will provide the formal definitions of the accuracy relational operations.

Selection. Let \( r \) be a relation on scheme \( R \). Let \( P \) be a selection predicate formed by attributes in \( R \), comparators, constants in the domain of \( A \) for all \( A \in R \), accuracy information, and logical connectives. The selection on \( r \) for \( P \), written \( \sigma_P(r) \), is the set \( \{ x \in r | P(x) \} \).

Projection. Let \( r \) be a relation on scheme \( R \), and let \( S \subseteq R \). The projection of \( r \) onto \( S \) is defined as

\[
\Pi_S(r) = \begin{cases} 
\{ x(S) | x \in r \text{ and } x(\text{tuple\_accuracy}) = \bot \}, & \text{if } \text{key} \notin S \\
\{ x(S) | x \in r \text{ and } x(\text{tuple\_accuracy}) = P(S|\text{key}) \}, & \text{if } \text{key} \in S 
\end{cases}
\]

Natural Join. Let \( r \) and \( s \) be any two relations on schemes \( R \) and \( S \) respectively. The natural join of \( r \) and \( s \) is defined as:

\[
r \bowtie s = \{ x(R \cup S) | \exists (y \in r \land z \in s), (x(R) = y(R)) \land (x(S) = z(S)) \\
\land (x(\text{tuple\_accuracy}) = y(\text{tuple\_accuracy}) \cdot z(\text{tuple\_accuracy})) \}
\]

Note that the attributes in \( R \) and \( S \) should be independent for the natural join operation to yield meaningful results.

Union. Let \( r \) and \( s \) be relations on the same scheme \( R \). Then the union of these two relations is defined as:

\[
r \cup s = \{ x(R) | (x \in r \land x \notin s) \lor (x \in s \land x \notin r) \lor \\
(\exists y \in r \land \exists z \in s \land (\forall A \in R, x(A) = y(A) = z(A))) \}
\]
Note that accuracy for union is undefined in general except for several special cases.

**Difference.** Let \( r \) and \( s \) be relations on the same scheme \( R \). Let \( K \) be the primary key of \( R \). Then the difference of these two relations is defined as:

\[
r - s = \{ x(R) | x \in r \land (\forall y \in s, x(K) \neq y(K)) \}
\]

**Intersection.** Let \( r \) and \( s \) be relations on the same scheme \( R \). Then the intersection of these two relations is defined as:

\[
r \cap s = \{ x(R) | (\exists y \in r \land \exists z \in s \land (\forall A \in R, x(A) = y(A) = z(A)) \land x(accuracy) = \min(y(accuracy), z(accuracy))) \}
\]

**Theta-join.** Let \( r \) and \( s \) be any two relations on schemes \( R \) and \( S \) respectively, and let \( P \) be any predicate formed by attributes in \((R \cup S)\), comparators, constants in the domain of \( A \) for all \( A \in (R \cup S)\), accuracy information, and logical connectives. The theta-join between \( r \) and \( s \) is given by

\[
r \bowtie_P s = \sigma_P(r \bowtie s)
\]

### 5.2 Properties of the Accuracy Relational Model

Let \( R, S \) and \( T \) be relation schemes. Let \( r \) be a relation on scheme \( R \), let \( s, s_1, s_2 \) be relations on scheme \( s \), and let \( t \) be a relation on scheme \( T \). Let \( Q \) and \( W \) be subsets of \( R \) and \( Q \subset W \). Let \( P, P_1, P_2 \) be any selection predicate involving attributes of \( S \). Then, the following equations hold:

(a) \( \sigma_{P_1}(\sigma_{P_2}(s)) = \sigma_{P_2}(\sigma_{P_1}(s)) = \sigma_{P_1 \land P_2}(s) \)

(b) \( \Pi_Q(\Pi_W(r)) = \Pi_Q(r) \)

(c) \( r \bowtie s = s \bowtie r \)

(d) \( r \bowtie s \bowtie t = r \bowtie (s \bowtie t) \)

(e) \( r \cup s = s \cup r \)
(f) \( r \cup s \cup t = r \cup (s \cup t) \)

(g) \( r \wedge (s_1 \cup s_2) = (r \wedge s_1) \cup (r \wedge s_2) \)

(h) \( r \wedge (s_1 - s_2) = (r \wedge s_1) - (r \wedge s_2) \)

(i) \( \sigma_P(s_1 \cup s_2) = \sigma_P(s_1) \cup \sigma_P(s_2) \)

(j) \( \sigma_P(s_1 - s_2) = \sigma_P(s_1) - \sigma_P(s_2) \)

**Proof:** Straightforward from the definitions of the operators.
CHAPTER 6. ACCURACY RELATIONAL QUERY LANGUAGE

To implement the accuracy relational model, we propose the Accuracy Structured Query Language (ASQL). The syntax of ASQL is similar with that of SQL in that it also uses the SELECT, FROM, and WHERE clauses, but it extends the SQL syntax to allow users to introduce restrictions on what they will accept for accuracy levels. In this chapter, we discuss the forms of queries in ASQL.

Basically, there are four forms of queries in ASQL.

1. queries in the form of traditional SQL

ASQL is an extension of SQL, therefore, it still supports queries expressed using SQL. Sometimes users do not care about accuracy information. They only want to query on the data. In ASQL, this form of queries is expressed in the same way as it was in SQL, i.e., it is still of the SELECT-FROM-WHERE pattern.

Example 1. Write the ASQL to list the employee number, first name, last name, department number, depart name, department location of those employees with all accuracy information returned by the query.

The ASQL query can be written as the following:

\[
\text{SELECT ACCURACY empNo, fName, lName, deptNo, deptName, location}
\]
FROM emp, dept
WHERE emp.deptNo = dept.deptNo;

This query will return the accuracy of each attribute and every tuple in the resulting relation, together with the data stored in the database.

3. queries for data that satisfy some accuracy condition for value-dependent attributes.

In this case users can add restrictions to the attribute accuracy, and the resulting data should satisfy the accuracy condition. The accuracy condition is expressed in a pair of parentheses following the corresponding attribute in the SELECT clause.

Example 2. Write the ASQL to list the employee number, first name, last name, department number, department name, department location of those employees with the accuracy levels of the attribute department number in the resulting relation higher than 0.8.

The ASQL query could be written as the following:

```
SELECT empNo, fName, lName, deptNo(>0.8), deptName, location
FROM emp, dept
WHERE emp.deptNo = dept.deptNo;
```

In this form of queries we should notice that we can not add accuracy restrictions to those attributes with accuracy that is value-independent. The reason is that if every value of the attribute has the same accuracy, then we can not distinguish the results based on the accuracy, and we can hardly get any useful information.

4. queries for data that satisfy some tuple accuracy condition

Users can also add restrictions to the tuple assignment accuracy of the resulting relation. In this form of queries, a WITH ACCURACY clause needs to be included, with the accuracy restriction specified after the clause keywords.

Example 3. Write the ASQL to list the employee number, first name, last name, salary, department number, department name, department location of those employees whose annual salary is higher than $55,000 with the accuracy levels of the tuples in the resulting
relation higher than 0.6, and the accuracy of the attribute department number in the resulting relation higher than 0.8.

The ASQL query could be written as the following:

```
SELECT e.empNo, e.fName, e.lName, e.Salary, e.deptNo(>0.8), d.deptName, d.location
FROM emp e, dept d
WHERE e.deptNo = d.deptNo and e.Salary>55000
WITH ACCURACY >0.6;
```

This query involves projection and join operations. The accuracy constant of 0.6 is to be compared with the accuracy levels of tuples in the resulting relation, which is the joined relation from the two participating relations: *emp* and *dept*. In the next chapter, we will discuss in detail on how the preprocessor would evaluate this ASQL query and translate it into an SQL query ready to use for the conventional relational database.
CHAPTER 7. PROTOTYPE

In this chapter, we discuss the implementation issues of the accuracy relational model. From the previous discussion on the accuracy relational model, we can see that the accuracy relational model is an extension of the conventional relational model, with the handling capabilities of accuracy information included. Thus we can make use of the conventional relational database, with a preprocessor which takes the Accuracy Structured Query Language (ASQL) based on the accuracy model, evaluates the query, makes some transitions on the input query and then sends the transformed query into the relational database. Figure 7.1 shows the function of the preprocessor:

![Figure 7.1 The function of the query preprocessor](image)

7.1 Query Evaluation

This section will discuss on how the query preprocessor evaluates the query written in ASQL and translates the query into SQL ready to use for the conventional relational database.

Step 1. Parse the ASQL query and extract SELECT, FROM, WHERE, and WITH ACURACY clauses.

The input of the query preprocessor is a query string written in ASQL, such as the query written in example 1. After the first step, the query preprocessor will generate four strings:

- **select-string**: the string between “SELECT” and “FROM” in the ASQL query
from-string: the string between “FROM” and “WHERE” in ASQL query

where-string: the string between “WHERE” and “WITH ACCURACY” in the ASQL query if there is a WITH ACCURACY clause, otherwise where-string is the string after “where” ASQL query

accuracy-string: the string after “WITH ACCURACY”.

All these strings can be easily extracted with the string operations. If there is no WITH ACCURACY clause in the query, then the accuracy-string is set to be NULL.

Step 2. Parse the SELECT clause to get accuracy information, if there is any.

The select-string may contain the ACCURACY keyword at the beginning, since the query may start with SELECT ACCURACY. If there is such keyword in select-string, the preprocessor will find the parentheses and extract the accuracy information specified inside, assemble a query condition, and add the accuracy query condition to the WHERE clause.

Step 3. Parse the FROM clause to get the number of relations involved in the query.

In this step, the preprocessor needs to get the number of participating relations and store the information of each relation, such as relation name and alias. This information is useful when parsing the select-string and the where-string. Here we use a data structure named relation. It is defined as the following in Java:

```java
class relation {
    String name;
    String alias;
}
```

When the preprocessor reads the information of a relation, it will initiate the class relation and set the values of name and alias with the constructor function. If the there is no alias mentioned in the FROM clause, the variable alias is set as NULL.

Step 4. Parse the WHERE clause to verify each condition.

If the selection condition in the WHERE clause contains ordinal or ratio attributes, then the preprocessor should modify the condition according to the range error $\pm e$. For example,
if the query contains a condition of $Salary > 55000$ in the WHERE clause and the accuracy of $Salary$ is $\pm 10$, then the preprocessor will modify the condition as $Salary > 54990$.

**Step 5. Translate the ASQL query into SQL query.**

With the select, from, where and accuracy strings ready, as well as the information of the participating relations stored, the preprocessor can use some algorithms to interpret the ASQL query.

If the number of relations is 1, i.e., there is only one relation involved in the query, then we can tell that there is no join operation in the query, but only selection or projection or both are involved in the ASQL query. In this simple case, the preprocessor only needs to deal with the accuracy level in the SELECT ACCURACY clause and in the WITH ACCURACY clause. It will rewrite the query into SQL with all accuracy information moved to the WHERE clause.

For example, if the query is written in ASQL as:

```sql
SELECT ACCURACY empNo, fName, lName, deptNo (> 0.8), Salary
FROM emp
WHERE Salary > 55000
WITH ACCURACY > 0.6;
```

The query involves projection and selection, and there are two accuracy conditions: the accuracy condition in the SELECT ACCURACY clause: $deptNo (> 0.8)$, and the accuracy condition in the WITH ACCURACY clause: $accuracy > 0.6$. Also, in WHERE clause, there is a ratio attribute $Salary$ involved. Then the corresponding SQL query is:

```sql
SELECT empNo, fName, lName, deptNo
FROM emp
WHERE Salary > 54990 and deptNo_acc > 0.8 and tuple_acc > 0.6;
```

On the other hand, if there is more than one relation involved in this query, then a join operation has been introduced. In this case, except taking care of all accuracy information, the preprocessor also needs to ensure that the join operation is over a foreign key of one participating relation. To achieve this purpose, we define a data structure `joinrel`, which stores the information of the pair of relations that are joined over a set of attributes. In java, the structure is defined as the following:

```java
class joinrel {
}
```
Relation1 and relation2 are the name of the two relations joined together. List1 and List2 are two lists of attributes over which the join operation is posed on the two relations respectively. These information could be extracted from the WHERE clause. If list1 contains the key of relation2, then the two relations are joined over the foreign key of relation1. Similarly, if list2 contains the key of relation1, then the two relations are joined over the foreign key of relation2.

After ensuring that the relations are joined over the foreign key of one relation, the accuracy level of each tuple in the resulting joined relation can be estimated using the method we have mentioned in the model design chapter, and the preprocessor can assemble the SQL query statement thereafter.

Note that the preprocessor also needs to take care of the accuracy conditions in the SELECT ACCURACY clause and in the WITH ACCURACY clause. The accuracy level of each tuple in the joined relation is estimated by taking the product of the accuracy levels of the tuples in the participating relations.

7.2 An Example

Here we will demonstrate on how the preprocessor interprets an ASQL query and turns it into an SQL query. We rewrite the ASQL in Example3:

```sql
SELECT ACCURACY e.empNo, e.fName, e.lName, e.Salary, e.deptNo(> 0.8), d.deptName, d.location
FROM emp e, dept d
WHERE e.deptNo = d.deptNo and e.Salary > 55000
WITH ACCURACY > 0.6;
```

This query has two relations involved: the employee relation `emp` and the department relation `dept`. The schemes of these relations are the same as they were defined in the pre-
vious chapters. Three operations are involved in this query: selection, projection, and join. Also, both the SELECT ACCURACY clause and WITH ACCURACY clause have accuracy conditions. Here are the three steps the preprocessor will take during interpretation.

(1) The four strings are extracted from the input query statement.

select-string: ACCURACY e.empNo, e.fName, e.lName, e.Salary, e.deptNo(L, 0.8), d.deptName, d.location

from-string: emp e, dept d

where-string: e.deptNo = d.deptNo and e.Salary > 55000

accuracy-string: > 0.6

(2) The preprocessor gets the number of relations and initiate the two instances of the class relation.

By interpreting the from-string, the preprocessor sets the variable which denotes the number of relations as 2. Also, it will initiate the two instances of class relation and store them into a vector of relations:

<table>
<thead>
<tr>
<th>element</th>
<th>name</th>
<th>alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>emp</td>
<td>e</td>
</tr>
<tr>
<td>2</td>
<td>dept</td>
<td>d</td>
</tr>
</tbody>
</table>

(3) Since there are two relations involved in this query, there should be a join operation. Therefore, the where-string is parsed to get the information that which operations are joined together. The result will be stored in an instance of the class joinrel for each pair of joined relations. A vector is also used to store these instances. In this example, the vector has only one element.

<table>
<thead>
<tr>
<th>element</th>
<th>relation1</th>
<th>relation2</th>
<th>list1</th>
<th>list2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>emp</td>
<td>dept</td>
<td>deptNo</td>
<td>deptNo</td>
</tr>
</tbody>
</table>

By examining the lists of attributes being joined over from the two relations, the preprocessor is able to tell that the two relations are joined over the foreign key of relation
emp. Therefore, it can estimate the accuracy level of each tuple in the resulting joined relation by taking the product of the assignment accuracy of the tuples from the two participating relations.

Since there are accuracy conditions specified both in the select-string and the accuracy-string, the preprocessor will translate the original ASQL query statement into an SQL query statement. The translated SQL statement is:

```
SELECT e.empNo, e.fName, e.lName, e.Salary, e.deptNo, d.deptName, d.location
FROM emp e, dept d
WHERE e.deptNo = d.deptNo and Salary > 54990 and e.deptNo_acc > 0.8 and e.tuple_acc * d.tuple_acc > 0.6;
```
CHAPTER 8. CONCLUSION

Uncertainty in databases has become an important issue in relational databases. Previous research efforts have been developed to deal with uncertainty. However, few of them addressed the accuracy issue. Because of this concern, we have proposed an extension to the relational databases to support accuracy. A statistical approach to estimate accuracy is stated in this thesis. Accordingly, relational algebra has been extended for accuracy support and ASQL (Accuracy Structural Query Language) is also provided as an extension to SQL.
BIBLIOGRAPHY


ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis.

First and foremost, Dr. Leslie Miller for his guidance, patience and support throughout this research and the writing of this thesis.

Second, Dr. Sarah M. Nusser for her advice on the statistical part of the model.

I would also like to thank Dr. Shashi K. Gadia for his efforts and contributions to be a member of my POS committee.