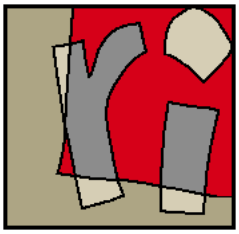
Approximation of OLAP queries on data warehouses

Phuong Thao Cao



UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE INFORMATIQUE PARIS-SUD

Laboratoire de recherche en informatique

DISCIPLINE: INFORMATIQUE

THÈSE DE DOCTORAT

soutenue le 20/06/2013

par

Phuong Thao CAO

Approximation of OLAP queries on data warehouses

Jury

Directeur de thèse	Pr. Michel De Rougemont	Université Panthéon-Assas
Co-directeur de thèse	Pr. Nicolas Spyratos	Université Paris-Sud
Président	Pr. Khaldoun Al Agha	Université Paris-Sud
Rapporteur	Pr. Mohand-Said Hacid	Université Lyon 1
Rapporteur	Pr. Kavé Salamatian	Université Savoie
Examinateur	Pr. Halim Msilti	Université Panthéon-Assas

Abstract

We study the approximate answers to OLAP queries on data warehouses. We consider the relative answers to OLAP queries on a schema, as distributions with the L_1 distance and approximate the answers without storing the entire data warehouse. We first introduce three specific methods: the uniform sampling, the *measure* -based sampling and the statistical model. We introduce also an edit distance between data warehouses with edit operations adapted for data warehouses. Then, in the OLAP data exchange, we study how to sample each source and combine the samples to approximate any OLAP query. We next consider a streaming context, where a data warehouse is built by streams of different sources. We show a lower bound on the size of the memory necessary to approximate queries. In this case, we approximate OLAP queries with a finite memory. We describe also a method to discover the statistical dependencies, a new notion we introduce. We are looking for them based on the decision tree. We apply the method to two data warehouses. The first one simulates the data of sensors, which provide weather parameters over time and location from different sources. The second one is the collection of RSS from the web sites on Internet.

Keywords: OLAP, approximate query answering, OLAP data exchange, streaming data, edit distance, sampling algorithm, statistical dependencies, statistical model.

Résumé

Nous étudions les réponses proches à des requêtes OLAP sur les entrepôts de données. Nous considérons les réponses relatives aux requêtes OLAP sur un schéma, comme les distributions avec la distance L_1 et rapprocher les réponses sans stocker totalement l'entrepôt de données. Nous présentons d'abord trois méthodes spécifiques: l'échantillonnage uniforme, l'échantillonnage basé sur la *mesure* et le modèle statistique. Nous introduisons également une distance d'édition entre les entrepôts de données avec des opérations d'édition adaptées aux entrepôts de données. Puis, dans l'échange de données OLAP, nous étudions comment échantillonner chaque source et combiner les échantillons pour rapprocher toutes requêtes OLAP. Nous examinons ensuite un contexte streaming, où un entrepôt de données est construit par les flux de différentes sources. Nous montrons une borne inférieure de la taille de la mémoire nécessaire aux requêtes approximatives. Dans ce cas, nous avons les réponses pour les requêtes OLAP avec une mémoire finie. Nous décrivons également une méthode pour découvrir les dépendances statistique, une nouvelle notion que nous introduisons. Nous recherchons ces dépendances en basant sur l'arbre de décision. Nous appliquons la méthode à deux entrepôts de données. Le premier simule les données de capteurs, qui fournissent des paramètres météorologiques au fil du temps et de l'emplacement à partir de différentes sources. Le deuxième est la collecte de RSS à partir des sites web sur Internet.

Mots-clés: OLAP, réponses proches de la requête, échange de données OLAP, des flux de données, distance d'édition, algorithme d'échantillonnage, dépendances statistiques, modèle statistique.

Acknowledgments

First and foremost, I am very grateful to professor Dr. Michel de Rougemont, my supervisor, for his guidance and encouragement throughout this thesis work. His rich experience and support has been invaluable to me. I would like to thank my supervisor, professor Dr. Nicolas Spyratos, for introducing me to database team, for review my work and making useful suggestions.

Deep thanks to Tsuyoshi Sugibuchi for his priceless technical support and advices. Also, I would like to take this opportunity to personally thank all of my colleagues and friends in database team for their help.

I would like to acknowledge professor Dr. Philippe Dague, Director of LRI, for his financial support.

I would like to acknowledge professor Dr. Mohand-Said Hacid, professor Dr. Kavé Salamatian, professor Dr. Halim Msilti and professor Dr. Khaldoun Al Agha for being my thesis committee members and for their advice and comments.

Finally, 1 love to thank my parents and my family for their love, encouragement and support. Without their understanding, my research could not have been realized.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Data warehouses collect the history of many physical processes, such as the sales of items, the measure of sensors, the traffic of data in a network. When these data are collected in a relational form, its analysis is the subject of *OnLine Anaytical Processing* (OLAP). An OLAP schema fixes a set of functional dependencies between attributes, and defines possible dimensions. The answer to OLAP queries can be considered as distributions, such as the distribution of sales per country or the distribution of sensors measures per city.

In practice, approximate answers to OLAP queries [1, 8, 11, 12, 18, 19, 25], may be sufficient, and could be obtained much more efficiently than exact answers. The algorithmic complexity theory studies the tradeoff between approximation and efficiency. In many cases we can use *randomized algorithms* which achieve an ε approximation with a high confidence $(1 - \delta)$ and are much more efficient than deterministic algorithms. In the case of large data warehouses, we exhibit such algorithms for the approximation of OLAP queries. We consider two sampling methods: a uniform sampling and a measure-based sampling which both give good approximations.

Data warehouses are built by collecting data from different Sources, and assembling them. The theory of *Data Exchange* studies how to efficiently decide consistency constraints given the Sources. We extend this approach to OLAP queries and ask if we can directly sample the sources, collect some statistical information, and approximately answer OLAP queries from the statistics, i.e. without storing the whole data. Similar questions concern the *Streaming model* [2, 9, 10, 23, 29], i.e.

when each Source streams data to a data warehouse. Could we replace the whole large data warehouse by some other data using logarithmic space? We answer both these questions, by generalizing the sampling approach.

The main results of the thesis concern the approximation of OLAP queries in a data exchange setting, and in the special case of *statistical dependencies*, a notion we introduce. It generalizes the notion the functional dependencies, as some attributes may imply fixed distributions on the measure. In this case, we can reduce the data to some limited statistics. We also study how to discover such dependencies by generalizing the construction of decision trees in data mining.

1.1 State of the art

Most database systems have modules which offer the possibility of OLAP analysis. Some originated from the relational model (Oracle, DB2, MS SQL/Server) and later included modules for the analysis, whereas some other systems (SAS) started with the analysis and later included the classical relational model.

In these systems, the data are well structured in a data warehouse and OLAP schema are constructed and modified. Queries are specified by filters, dimensions and aggregation operators, and results need to have good graphical representations. Dashboards integrate the graphical outputs. For open source software, *Pentaho* [17] is one of the most complete systems, which integrate modules for ETL (Extract, Transform, Load), generation for schemas and graphical interface (Jpivot). We will use this system in our implementation.

OLAP tools for semi structured data are less developed, and the analysis of massive data remains a research area. The classical systems maintain a physical data warehouse which can be extremely large.

We propose approximate solutions which may apply to the current technology or may extrapolate to new systems. In the first case, we just *sample* the data warehouse and obtain approximate answers more efficiently. In the second case, we propose a general setting, where we can replace the large data warehouse by some statistics and still keep the approximate OLAP analysis. This may apply to massive or streaming data.

The general question we ask is: can we replace these data by some statistics? In the case of OLAP analysis, when the attributes satisfy some statistical hypothesis (independence), we answer positively.

1.2 Thesis plan

This thesis is structured as follows:

Chapter 1 gives a brief overview about the context of our research. We present the objective and the main results.

In Chapter 2, we review the main notions in OLAP system: data warehouses and schemas. Moreover, we describe the components of a query and the definition of relative answers which we use. In this chapter, our new approximate methods are introduced briefly. These methods are based on the sampling techniques. The quality of our sampling algorithms is guaranteed by using Hoeffding-Chernoff bounds- the probability inequalities for the sum of bounded random variables. At the end of this chapter, we present the different contexts in which we study the approximation algorithms.

Our main contributions appear in Chapter 3, Chapter 4, Chapter 5, Chapter 6, Chapter 7 and Chapter 8.

The Chapter 3 introduces two specific methods: the uniform sampling and the measure-based sampling. They use two probabilistic spaces to approximate the answers to the OLAP query. These two methods produce a much smaller data warehouse on which we can approximate OLAP queries. We will prove that these two algorithms can approximate the answers with good accuracy ε and with high probability $1 - \delta$.

In Chapter 4, we study an edit distance between data warehouses. We introduce edit operations adapted for data warehouses. Moreover, we define the distance betweens two data warehouses. We show that this edit distance is a metric, i.e. it is always in $[0, 1]$, symmetric and has the triangular inequality property. We then prove the continuity theorem that the close data warehouses imply that the answers to OLAP queries must also be close.

In Chapter 5, we present the statistical model based on the statistical dependency. The core of this method is the reduction of data warehouse by one more compact structure: some limited statistics. Then, we combine these limited statistics and the statistical dependencies to approximate the OLAP query. Finally, we show how to learn the statistics of the model.

The Chapter 6 covers the approximation in the context of OLAP data exchange. In this context, we present the different approximate algorithms: the approximation with the uniform distribution, the approximation with the measure-based sampling, and the approximation by the statistical model.

In Chapter 7, we consider streaming data. We first show some lower bounds on the

space complexity for unbounded domains. For unbounded domains, we design the counters for the exact distribution used by statistical model.

In Chapter 8, we describe a method to discover the statistical dependencies. We may not know these dependencies. We generalize the construction of decision trees in data mining. In a classical decision tree, some attributes predict the value of the target attribute M. This value is predicted with high probability. In the case of statistical dependency, we built the decision tree in which some attributes predict the distribution of values of the target attribute with high probability.

In Chapter 9, we test our approximation methods: the approximation by uniform sampling, the approximation by measure-based sampling and by the statistical model. We use a data warehouse simulates sensors and a real data warehouse collecting RSS from the web sites on Internet. For each method, we are interested in the quality of approximation such as the error ratio and the confident level. At the end we analyze the results and compare the methods.

In Chapter 10, we present the conclusion and the future work.

CHAPTER 2

Preliminaries

2.1 Introduction

This chapter introduces basic notions, definitions and results which are used in this thesis. We first describe the notations for OLAP (On-Line Analytical Processing) such as schemas, queries, query answering. We then present the approximation model by mainly using the sampling technique. Finally, these methods will be used in the context of data exchange, of distributed data and of streaming data.

2.2 On-Line Analytical Processing (OLAP)

On-Line Analytical Processing, or OLAP for short is the main activity carried out by analysts and decision makers [27]. The applications are widely used to assist management from dashboards, which assemble graphical representations of OLAP queries. We present the notions and the definitions of schemas, data warehouses, queries and answers to queries.

2.2.1 Schemas

We follow the functional model associated with an OLAP schema [27].

The OLAP or star schema is a tree where each node is a set of attributes, the root is the set of all the attributes of the data warehouse relation, and an edge exists if there is a functional dependency between the attributes of the origin node and the

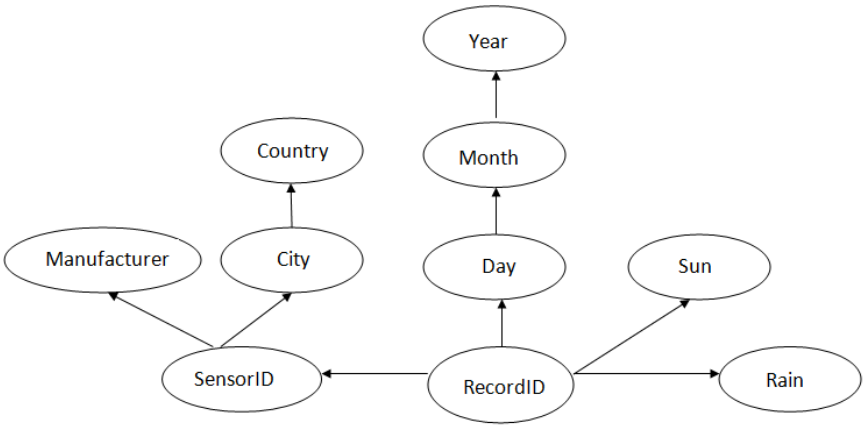


Figure 2.1: An OLAP schema.

attributes of the extremity node. The *measures* are specific nodes at depth 1 from the root.

Example 2.2.1. *Consider the simple OLAP schema of Figure 2.1, where the functional dependencies follow the edges up. The* measures *are sun and rain.*

2.2.2 Relational representation

We describe the mapping from a schema to a relational representation.[27]

Definition 2.2.1. *Let S be an OLAP schema, the* relational representation *of S consists of a fact table and a set of dimensional tables:*

- *All nodes at depth* 1 *from the root constitutes the* fact table. *These nodes called dimensions are considered as the key of the fact table except for the* measures.
- *Every dimension D of the fact table gives rise to a* dimensional table *defined as follows. If $A_1, ..., A_m$ are the attributes appearing in all functional dependencies starting at D, then the* dimensional table *of D is a relational table with attributes $D, A_1, ..., A_m$ having D as its only key.*

Example 2.2.2. *We can represent the OLAP schema of Figure 9.5 by two relations: the fact table DW(recordID, sensorID, day, month, year, sun, rain) which stores every day the* measures *sun and rain, and a dimensional table C(sensorID, manuf, city, country).*

2.2.3 Data warehouses

The fact table is called the *data warehouse*. It provides integrated, consolidated and historic data for analysis. A data warehouse functions just like a database, with the following important differences [27]:

- (Integration) The data of a data warehouse is not production data but the result of integration of production data coming from various sources. The data warehouse is organized by subjects or themes. All the necessary data for performing a particular analysis can be found in the data warehouse.
- (Historic data) The data of a data warehouse can include data accumulated over time.
- (No volatility) The access to the data warehouse by analysts is almost exclusively for reading and not for writing.
- (Periodical updating) The changes of data happen only at the sources, and such changes are propagated periodically to the data warehouse.

2.2.4 Queries and answers

Definition 2.2.2. *An OLAP query for a schema S is determined by:*

1. *A condition filter which selects a subset of the tuples of the data warehouse.*
2. *A* measure.
3. *The selection of dimensions or classifiers, $C_1, ..., C_p$ where each C_i is a node of the schema S.*
4. *An aggregation operator (COUNT, SUM, AVG, ...).*

Assume for simplicity that the aggregation operator is SUM.

Definition 2.2.3. *The relative answer to an OLAP query Q with respect to I is a function:*

$$Q_{C_1,...,C_p} : I \to \sigma = \begin{pmatrix} m_1 \\ ... \\ m_k \end{pmatrix}$$

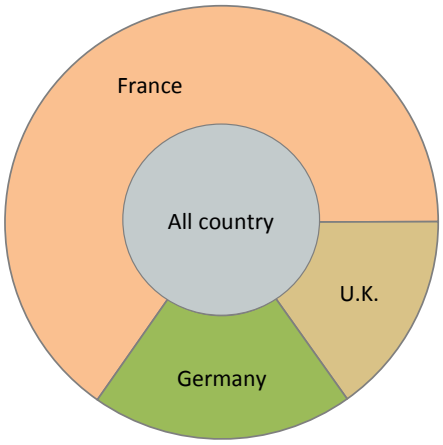
where σ is the density vector with $m_i = \dfrac{\sum_{t.C_1=c_1,...t.C_p=c_p} t.M}{\sum_{t \in I} t.M}$ on the tuples obtained after the filter and $k \leq |C_1| * |C_2| * ... * |C_p|$

We consider relative *measures* as answers to OLAP queries. This vector σ is also a probabilistic distribution. For simplicity, we write Q^I_C as the density vector for the answer to Q on dimension C and on data warehouse I. Each component of Q^I_C is written $Q^I_{C=c}$ or $Q^I_C[c]$ and is the relative density for the dimension $C = c$. In another way, we can interpret $Q^I_{C=c}$ as the probability that a random tuple t is such that $t.C = c$.

Example 2.2.3. *Consider the simple OLAP schema of Figure 2.1, where the functional dependencies follow the edges up.*

	Measures
country	SUM of sunlight
-All country	689,960
France	450,022
Germany	134,958
United Kingdom	104,980

(a) Absolute answer to Q1: Analysis for each *country*.



(b) Piechart answer of Q1: Analysis for each *country*.

Figure 2.2: Representations of answers.

We suppose two relations: the data warehouse *DW(recordID, sensorID, date, month, year, sun, rain)* which stores every day the *measures sun* and *rain* in hours in the interval $[1, 10]$ for all sensors, and an auxiliary table *C(sensorID, manuf, city, country)*.

A typical OLAP query may select *country* as a dimension, asking for the relative *sun* of the countries, and an absolute answer would be as in Figure 2.2(a).

A relative answer would be: (France, 0.65), (Germany, 0.2), (U.K., 0.15), as $0.65 = 450022/689960$.

The *answer to an OLAP query* Q is the vector of relative values. In our example, $Q^I_{country} = (0.65, 0.2, 0.15)$ as represented by a pie-chart in Figure 2.2(b).

2.3 Approximation

The goal of our research is to approximate the answers to OLAP queries. We design randomized algorithms with two parameters $0 \leq \varepsilon, \delta \leq 1$ where ε is the error, and $1 - \delta$ is the confidence. Let us recall the notion of a (ε, δ)-approximation for a function $F : \Sigma^* \to \mathbb{R}$.

Definition 2.3.1. *Let A be a randomized algorithm with input x and output $y = A(x)$. The algorithm A (ε, δ)-approximates the function F if for all x,*

$$Prob[F(x) - \varepsilon \leq A(x) \leq F(x) + \varepsilon] \ \geq \ 1 - \delta$$

In our case, the probabilistic space will be the samples, which may be selected with a uniform distribution. If we have a data warehouse with N samples, we select each tuple t with probability $1/N$. We will call $\widehat{I_u}$ the probabilistic data warehouse obtained by m samples, selected with the uniform distribution. Each one has a probability of $(1/N)^m$.

2.3.1 Approximate query answering

In our context, we approximate vectors whose values are less than 1 which are relative answers to an OLAP query. In this situation, $F(x)$ is $Q^I_{C=c}$ (for all c) and $x = I$. We use randomized algorithms with an additive error and mainly use the L_1 distance between two vectors. Strictly speaking, the L_1 distances are less than 2, and we usually have a $\frac{1}{2}$ factor to normalize the distance in [0..1], which we omit for simplicity.

Definition 2.3.2. *The* distance between two relative answers to an OLAP query *is the L_1 distance between the relative densities.*

Let σ, μ be two distributions. They are ε-close if $|\sigma - \mu|_1 = \sum_v |\sigma(v) - \mu(v)| \leq \varepsilon$.

Example 2.3.1. *For example, the distance between the* $(0.65, 0.2, 0.15)$ *distribution over (France, Germany, U.K.) and the* $(0.6, 0.4)$ *distribution over (France, Germany) is* $(0.05 + 0.2 + 0.15) = 0.4$. *We can see that the distance between two distributions are in* $[0, 2]$.

In our case, the ε, δ-approximation is defined as follows. A randomized algorithm $\hat{Q}$ defines a function:

$$\hat{Q_C} : I \to \widehat{\sigma} = \begin{pmatrix} m'_1 \\ ... \\ m'_k \end{pmatrix},$$

where $k \leq |C|$, and let $\widehat{\sigma} = \hat{Q}_C^I$.

Definition 2.3.3. *The algorithm $\hat{Q}$ (ε, δ)-approximates the function Q_C if for all I,*

$$[Prob[Q_C^I - \varepsilon \leq \hat{Q}_C^I \leq Q_C^I + \varepsilon] \ \geq \ 1 - \delta]$$

The approximation by sampling is introduced in chapter 3, and the approximation by a statistical model is discussed in chapter 5.

2.3.2 Sampling methods

The objective of sampling is to approximate the answer on the small set of samples instead of finding the answers on the entire data warehouse. The randomized algorithms may take samples $t \in I$ with different distributions. The probability is over these probabilistic spaces.

2.3.2.1 Uniform sampling

In this case, we select $\widehat{I_u}$, made of m distinct samples of the data warehouse I, with a uniform distribution on the N tuples. In order to implement the technique, we use a standard random generator. A function $random(N)$ generates an element $i \in_r \{1, 2, ...N\}$ with probability $1/N$. There is a constraint for the number of samples m, in order to have an (ε, δ)-approximation. We will show that in chapter 4 that m only depends on the parameters ε and δ but not on N.

2.3.2.2 Measure-based sampling

This technique samples the data warehouse to have m distinct tuples. But the probability of selection of each tuple depends on its *measure*. Moreover, the algorithm executes in two steps:

1. We first select a tuple t with a uniform distribution
2. Keep t with probability proportional to its *measure*

When we generate the samples $\widehat{I_e}$, we replace the measure by 1. We assume that max is the maximum value of the *measure*. If max is small in relation with the size of the data warehouse, both techniques allow to approximate OLAP queries. But if max is large (unbounded), only the measure-based sampling can be applied on our sense. The context of application and the comparison between two techniques are introduced in chapter 4.

2.4 Probabilistic tools

In this thesis, we use the probability inequalities for the sums of bounded random variables [20],to guarantee that our algorithms (ε, δ)-approximate the queries. These bounds give exponentially decreasing upper bounds on tail distributions (ie. $Pr(X \geq a)$, where $a \geq \mathbb{E}[a]$) that the sums of independent random variables vary from their expectation by some ε.

We need to bound

$$[Prob[Q_C^I - \varepsilon \leq \hat{Q}_C^I \leq Q_C^I + \varepsilon] \ \geq \ 1 - \delta$$

or equivalently that

$$[Prob[|\hat{Q}_C^I - Q_C^I| > \varepsilon] \leq \delta$$

Notice however that $\mathbb{E}(\hat{Q}_C^I) = Q_C^I$. Hence we need to show that:

$$[Prob[|\hat{Q}_C^I - \mathbb{E}(\hat{Q}_C^I)| > \varepsilon] \leq \delta$$

which is of the form $[Prob[|X - \mathbb{E}(X)| > \varepsilon] \leq \delta$ and we say that the error is additive. In the case of $[Prob[|X - \mathbb{E}(X)| > \varepsilon.\mathbb{E}(X)] \leq \delta$ we say that the error is multiplicative.

We first recall the Chernoff bounds before coming to the Hoeffding bounds which we will use.

2.4.1 Chernoff bounds

Chernoff bounds give estimates on the probability that the sum of random variables vary from its expected value.[7]

2.4.1.1 Additive form (absolute error)

Theorem 2.4.1. *(Chernoff-Hoeffding) Assume random variables X_1, X_2 , ..., X_m are independent and identically distributed random variables. Let $p = \mathbb{E}[X_i]$, $X_i \in \{0, 1\}$, and $\varepsilon \geq 0$. Then,*

1. $Pr[\frac{1}{m}\sum X_i \geq p + \varepsilon] \leq ((\frac{p}{p+\varepsilon})^{p+\varepsilon}(\frac{1-p}{1-p-\varepsilon})^{1-p-\varepsilon})^m = e^{-D(p+\varepsilon\|p)m}$
2. $Pr[\frac{1}{m}\sum X_i \leq p - \varepsilon] \leq ((\frac{p}{p-\varepsilon})^{p-\varepsilon}(\frac{1-p}{1-p+\varepsilon})^{1-p+\varepsilon})^m = e^{-D(p-\varepsilon\|p)m}$

where $D(x\|y) = x log\frac{x}{y} + (1-x) log\frac{1-x}{1-y}$.

2.4.1.2 Multiplicative form (relative error)

Theorem 2.4.2. *Let random variables X_1, X_2, ..., X_m be independent variables taking on values 0 or 1 with $Pr(X_i = 1) = p_i$, $X = \sum_{i=1}^{n} X_i$, and $\mu = \mathbb{E}[X]$. Then:*

1. *for any $\delta > 0$,*

$$Pr[X \geq (1+\delta)\mu] < (\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}})^{\mu}$$

and

$$Pr[X \leq (1-\delta)\mu] \leq (\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}})^{\mu}.$$

2. *for any $0 < \delta < 1$,*

$$Pr[X > (1+\delta)\mu] \leq e^{\frac{-\delta^2\mu}{3}}$$

and

$$Pr[X < (1-\delta)\mu] \leq e^{\frac{-\delta^2\mu}{2}}.$$

In the theorem 2.4.2, the second bound for any $0 < \delta < 1$ is derived from the first one. The absolute error ε is greater and easier to compute than the relative error δ.

2.4.1.3 Better Chernoff bounds for some special cases

We can derive many deviations of Chernoff bounds by using different proof techniques. Here are some special cases of symmetric random variables.

Theorem 2.4.3. *Let X be a sum of n independent random variables $\{X_i\}$. Let μ denote the expected value of X. If $Pr(X_i = 1) = Pr(X_i = -1) = \frac{1}{2}$ then*

- $Pr[X \geq a] \leq e^{\frac{-a^2}{2n}}$, *with* $a > 0$
- $Pr[|X| \geq a] \leq 2e^{\frac{-a^2}{2n}}$, *with* $a > 0$.

Theorem 2.4.4. *Let X be a sum of n independent random variables $\{X_i\}$. Let μ denote the expected value of X. If $Pr(X_i = 1) = Pr(X_i = -0) = \frac{1}{2}$, $\mathbb{E}[X] = \mu = \frac{n}{2}$ then*

- $Pr[X \geq \mu + a] \leq e^{\frac{-2a^2}{n}}$, *with* $a > 0$
- $Pr[X \leq \mu - a] \leq e^{\frac{-2a^2}{n}}$, *with* $0 < a < \mu$.

2.4.2 Hoeffding bounds

Theorem 2.4.5 (Hoeffding bound)**.** *[16] Let X be a sum of m independent random variables $\{X_i\}$, with $\mathbb{E}[X_i] = p$ and $X_i \in [0,1]$ for all $i \leq m$. Let μ denote the expected value of $\frac{X}{m}$. For all t: $0 \leq t \leq 1 - \mu$, then:*

$$Pr[\frac{X}{m} - \mathbb{E}(\frac{X}{m}) \geq t] \leq e^{-2t^2 m}.$$

This last bound is the one we use.

2.4.3 Number of samples

The goal of sampling is to find a close answer. But how large does the number of samples has to be so that we obtain a answer with a good accuracy ε and a hight confidence $1 - \delta$? In this section, by using Hoeffding's bound, we show the constraint for the number of samples to satisfy the property of a (ε, δ) approximate algorithm.

Theorem 2.4.6. *Let X be a sum of m independent random variables $\{X_i\}$, with $\mathbb{E}[X_i] = p_i$ and $X_i \in \{0,1\}$ for all $i \leq m$. For all ε, δ: $0 \leq \varepsilon, \delta \leq 1$, if the number of samples m we use satisfy*

$$m \geq \frac{1}{2}.(\frac{1}{\varepsilon})^2.\log\frac{1}{\delta},$$

then $$Pr[\frac{X}{m} - \mathbb{E}(\frac{X}{m}) \geq \varepsilon] \leq \delta.$$

Proof. With the above hypothesis, we can apply Hoeffding's bound for $\frac{X}{m}$. Let μ denote the expected value of $\frac{X}{m}$. For all t: $0 \leq t \leq 1 - \mu$, we have

$$Pr[\frac{X}{m} - \mathbb{E}(\frac{X}{m}) \geq t] \leq e^{-2t^2 m}$$

As we want the accuracy ε, we need to get $t = \varepsilon$. Moreover, if we want the confidence $1 - \delta$ in the estimate, we would like the right-hand side in the above to be at most δ. Therefore we have,

$$\delta \geq e^{-2t^2 m} \Leftrightarrow \delta \geq e^{-2\varepsilon^2 m} \Leftrightarrow 2\varepsilon^2 m \geq \log \frac{1}{\delta} \Leftrightarrow m \geq \frac{1}{2}.(\frac{1}{\varepsilon})^2.\log \frac{1}{\delta} \qquad \square$$

2.5 Contexts

We study the approximation in different contexts: data exchange and streaming data. In the next subsections, we present each context.

2.5.1 Data exchange

A data exchange context captures the situation when a source I exchanges data with a target J [15]. A setting is a triple (S, Σ, T) where S is a source schema, T a target schema and Σ a set of dependencies between the source and the target. A tuple dependency states that if a tuple t is in a relation R of the S, it is also in a relation R' of T, maybe slightly modified. In [13], an approximate version of the data exchange problems is introduced in order to cope with errors in the source and in the target. In [14], the data exchange setting is generalized to probabilistic sources and targets. Standard problems are: *Consistency* i.e. whether there is a solution J which satisfies the setting (S, Σ, T) for a given source I, *Typechecking* and *Query-answering*.

In this thesis, we study approximate query answering in the context of an OLAP target data warehouse with several sources (Figure 2.3). We consider the simplest situation where the tuples of each source are copied in the target: in this case the tuple dependency Σ states this constraint. In a more general situation, different sources may follow different statistical models and the main difficulty is to combine them for query answering. The approximation in the OLAP data exchange will be discussed in Chapter 6.

2.5.2 Streaming

We consider tuples of a data warehouse which arrive as a stream and study if it is possible to approximately answer OLAP queries with a small memory. If there are N tuples, an $O((\log N)^k)$ memory is considered as small, i.e. polylogarithmic. In the classical literature [24], there are many examples for upper and lower bounds. A standard example [3] is the stream of values $a_1, a_2, ..., a_n \in [1, ..., m]$ where n is unknown and n, m are arbitrarily large.

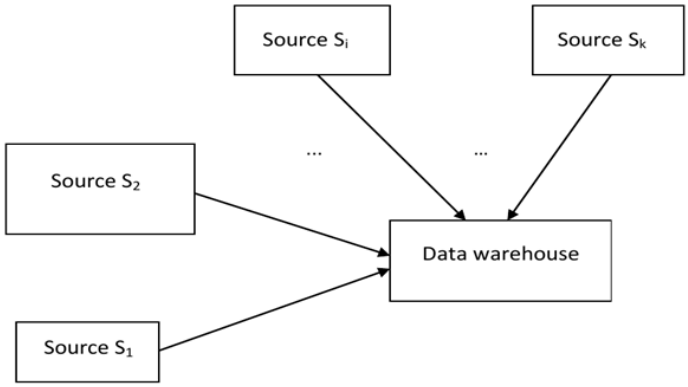


Figure 2.3: OLAP data exchange

The Frequencies $f_j = |\{i \in [1;n], a_i = j\}|, j \in [1;m]$ and the Moments $F_k = \sum_{j=1}^{m} (f_j)^k$. The Moments $F_0 = |\{j \in [1;m], f_j \neq 0\}|$, $F_1 = n$ and F_2 are approximable with a small memory, whereas F^p is not approximable for $p > 2$ and in particular $F_\infty = \max_j f_j$ is not approximable. In the OLAP setting (Figure 2.4), tuples $t \in I$ are the a_i , and we only want to approximate OLAP queries. We will show a lower bound based on the non approximability of F_∞ in the general case where the domains are unbounded. We consider the bounded case, and want to approximate the OLAP queries with the smallest possible memory. Our results will be presented in chapter 7.

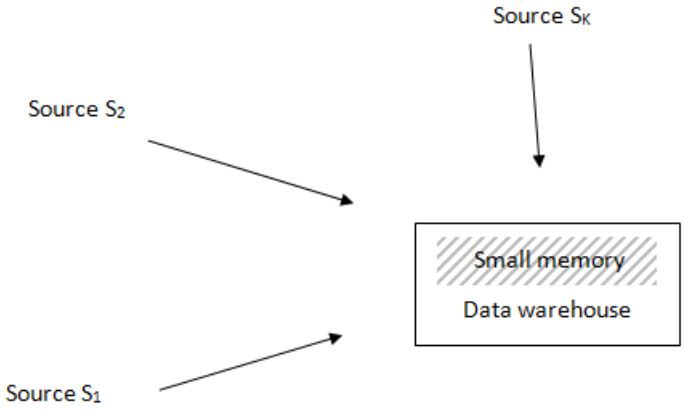


Figure 2.4: OLAP streaming

2.6 Conclusion

This chapter gives an overview of the main notions. We presented OnLine Analytical Processing, schemas, OLAP queries, query answering. We explained the probability inequalities for the sum of bounded random variables, and the notions of approximation by sampling. The goal is to approximate answers in different contexts such as the data exchange and the streaming model. We propose different methods of approximation in the next chapters.

CHAPTER 3

Approximation by Sampling a Data Warehouse

There are many different methods to sample a data warehouse. We introduce two specific techniques: the uniform sampling and the measure-based sampling. We use two probabilistic spaces to approximate the answers to the OLAP query. These techniques define a set of samples, a much smaller data warehouse, on which we approximate OLAP queries.

Massive data can not be totally read but can be sampled. Our model tries to capture when queries on such massive data can be approximated. OLAP queries on data warehouses is an example where it is possible.

3.1 Uniform sampling

Sampling a data warehouse with a uniform distribution is a classical technique. Tuples are selected with the same probability. We select $\widehat{I}_u$, made of m distinct samples of I, with a uniform distribution on the N tuples. It is important that the number of samples is large enough but independent of N.

We consider relative *measures* as answers to OLAP queries which are vectors or probabilistic distributions. For simplicity, we write Q^I_C as the density vector for the answer to Q on dimension C and on data warehouse I. Each component of Q^I_C is written $Q^I_{C=c}$ or $Q^I_C[c]$ and is the relative density for the dimension $C = c$. In another way, we can interpret $Q^I_{C=c}$ as the probability that a random tuple t is such

that $t.C = c$.

The distance between two relative answers to an OLAP query is the L_1 distance between relative densities. Let σ, μ be two distributions of two answers. These two answers are ε-close if

$$|\sigma - \mu|_1 = \sum_v |\sigma(v) - \mu(v)| \leq \varepsilon$$

We approximate a query Q^I_C by $Q^{\widehat{I}}_C$, i.e. replace the large source I by a small $\widehat{I}$. We prove that if $m \geq \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ then the answer $Q^I_{C_1,...,C_p}$ to any OLAP query Q on dimensions $C_1, \ldots, C_p$ without selection, is ε-close to $Q^{\hat{I}}_{C_1,...,C_p}$ with probability larger than $1 - \delta$, where $|C| = |C_1| * |C_2| * ... * |C_p|$.

We will select $m = \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ tuples as follows:

Data: I
Result: $\widehat{I}$
for $i := 0$ **to** m **do**
  Select t_i in I with probability $1/\texttt{size}(I)$;
  t_i in $\hat{I}$;
end

Algorithm 1: Sampling Algorithm

Consider any OLAP query without filter condition. In case of a filter, we would apply the same procedure after the filter. For simplicity, let us suppose we have a single classifier C with values $c_1, ...c_k$. By definition

$$Q^I_{C=c_1} = \frac{\sum_{t.C=c_1} t_i.M}{\sum t_i.M}$$

The answer to the query Q^I_C is a density vector, whose component on the value c_1, i.e. $Q^I_C[c_1]$ is also written $Q^I_{C=c_1}$. It is by definition the sum of the *measures* $t_i.M$ for all tuples such that $t.C = c_1$, divided by the sum of the *measures* of all tuples. We first want to show that $Pr[\mid Q^I_{C=c_1} - Q^{\hat{I}}_{C=c_1} \mid \leq \frac{\varepsilon}{|C|}] \geq 1 - \delta$. and similarly for $c_2, ..., c_k$. We can then conclude that:

$$Pr[\mid Q^I_C - Q^{\hat{I}}_C \mid \leq \varepsilon] \geq 1 - \delta.$$

3.1.1 Close answers for each component

Let X_i, Y_i be random variables associated with the sampling algorithm such that $X_i = t.M$ if the i-th random tuple t_i of $\widehat{I}$ is such that $t_i.C = c_1$, and $X_i = 0$ otherwise. Let $Y_i = t.M$. Let $X' = \sum_{i=1,..m} X_i$ and $Y' = \sum_{i=1,..m} Y_i$. Equivalently,

$$Q^{\widehat{I}}_{C=c_1} = X'/Y'.$$

Lemma 3.1.1. *For $i = 1..m$, $\mathbb{E}(X_i) = Q^I_{C=c_1}.\sum t.M/N$*

Proof. In the data warehouse I with N tuples, let n_i be the number of tuples where $t.M = i$ and $n_{i,c}$ be the number of tuples where $t.M = i$ and $t.C = c$, for $i = 0, 1, ...max$.

By definition, $\mathbb{E}(X_i) = \sum_i i.n_{i,c}/N$ as each tuple t_i has $X_i = t.M = i$ with probability $n_{i,c}/N$. The numerator is also $Q^I_{C=c_1}.\sum t.M$. Hence,

$$\mathbb{E}(X_i) = Q^I_{C=c_1}.\sum t.M/N.$$

□

Lemma 3.1.2. $\mathbb{E}(Y') = m.\sum t.M/N$

Proof. By definition, $\mathbb{E}(Y_i) = \sum_i i.n_i/N$ as each tuple t_i has $Y_i = t.M = i$ with probability n_i/N. The numerator is also $\sum t.M$ Hence, $\mathbb{E}(Y_i) = \sum t.M/N$ and by linearity $\mathbb{E}(Y') = m.\sum t.M/N$ □

All $X_1, .., X_m$ are independent with the same expectation $\mathbb{E}(X_i) = Q^I_{C=c_1}.\sum t.M/N$. By linearity $\mathbb{E}(\frac{X_1 + .. + X_m}{Y'}) = \mathbb{E}(\frac{X'}{Y'}) = \frac{m.\mathbb{E}(X_i)}{\mathbb{E}(Y')} = Q^I_{C=c_1}$.

Chernoff bounds give upper bounds for the probability that the sum of independent variables vary from their expectation by some ε. When the variables are bounded, we can use Hoeffding's bound [16] which deals with an additive error:

$$Pr[\frac{X'}{Y'} - \mathbb{E}(\frac{X'}{Y'}) \geq t] \leq e^{-2t^2.m}$$

In this form, t is the absolute error, and $e^{-2t^2.m}$ is also called the confidence δ. In our case, we estimate the answer as a distribution. The components of this distribution have very small values. So, we choose the additive form of Chernoff bounds to verify the distance between the close answer and the exact answer.

Theorem 3.1.1. *If* $m \geq \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ *then*

$$Pr[\mid Q^I_{C=c_1} - Q^{\hat{I}}_{C=c_1} \mid \geq \frac{\varepsilon}{|C|}] \leq \delta.$$

Proof. Because all X_i are independent, bounded and have the same expectations, we can use the previous Hoeffding's bound with $t = \frac{\varepsilon}{|C|}$.

Because $I\!E(\frac{X'}{Y'}) = Q^I_{C=c_1}$, then

$$Pr[\mid Q^I_{C=c_1} - Q^{\hat{I}}_{C=c_1} \mid \geq \frac{\varepsilon}{|C|}] \leq e^{-2(\frac{\varepsilon}{|C|})^2.m}$$

If $m = \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$, then

$$Pr[\mid Q^I_{C=c_1} - Q^{\hat{I}}_{C=c_1} \mid \geq \frac{\varepsilon}{|C|}] \leq \delta$$ □

.

3.1.2 Close answers of an OLAP query

We can apply the theorem 3.1.1 on each of the coordinates:

Theorem 3.1.2. *If* $m \geq \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ *then the answer* $Q^I_{C_1,\ldots,C_p}$ *to any query* Q *on dimensions* $C_1,\ldots,C_p$, *is* ε*-close to* $Q^{\hat{I}}_{C_1,\ldots,C_p}$.

Proof. In the case of a single dimension C, we apply the previous theorem for all values c_i,

$$Pr[\mid Q^I_{C=c_i} - Q^{\hat{I}}_{C=c_i} \mid \geq \frac{\varepsilon}{|C|}] \leq \delta$$

and with a union bound for $c_1, ..., c_k \in C$, we can conclude that

$$Pr[\mid Q^I_C - Q^{\hat{I}}_C \mid \geq \varepsilon] \leq \delta$$ □

Notice that this theorem gives the number of samples m necessary to guarantee the approximation, and it is independent of N.

3.2 Measure-based sampling

This sampling technique is different from the uniform sampling. It samples the data warehouse to get m distinct tuples, but the probability of selection of each tuple depends on its *measure*. We select $\widehat{I_M}$ made of m distinct samples in two phases:

1. We first select a tuple t with a uniform distribution
2. We then keep t with probability which is proportional to its *measure*, $t.M/max$ where max is the maximum value of the *measure*. We then replace the value of the *measure* by 1 if we keep the sample.

We approximate a query Q_C^I by $Q_C^{\widehat{I}}$, i.e. replace the large source I by a small $\widehat{I}$. We prove that if $m \geq \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ then the answer $Q^I_{C_1,\ldots,C_p}$ to any OLAP query Q on dimensions $C_1,\ldots,C_p$ without selection, is ε-close to $Q^{\hat{I}}_{C_1,\ldots,C_p}$ with probability is larger than $1-\delta$, where $|C| = |C_1| * |C_2| * ... * |C_p|$.

We will select $m = \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ tuples as follows:

Data: I
Result: $\widehat{I}$
for $i := 0$ **to** m **do**
  Select t_i in I with probability $1/\texttt{size}(I)$;
  Select k in $[1, max]$ with uniform distribution ;
  if $k \leq t_i.M$ **then**
    $t_i.M := 1$;
    Add t_i in $\hat{I}$;
  end
end

Algorithm 2: Measure-based Sampling Algorithm

Consider any OLAP query without filter condition. In case of a filter, we would apply the same procedure after the filter. For simplicity, let us suppose we have a single classifier C with values $c_1, ...c_k$. We first want to show that

$$Pr[|\ Q^I_{C=c_1} - Q^{\hat{I}}_{C=c_1}\ | \leq \frac{\varepsilon}{|C|}] \geq 1-\delta$$

and similarly for $c_2, ...c_k$. We can then conclude that:

$$Pr[\mid Q_C^I - Q_C^{\hat{I}} \mid \leq \varepsilon] \geq 1 - \delta.$$

3.2.1 Close answers for each component

Let X_i be the random variable associated with the sampling algorithm such that $X_i = 1$ if the i-th tuple of $\widehat{I}$ is such that $t_i.C = c_1$, and $X_i = 0$ otherwise. Let $I\!E(X_i)$ be the expectation of X_i.

Let $X' = \sum_{i=1,..m} X_i$. Equivalently,

$$Q_{C=c_1}^{\widehat{I}} = X'/m.$$

Lemma 3.2.1. *For $i = 1..m$, $I\!E(X_i) = Q_{C=c_1}^I$.*

Proof. For each tuple t_i in I, $i = 1..N$, the probability to randomly choose t_i is $1/N$. The probability p_i that t_i is chosen and kept in $\widehat{I}$ is:

$$p_i = (1/N).(t_i.M/max).$$

The probability p' to choose one tuple in $\widehat{I}$ is the sum of the p_i, i.e.

$$p' = (1/N.max).\Sigma(t_i.M).$$

The probability to choose one tuple in $\widehat{I}$ such that $t.C = c_1$ is:

$$p'' = (1/N.max).\Sigma_{t.C=c_1}(t_i.M).$$

Hence

$$I\!E(X_i) = P(X_i = 1) = p''/p' = \frac{\Sigma_{t.C=c_1}(t_i.M)}{\Sigma(t_i.M)} = Q_{C=c_1}^I. \qquad \square$$

All $X_1, .., X_m$ are independent with the same expectation $I\!E(X_i) = Q_{C=c_1}^I$. By linearity $I\!E(\frac{X_1 + .. + X_m}{m}) = I\!E(\frac{X'}{m}) = Q_{C=c_1}^I$.

The variables are bounded, so we can use Hoeffding's bound [16] again:

$$Pr[\frac{X'}{m} - I\!E(\frac{X'}{m}) \geq t] \leq e^{-2t^2.m}$$

.

Theorem 3.2.1. *If* $m \geq \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ *then*

$$Pr[|\ Q^I_{C=c_1} - Q^{\hat{I}}_{C=c_1}\ | \geq \frac{\varepsilon}{|C|}] \leq \delta.$$

Proof. Because all X_i are independent, bounded and have the same expectations, we can use the previous Hoeffding's bound with $t = \frac{\varepsilon}{|C|}$. Because $I\!E(\frac{X'}{m}) = Q^I_{C=c_1}$, then

$$Pr[|\ Q^I_{C=c_1} - Q^{\hat{I}}_{C=c_1}\ | \geq \frac{\varepsilon}{|C|}] \leq e^{-2(\frac{\varepsilon}{|C|})^2.m}$$

If $m = \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$, then

$$Pr[|\ Q^I_{C=c_1} - Q^{\hat{I}}_{C=c_1}\ | \geq \frac{\varepsilon}{|C|}] \leq \delta.$$ □

3.2.2 Close answers of an OLAP query

We can apply the theorem 3.2.1 on each of the coordinates:

Theorem 3.2.2. *If* $m \geq \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ *then the answer* $Q^I_{C_1,\ldots,C_p}$ *to any query* Q *on dimensions* $C_1,\ldots,C_p$*, is* ε*-close to* $Q^{\hat{I}}_{C_1,\ldots,C_p}$.

Proof. In the case of a single dimension C, we apply the theorem 3.2.1,

$$Pr[|\ Q^I_{C=c_1} - Q^{\hat{I}}_{C=c_1}\ | \geq \frac{\varepsilon}{|C|}] \leq \delta$$

With a union bound for $c_1, ..., c_k \in C$, we can conclude that

$$Pr[|\ Q^I_C - Q^{\hat{I}}_C\ | \geq \varepsilon] \leq \delta$$ □

3.3 Important comparisons

These two methods produce distinct random data warehouses $\widehat{I}$. We approximate a query Q^I_C by $Q^{\widehat{I}}_C$, i.e. replacing the large source I by a small $\widehat{I}$. If max is small in

relation with N, both techniques allow to approximate OLAP queries if the number of samples is large enough, but independent of N.

We proved that if $m \geq \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ then the answer $Q^I_{C_1,\ldots,C_p}$ to any OLAP query Q on dimensions $C_1,\ldots,C_p$ without selection, is ε-close to $Q^{\hat{I}}_{C_1,\ldots,C_p}$ with probability is larger than $1-\delta$, where $|C| = |C_1| * |C_2| * ... * |C_p|$.

This is an application of a Chernoff-Hoeffding bound to estimate the error on each density, and a union bound on the global vectors. It is important that the number of tuples is large, and possible selections may alter the result. In case of a selection, the number of tuples after the selection must be large to apply for the result.

Notice that if *max* is large (unbounded), these two distributions differ, and only the second one can approximate OLAP queries in our sense. Another important difference is when we have several *measures* (2 in our example): one set of samples is enough for the uniform distribution, but we need two sets for the *measure*-based distribution, one for each *measure*.

The specific L_1 distance between answers is important. As noticed in [1], the uniform sampling may miss small groups of data, and the relative error for such a group could be 100%. As the *measures* are bounded, the total value would be small and our relative error would then be less than ε. In case of a selection, if the number of selected tuples is too small, the error could be large for the same reason. If the number of selected tuples is large, the same approximation holds. We give the

	Sampling by uniform distribution	Measure- based sampling
Number of samples	$m = \frac{1}{2}.(\frac{\|C\|}{\varepsilon})^2.\log\frac{1}{\delta}$	$m = \frac{1}{2}.(\frac{\|C\|}{\varepsilon})^2.\log\frac{1}{\delta}$
Number of set of samples	1	One set of samples for each *measures*
Probability that one tuple is chosen and kept in the set of samples	That does not depend on its *measures*	That depends on its *measures*
Situation for applying	Small *max*	Small and large *max*

Table 3.1: Two sampling methods.

summary of the two methods in Table 3.1

3.4 Conclusion

We presented two algorithms for estimating the answers to OLAP queries. The first is the uniform sampling. The second is the measured-based sampling. Chernoff-Hoeffding bounds were applied to guarantee the quality of the approximation. We proved that these two algorithms can approximate the answers with good accuracy ε and with high probability $1-\delta$. The comparison between two algorithms is discussed. The chapter 9 will give us the experimental results on a data warehouse collecting sensors values and on a data warehouse collecting RSS data.

CHAPTER 4

Edit Distance on Data Warehouses

4.1 Introduction

We introduce an edit distance on data warehouses, i.e. on tables where one column, the *measure*, plays a specific role. We introduce the M-Edit distance which adapts the classical edit distances on words and trees to relations with a specific *measure* attribute. Edit operations on the standard attributes have a unit cost, but edit operations on the *measure* attribute have a cost proportional to the variation of the *measure*.

4.2 Edit operations and cost functions

There are three types of edit operations: tuple deletion, tuple insertion and tuple modification. The cost function w assigns to each operation a positive real number to denote its cost. Let $t.M$ be the value of the *measure* M of tuple t, which we assume bounded by a maximum fixed value max. Let d be the number of dimensions of the relation I, and $A_1, A_2, ..., A_d, M$ be the sequence of attributes. The (absolute) cost for each edit operation is defined as:

- (Tuple deletion) The cost to delete a tuple t in the relation I is equal to its *measure* plus the number of dimensions:

$$w_d = t.M + d$$

- (Tuple insertion) The cost to insert a tuple t in the relation I is equal to its *measure* plus the number of dimensions:

$$w_i = t.M + d$$

- (Tuple modification) A modification may concern several attributes, including the *measure*.

$$w_m = \Sigma_{\text{modified} \quad \text{attributes}} w(A_i)$$

  If $A_i = M$, i.e. is the *measure*: $w(A_i) =\mid t.M - t'.M \mid$ where $t.M$ and $t'.M$ are the old *measure* and the new *measure* of tuple t.

  If A_i is an attribute (string, integer, real,...): $w(A_i) = 1$.

A sequence of edit operations is abbreviated: $i_1, m_2, i_3, d_4, ...$ if the first edit is an *insertion*, the second a *modification*, the third a *insertion*, the fourth a *deletion* and so on.

If we assume that all modifications apply to distinct tuples, we can then do all the insertion first, then all the modifications and then all the deletions. The change of its position in the sequence does not influence its cost.

Let $S = (\bar{i}, \bar{m}, \bar{d})$ be a normalized sequence of insertions, modifications and deletions. The absolute cost of S is defined by formula: $(w(\bar{i}) + w(\bar{m}) + w(\bar{d}))$ where $w(\bar{i}) = \sum_{i_j \in \bar{i}} w_i(i_j)$ and similarly for $w(\bar{m})$ and $w(\bar{d})$.

We normalize the cost to obtain a relative cost between 0 and 1. Let s be the number of inserted tuples in the transformation S plus the size of I before the transformation, i.e. the maximum size along the transformation. Let $Max(t.M)$ be the maximum value of the *measure* M of a tuple t along the transformation. If no modification is applied to t, then $Max(t.M) = t.M$.

The relative cost of S, $w(S)$ is defined by:

$$w(S) = \frac{(w(\bar{i}) + w(\bar{m}) + w(\bar{d}))}{(d.s + \Sigma_t \ Max(t.M))}$$

4.3 Distance between two data warehouses

Definition 4.3.1 (M-Edit Distance)**.** *The distance $d(I, I')$ between two data warehouses I and I' on the same schema is the minimum relative cost over all sequences S of edit operations that transforms I into I':*

$$d(I, I') = Min_S \ \{w(S)\}$$

According the above definition, the bigger the distance is, the more different two relations are. If $d(I, I') = 0$, I and I' are identical.

4.4 Distance between two answers

A typical OLAP query may select *COUNTRY* as dimension and an absolute answer would be: (France, 600), (Germany, 400). We consider the *answer to any OLAP Query* as the vector of relative values. In our example, $Q^I_{Country} = (0.6, 0.4)$ and $Q^I_{Country=Germany} = 0.4$.

Definition 4.4.1. *The distance between two relative answers to an OLAP query is the L_1 distance between relative densities.*

Therefore, the values of the distance are in $[0, 2]$. For example, the distance between the $(0.65, 0.2, 0.15)$ distribution over (France, Germany, U.K.) and the $(0.6, 0.4)$ distribution over (France, Germany) is $(0.05 + 0.2 + 0.15) = 0.4$.

4.5 Properties of distance between two data warehouses

We show that the M-Edit distance is a metric, i.e. it is always in $[0, 1]$, symmetric and has the triangular inequality property.

Lemma 4.5.1. *Let I and I' be two data warehouses. The distance between I and I' is always such that $0 \leq d(I, I') \leq 1$.*

Proof. If I is identical to I',then $d(I, I') = 0$. If I is different from I', then let us show that $0 < (w(\bar{i}) + w(\bar{m}) + w(\bar{d})) \leq (d.s + \Sigma_t \ Max(t.M))$ for the minimum cost sequence $S = (\bar{i}, \bar{m}, \bar{d})$ which transforms I into I',

Let v the number of modified attributes and let j an index ranging on the modified tuples. Then

$$(w(\bar{i}) + w(\bar{m}) + w(\bar{d})) = \Sigma_j d + \Sigma_j v + \Sigma_j t.M + \Sigma_j(\mid t.M - t'.M \mid)$$

Moreover,

$$\Sigma_j d + \Sigma_j v \leq d.s$$

,

and

$$\Sigma_j t.M + \Sigma_j(\mid t.M - t'.M \mid) \leq \Sigma_t \ Max(t.M)$$

So,

$$0 < (w(\bar{i}) + w(\bar{m}) + w(\bar{d})) \leq (d.s + \Sigma_t \ Max(t.M))$$

hence

$$0 < d(I, I') \leq 1$$ □

Lemma 4.5.2. *The function $d(I, I')$ is symmetric.*

Proof. Let $S = (\bar{i}, \bar{m}, \bar{d})$ be the minimum cost sequence which transforms I into I'. We construct a sequence $S' = (\bar{i'}, \bar{m'}, \bar{d'})$ of edit operations which transforms I' into I with the following operations:

1. An insertion in S is replaced by the corresponding deletion in S'.
2. A deletion in S is replaced by the corresponding insertion in S'.
3. If a modification in $\bar{m}$ increases the measure, then the corresponding modification in $\bar{m'}$ decreases the measure by the same value.
4. If a modification in $\bar{m}$ modifies the values of the standard attributes, then the corresponding modification in $\bar{m'}$ gives back the old values.

After we normalize S',

$$(w(\bar{i'}) + w(\bar{m'}) + w(\bar{d'})) = (w(\bar{i}) + w(\bar{m}) + w(\bar{d}))$$

The maximum number of tuples s from I along the transformation is equal to the size of I plus the number of insertions in S.

$$s = \mid I \mid + \mid \bar{i} \mid$$

The maximum number of tuples s' from I' along the transformation is equal to the size of I' plus the number of insertions in S'.

Moreover, the number of insertions in S' is equal to the number of deletions in S.

$$\mid \bar{i'} \mid = \mid \bar{d} \mid$$

We have

$$s' = \mid I' \mid + \mid \bar{i'} \mid = (\mid I \mid + \mid \bar{i} \mid - \mid \bar{d} \mid) + \mid \bar{d} \mid = \mid I \mid + \mid \bar{i} \mid$$

We see that: $s' = s$

then,

$$(d.s' + \Sigma_{t'}\ Max(t'.M)) = (d.s + \Sigma_t\ Max(t.M))$$

So,

$$d(I', I) = d(I, I') \qquad \square$$

In the lemma below, we show that function $d(I, I')$ has the property of triangular inequality.

Lemma 4.5.3. *The function $d(I, I')$ satisfies the triangular inequality.*

Proof. Let I'' be an arbitrary relation. Let $S_1 = (\bar{i_1}, \bar{m_1}, \bar{d_1})$ be the sequence has the minimum relative cost to transform I into I''. Let $S_2 = (\bar{i_2}, \bar{m_2}, \bar{d_2})$ be the sequence has the minimum relative cost to transform I'' into I'. Then, $S = (\bar{i_1}, \bar{i_2}, \bar{m_1}, \bar{m_2}, \bar{d_1}, \bar{d_2})$ is the sequence to transform I into I'. We have:

$$d(I, I'') = \frac{(w(\bar{i_1}) + w(\bar{m_1}) + w(\bar{d_1}))}{(d.(\mid I \mid + \mid \bar{i_1} \mid) + \Sigma_t\ Max(t.M))}$$

with t along the tranformation in S_1.

$$d(I'', I') = \frac{(w(\bar{i_2}) + w(\bar{m_2}) + w(\bar{d_2}))}{(d.(\mid I \mid + \mid \bar{i_1} \mid - \mid \bar{d_1} \mid + \mid \bar{i_2} \mid) + \Sigma_t\ Max(t.M))}$$

with t along the tranformation in S_2.

$$w(S) = \frac{(w(\bar{i_1}) + w(\bar{i_2}) + w(\bar{m_1}) + w(\bar{m_2}) + w(\bar{d_1}) + w(\bar{d_2}))}{(d.(\mid I \mid + \mid \bar{i_1} \mid + \mid \bar{i_2} \mid) + \Sigma_t Max(t.M))}$$

with t along the tranformation in S.

We can see that $w(S) \leq d(I, I'') + d(I'', I')$.

According the definition of distance between two relations, $d(I, I')$ is the miminum value: $d(I, I') \leq w(S)$.

Then,

$$d(I, I') \leq d(I, I'') + d(I'', I'). \qquad \square$$

4.6 Continuity theorem

Suppose that max is the maximum value of the *measure* and $\mid C \mid$ is the cardinality of dimension C.

Theorem 4.6.1. *If the distance between two data warehouses is smaller than ε then the distance between two answers on C is smaller than $\mid C \mid .max.\varepsilon$.*

Proof. The answer to the query Q^I_C for the component on the value c.

$$Q^I_{C=c} = \frac{\sum_{t.C=c} t.M}{\sum t.M}$$

Consider the worst case for a table T in Table 4.1 with N tuples. There are $\varepsilon.N$ tuples having $t.C = c$ and $t.M = max$. All $(N - \varepsilon.N)$ other tuples having $t.M = 1$. Suppose that we delete all tuples whose $t.C = c$ and $t.M = max$.

ID	B	C	M
1		c	max
...	...	...	...
$\varepsilon.N$		c	max
$\varepsilon.N + 1$		a	1
...	...	...	...
N		a	1

Table 4.1: Table T.

The answer for the query $Q^{I'}_C$ whose component on the value c.

$$Q^{I'}_{C=c} = 0$$

So, we have the distance between two answers on the value c:

$$| Q^I_{C=c} - Q^{I'}_{C=c} | = | \frac{max.\varepsilon.N}{\sum t.M} - 0 | = \frac{max.\varepsilon.N}{\sum t.M}$$

$$\Rightarrow | Q^I_{C=c} - Q^{I'}_{C=c} | = \frac{max.\varepsilon.N}{(N - \varepsilon.N).1 + \varepsilon.N.max}$$

$$\Rightarrow | Q^I_{C=c} - Q^{I'}_{C=c} | = \frac{max.\varepsilon}{1 - \varepsilon + \varepsilon.max}$$

$$\Rightarrow | Q^I_{C=c} - Q^{I'}_{C=c} | \leq max.\varepsilon$$

With a union bound for $c_1, ..., c_k \in C$, the distance between two answers:

$$\Rightarrow | Q^I_C - Q^{I'}_C | \leq | C | .max.\varepsilon$$

□

The bound on the error is only meaningful for small values of ε, i.e. $\varepsilon < |C|.max$

4.7 Conclusion

We studies the edit distance for data warehouses. We describe the edit operations: tuples deletion, tuple insertion and tuple modification. Moreover, we define the distance betweens two data warehouses. We prove that this edit distance is a metric, i.e. it is always in $[0, 1]$, symmetric and has the triangular inequality property. We then prove the continuity theorem.

The edit distance we introduced is adapted to data warehouses, as it guarantees that close data warehouses imply that the answers to OLAP queries must also be close.

CHAPTER 5

Statistical Model

In the relational model, classical dependencies, such as *Functional dependencie* play an important role. For data warehouses, there are some other important dependencies, in particular *statistical dependencies*, which we introduce. In this case, some attributes imply fixed distributions of the measure, and we can then approximate a data warehouse by some fixed set of distributions.

In this context, we study another method of approximation for query answering. It is the statistical model. In our model [26], a set of attributes determines **f**ixed distributions of the *measure* M with high probability. In this section, the notion of statistical dependency is presented. Then, we prove that the OLAP query answering can be obtained from the statistical model. Finally, we show how to learn the distributions of statistical dependencies of the model.

5.1 Statistical dependencies

Assume each set of attributes ranges over a finite domain D and let Δ be the set of distributions over D. We first define the notion of a *distribution for a set of attributes* $A_1..A_k$ and then the notion of a *statistical dependency*, which generalizes the classical functional dependency. For simplicity, we assume only one attribute A which follows a distribution σ, and we note $\sigma(a)$ the probability that $A = a$. We define $A \triangleleft M$ if each value $a \in A$ implies a fixed distribution μ_a over the values of M.

Definition 5.1.1. *(Distribution for a set of attributes) A set of attributes $A_1, .., A_k$ follows a distribution $\sigma(A_1, .., A_k)$ on the data warehouse I of size N of an OLAP*

schema, if for all $a_1 \in A_1, .., a_k \in A_k$,

$$Prob_{t \in_r I}[t.A_1 = a_1, .., t.A_k = a_k] = \sigma_{A_1,..,A_k}(a_1, .., a_k)$$

The notation $t \in_r I$ means that the tuple t is taken with the uniform distribution on I.

Let a family of data warehouse $\mathcal{I} = \{I_1, .., I_N ...\}$ of increasing sizes N of an OLAP schema. A statistical dependency applies to such a family as the distributions are limit objects, as for graph limits [22].

Definition 5.1.2. *(Statistical dependency) The attribute* A *over values* $\{a_1, ..., a_p\}$ statistically implies M *on* $\mathcal{I}$*, written* $A \triangleleft M$*, if there are fixed distributions* $\mu_{a_1}, \mu_{a_2}, ..., \mu_{a_p}$ *over the domain of* M*, if for all* $i = 1, .., p$,

$$\lim_{N \to \infty} Prob_{t \in_r I}[t.M = b / t.A = a_i] = \mu_{a_i}(b)$$

Given some statistical dependencies $A \triangleleft M$, we map a data warehouse I to the statistics σ_A, and σ_{A,C_i} for all the dimensions C_i.

Definition 5.1.3. *(Statistical model) Assume that* $A \triangleleft M$*, a* statistical model *of a data warehouse* I *on a schema* T *is the sequence of distributions* $\sigma_A, \sigma_{A,C_i}, \sigma_{C_i,C_j}$ *for the dimensions* C_i.

5.2 Relation between the statistical model and the query answering

We use the statistical model to approximate the answers to OLAP queries. We show that for a *measure* M, if there is an attribute A such that $A \triangleleft M$, then the distribution over $C.A$ is enough to approximate an OLAP query over the dimension C.

The advantage of this technique is that the compact structure of the statistical model makes analysis simpler. Let S be the statistical model for the data warehouse I. Let Q be the answer and $\hat{Q}$ be the approximate answer. Figure 5.1 describes the setting. We can approximate a data warehouse by some fixed set of distributions. Moreover, we can have the close answer by the statistical model.

5.3 Approximation by the statistical model

5.3.1 Analysis of the *measure* on one dimension

Let M be the *measure* in the model, where we approximate all the distributions C_i, A for the all the dimensions C_i of the OLAP schema. These distributions require

$$\begin{array}{ccc} I & \approx & S \\ \downarrow & & \downarrow \\ Q & \approx & \hat{Q} \end{array}$$

Figure 5.1: Relation between statistical model and query answering.

$O(\sum_i |C_i| * |A| + |M| * |A|)$ space, i.e. independent of N.

For each distribution μ_{a_i}, let $Avg_\mu(a_i)$ be the average value of *measure* with the distribution $\mu(a_i)$. Let $\sigma_{(A,C)}(a_i, c_j)$ be the probability such that $A = a_i$ and $C = c_j$. We will estimate $Q^I_{C=c}$ by

$$Q^S_{C=c_j} = \frac{\sum_{a_i} \sigma_{(A,C)}(a_i, c_j) * Avg_\mu(a_i)}{\sum_{c_j} \sum_{a_i} \sigma_{C=c_i}(C.A)(a_i) * Avg_\mu(a_i)}$$

and define the approximate answer Q^S_C .

We fist prove the correct answer for $Q^I_{A=a_i}$ when $C = A$.

5.3.1.1 Analysis of the *measure* on A

Theorem 5.3.1. *Let S be the statistical model with the sequence of distributions $\sigma_A, \sigma_{A,C_i}, \sigma_{C_i,C_j}$ for the dimensions C_i. Then, $Q^S_{A=a_i}$ is equal to $Q^I_{A=a_i}$ where*

$$Q^S_{A=a_i} = \frac{\sigma_{(A)}(a_i) * Avg_\mu(a_i)}{\sum_{a_i} \sigma_{(A)}(a_i) * Avg_\mu(a_i)}.$$

Proof. Assume that N is the number of tuples of the data warehouse. $N_{A=a_i}$ is the number of tuples that $A = a_i$. Let $\sigma_{(A)}(a_i)$ be the probability that $A = a_i$.

$$\sigma_{(A)}(a_i) = \frac{N_{A=a_i}}{N} \tag{5.1}$$

For each distribution μ_{a_i}, let $Avg_\mu(a_i)$ be the average value of *measure* with the distribution $\mu(a_i)$.

$$Avg_\mu(a_i) = \frac{\sum_{A=a_i} t.M}{N_{A=a_i}} \tag{5.2}$$

From (5.1) and (5.2), we have:

$$
\begin{aligned}
Q^S_{A=a_i} &= \frac{\sigma_{(A)}(a_i) * Avg_\mu(a_i)}{\sum_{a_i} \sigma_{(A)}(a_i) * Avg_\mu(a_i)} \\
&= \frac{\frac{N_{A=a_i}}{N} * \frac{\sum_{A=a_i} t.M}{N_{A=a_i}}}{\sum_{a_i} \frac{N_{A=a_i}}{N} * \frac{\sum_{A=a_i} t.M}{N_{A=a_i}}} \\
&= \frac{\frac{1}{N} * \sum_{A=a_i} t.M}{\sum_{a_i} \frac{1}{N} * \sum_{A=a_i} t.M} \\
&= \frac{\sum_{A=a_i} t.M}{\sum_{a_i} \sum_{A=a_i} t.M} = Q^I_{A=a_i}
\end{aligned}
$$

where $t.M$ is the *measure* of tuples t. □

Example 5.3.1. *Consider the special case of an OLAP schema T_1 where the data warehouse I_1 (with an attribute A and a measure M) in Table 5.1 contains the tuples:*

A	M
a_1	10
a_1	15
a_2	10
a_2	20

Table 5.1: Data warehouse I_1.

We have $\mu_{a_1}(10) = \frac{1}{2}$, $\mu_{a_1}(15) = \frac{1}{2}$ and $\mu_{a_1}(20) = 0 \rightarrow Avg_\mu(a_1) = \frac{25}{2}$

$\mu_{a_2}(10) = \frac{1}{2}$, $\mu_{a_2}(15) = 0$ and $\mu_{a_2}(20) = \frac{1}{2} \rightarrow Avg_\mu(a_2) = \frac{1}{2} * 10 + \frac{1}{2} * 20 = 15$

Moreover, $\sigma_{(A)}(a_1) = \frac{1}{2}$, $\sigma_{(A)}(a_2) = \frac{1}{2}$.

Hence, we have

$$
Q^I_{A=a_1} = Q^S_{A=a_1} = \frac{\sigma_{(A)}(a_1) * Avg_\mu(a_1)}{\sum_{a_i} \sigma_{(A)}(a_i) * Avg_\mu(a_i)} = \frac{\frac{1}{2} * \frac{25}{2}}{\frac{1}{2} * \frac{25}{2} + \frac{1}{2} * 15} = \frac{5}{11}
$$

$$Q^I_{A=a_2} = Q^S_{A=a_2} = \frac{\sigma_{(A)}(a_2) * Avg_\mu(a_2)}{\sum_{a_i} \sigma_{(A)}(a_i) * Avg_\mu(a_i)} = \frac{\frac{1}{2} * 15}{\frac{1}{2} * \frac{25}{2} + \frac{1}{2} * 15} = \frac{6}{11}$$

5.3.1.2 Analysis of the *measure* on C

We distinguish two cases. The first case is the analysis on C when C and A are independent. The second case is the analysis on C when C and A are dependent.

5.3.1.3 C and A are two independent attributes for M

We recall the the property of the conditional probability of two independent variables.

Definition 5.3.1. *(Independent variables) Let A and C two variables. If they are independent then:*

$$Pr[A] = Pr[A/C]$$

Definition 5.3.2. *(Two independent attributes) Let A and C be two attributes. C and A are independent for M if for all $a_i \in A, c_j \in C$,*

$$\mu(a_i) = \mu(a_i|C = c_j)$$

Theorem 5.3.2. *Let S be the statistical model with the sequence of distributions $\sigma_A, \sigma_{A,C_i}, \sigma_{C_i,C_j}$ for the dimensions C_i. Then, $Q^S_{C=c_j}$ is equal to $Q^I_{C=c_j}$ where*

$$Q^S_{C=c_j} = \frac{\sum_{a_i} \sigma_{(C,A)}(c_j, a_i) * Avg_\mu(a_i)}{\sum_{c_j} \sum_{a_i} \sigma_{(C,A)}(c_j, a_i) * Avg_\mu(a_i)}$$

Proof. From the definition 5.3.2, we have: $\mu(a_i) = \mu(a_i|C = c_j)$

For all $b_k \in M$, let $Avg_\mu(a_i)$ be the average value of *measure* with the distribution μ_{a_i}.

$$Avg_\mu(a_i) = \sum_{b_k} \mu_{a_i}(b_k) * b_k$$

Let $Avg_\mu(a_i/C = c_j)$ be the average value of *measure* with the distribution $\mu_{(a_i/C=c_j)}$.

$$Avg_\mu(a_i/C = c_j) = \sum_{b_k} \mu_{(a_i/C=c_j)}(b_k) * b_k$$

Hence,

$$Avg_\mu(a_i) = Avg_\mu(a_i/C = c_j) \tag{5.3}$$

Assume that $N_{A=a_i,C=c_j}$ is the number of tuples such that $A = a_i$ and $C = c_j$. Let $\sigma_{(C,A)}(c_j, a_i)$ be the probability such that $A = a_i$ and $C = c_j$.

$$\sigma_{C,A}(c_j, a_i) = \frac{N_{A=a_i,C=c_j}}{N} \tag{5.4}$$

$Avg_\mu(a_i/C = c_j)$, the average value of *measure* with the distribution $\mu(a_i/C = c_i)$ can be calculated as follows:

$$Avg_\mu(a_i/C = c_j) = \frac{\sum_{A=a_i,C=c_j} t.M}{N_{A=a_i,C=c_j}} \tag{5.5}$$

So, we have:

$$\begin{aligned}
Q^S_{C=c_j} &= \frac{\sum_{a_i} \sigma_{C,A}(c_j, a_i) * Avg_\mu(a_i)}{\sum_{c_j}\sum_{a_i} \sigma_{(C,A)}(c_j, a_i) * Avg_\mu(a_i)} \\
&= \frac{\sum_{a_i} \sigma_{(C,A)}(c_j, a_i) * Avg_\mu(a_i/C = c_j)}{\sum_{c_j}\sum_{a_i} \sigma_{(C,A)}(c_j, a_i) * Avg_\mu(a_i/C = c_j)} && \text{(From (5.3))} \\
&= \frac{\sum_{a_i} (\frac{N_{A=a_i,C=c_j}}{N} * \frac{\sum_{A=a_i,C=c_j} t.M}{N_{A=a_i,C=c_j}})}{\sum_{c_j}\sum_{a_i} (\frac{N_{A=a_i,C=c_j}}{N} * \frac{\sum_{A=a_i,C=c_j} t.M}{N_{A=a_i,C=c_j}})} && \text{(From (5.4) and (5.5))} \\
&= \frac{\sum_{a_i} (\frac{1}{N} * \sum_{A=a_i,C=c_j} t.M)}{\sum_{c_j}\sum_{a_i} (\frac{1}{N} * \sum_{A=a_i,C=c_j} t.M)} \\
&= \frac{\sum_{a_i} (\sum_{A=a_i,C=c_j} t.M)}{\sum_{c_j}\sum_{a_i} (\sum_{A=a_i,C=c_j} t.M)} \\
&= \frac{\sum_{C=c_j} t.M}{\sum t.M} = Q^I_{C=c_j}
\end{aligned}$$

where $t.M$ is the *measure* of tuples t. □

Example 5.3.2. *Consider the special case of an OLAP schema T_2 where the data warehouse I_2 with three attributes in Table 5.2 contains the tuples:*

C_1	A	M
d	a_1	10
c	a_1	10
c	a_2	10
d	a_2	20
d	a_2	10
c	a_2	20

Table 5.2: Data warehouse I_2.

We have $\mu_{a_1}(10) = 1$ and $\mu_{a_1}(20) = 0 \rightarrow Avg_\mu(a_1) = 10$

$\mu_{a_1/C=c}(10) = 1$ and $\mu_{a_1/C=c}(20) = 0 \rightarrow Avg_\mu(a_1/C = c) = 10$

$\mu_{a_1/C=d}(10) = 1$ and $\mu_{a_1/C=d}(20) = 0 \rightarrow Avg_\mu(a_1/C = d) = 10$

$\mu_{a_2}(10) = \frac{1}{2}$ and $\mu_{a_2}(20) = \frac{1}{2} \rightarrow Avg_\mu(a_2) = \frac{1}{2} * 10 + \frac{1}{2} * 20 = 15$

$\mu_{a_2/C=c}(10) = \frac{1}{2}$ and $\mu_{a_2/C=c}(20) = \frac{1}{2} \rightarrow Avg_\mu(a_2/C = c) = \frac{1}{2} * 10 + \frac{1}{2} * 20 = 15$

$\mu_{a_2/C=d}(10) = \frac{1}{2}$ and $\mu_{a_2/C=d}(20) = \frac{1}{2} \rightarrow Avg_\mu(a_2/C = d) = \frac{1}{2} * 10 + \frac{1}{2} * 20 = 15$

Moreover,

$$\sigma_{C,A}(c, a_1) = \frac{1}{6},\ \sigma_{C,A}(c, a_2) = \frac{2}{6},\ \sigma_{C,A}(d, a_1) = \frac{1}{6},\ \text{and } \sigma_{C,A}(d, a_2) = \frac{2}{6}$$

.

Hence, we have

$$Q^I_{C=c} = Q^S_{C=c} = \frac{\sigma_{C,A}(c, a_1) * Avg_\mu(a_1) + \sigma_{C,A}(c, a_2) * Avg_\mu(a_2)}{\sum_{a_i} \sigma_{C,A}(c, a_i) * Avg_\mu(a_i) + \sum_{a_i} \sigma_{C,A}(d, a_i) * Avg_\mu(a_i)} = \frac{1}{2}$$

$$Q^I_{C=d} = Q^S_{C=d} = \frac{\sigma_{C,A}(d, a_1) * Avg_\mu(a_1) + \sigma_{C,A}(d, a_2) * Avg_\mu(a_2)}{\sum_{a_i} \sigma_{C,A}(c, a_i) * Avg_\mu(a_i) + \sum_{a_i} \sigma_{C,A}(d, a_i) * Avg_\mu(a_i)} = \frac{1}{2}$$

5.3.1.4 C and A are two dependent attributes for M

In the case that C and A are two dependent attributes for M. We first calculate the distributions of *measure* on values c_j of C: μ_{c_j}. Then, for each $c_j \in C$, we calculate

the distributions of tuples on C: $\sigma_C(c_i)$. The analysis of *measure* on $C = c_j$ is computed as follows:

$$Q^S_{C=c_j} = \frac{\sigma_C(c_j) * Avg_\mu(c_j)}{\sum_{c_j} \sigma_C(c_j) * Avg_\mu(c_j)}$$

Definition 5.3.3. *(Set of independent attributes for M) Let $C_1, C_2, ..., C_k$ and A be attributes. $C_1, C_2, ..., C_k$ and A are independent for M, for all $a_i \in A, c_1 \in C_1, ..., c_k \in C_k$, if*

$$\mu(a_i) = \mu(a_i|C_1 = c_1, C_2 = c_2) = ... = \mu(a_i|C_1 = c_1, ..., C_k = c_k)$$

5.3.2 Analysis of the *measure* on two dimensions

We analyze in the case when C_1, C_2, A are independent for M. Moreover, C_1, C_2, A are independent variables.

Theorem 5.3.3. *Let S be the statistical model with the sequence of distributions $\sigma_A, \sigma_{A,C_i}, \sigma_{C_i,C_j}$ for the dimensions C_i. Then, $Q^S_{C_1=c_1,C_2=c_2}$ is equal to $Q^I_{C_1=c_1,C_2=c_2}$ where:*

$$Q^S_{C_1=c_1,C_2=c_2} = \frac{\sigma_{C_1,C_2}(c_1, c_2) * Q^S_{C_1=c_1}}{\sum_{c_1}\sum_{c_2} \sigma_{C_1,C_2}(c_1, c_2) * Q^S_{C_1=c_1}}$$

Proof. From the definition of independent variables for M, we have:

$$\mu(a_i) = \mu(a_i|C_1 = c_1) = \mu(a_i|C_1 = c_1, C_2 = c_2)$$

For all $b_k \in M$, let $Avg_\mu(a_i)$ be the average value of the *measure* with the distribution μ_{a_i}.

$$Avg_\mu(a_i) = \sum_{b_k} \mu_{a_i}(b_k) * b_k$$

Let $Avg_\mu(a_i/C_1 = c_1, C_2 = c_2)$ be the average value of the *measure* with the distribution $\mu(a_i/C_1 = c_1, C_2 = c_2)$.

$$Avg_\mu(a_i/C_1 = c_1, C_2 = c_2) = \sum_{b_k} \mu_{(a_i/C_1=c_1,C_2=c_2)}(b_k) * b_k$$

Hence,

$$Avg_\mu(a_i/C_1 = c_1, C_2 = c_2) = \ Avg_\mu(a_i/C_1 = c_1) = Avg_\mu(a_i) \tag{5.6}$$

Assume that $N_{C_1=c_1,C_2=c_2,A=a_i}$ is the number of tuples such that $C_1 = c_1, C_2 = c_2$ and $A = a_i$.

Let $\sigma_{(C_1,C_2,A)}(c_1, c_2, a_i)$ be the probability such that $A = a_i$ and $C_1 = c_1, C_2 = c_2$.

We have C_1, C_2, A are independent variables. So:

$$\sigma_{C_1,C_2}(c_1, c_2) * \sigma_{C_1,A}(c_1, a_i) = \sigma_{C_1,C_2,A}(c_1, c_2, a_i) \tag{5.7}$$

Moreover,

$$\sigma_{(C_1,C_2,A)}(c_1, c_2, a_i) = \frac{N_{C_1=c_1,C_2=c_2,A=a_i}}{N} \tag{5.8}$$

$Avg_\mu(a_i/C_1 = c_1, C_2 = c_2)$, the average value of the *measure* with the distribution $\mu(a_i/C_1 = c_1, C_2 = c_2)$ can be calculated as follows:

$$Avg_\mu(a_i/C_1 = c_1, C_2 = c_2) = \frac{\sum_{C_1=c_1,C_2=c_2,A=a_i} t.M}{N_{C_1=c_1,C_2=c_2,A=a_i}} \tag{5.9}$$

Hence,

$$\begin{aligned}
\sigma_{C_1,C_2}(c_1, c_2) * Q^S_{C_1=c_1} &= \sigma_{C_1,C_2}(c_1, c_2) * \frac{\sum_{a_i} \sigma_{C_1,A}(c_1, a_i) * Avg_\mu(a_i)}{\sum_{c_1}\sum_{a_i} \sigma_{(C_1,A)}(c_1, a_i) * Avg_\mu(a_i)} \\
&= \frac{\sum_{a_i} \sigma_{C_1,C_2}(c_1, c_2) * \sigma_{C_1,A}(c_1, a_i) * Avg_\mu(a_i)}{\sum_{c_1}\sum_{a_i} \sigma_{(C_1,A)}(c_1, a_i) * Avg_\mu(a_i)} \\
&= \frac{\sum_{a_i} \sigma_{C_1,C_2}(c_1, c_2) * \sigma_{C_1,A}(c_1, a_i) * Avg_\mu(a_i/C_1 = c_1, C_2 = c_2)}{\sum_{c_1}\sum_{a_i} \sigma_{(C_1,A)}(c_1, a_i) * Avg_\mu(a_i)} && \text{(From (5.6))} \\
&= \frac{\sum_{a_i} \sigma_{C_1,C_2,A}(c_1, c_2, a_i) * Avg_\mu(a_i/C_1 = c_1, C_2 = c_2)}{\sum_{c_1}\sum_{a_i} \sigma_{(C_1,A)}(c_1, a_i) * Avg_\mu(a_i/C_1 = c_1)} && \text{(From (5.6) and (5.7))} \\
&= \frac{\sum_{a_i} (\frac{N_{C_1=c_1,C_2=c_2,A=a_i}}{N} * \frac{\sum_{C_1=c_1,C_2=c_2,A=a_i} t.M}{N_{C_1=c_1,C_2=c_2,A=a_i}})}{\sum_{c_1}\sum_{a_i} (\frac{N_{C_1=c_1,A=a_i}}{N} * \frac{\sum_{C_1=c_1,A=a_i} t.M}{N_{C_1=c_1,A=a_i}})} && \text{(From (5.8) and (5.9))}
\end{aligned}$$

$$= \frac{\sum_{a_i}(\sum_{C_1=c_1,C_2=c_2,A=a_i} t.M)}{\sum_{c_1}\sum_{a_i}(\sum_{C_1=c_1,A=a_i} t.M)}$$
$$= \frac{\sum_{C_1=c_1,C_2=c_2} t.M}{\sum t.M} \tag{5.10}$$

We consider the definition $Q^S_{C_1=c_1,C_2=c_2}$:

$$Q^S_{C_1=c_1,C_2=c_2} = \frac{\sigma_{C_1,C_2}(c_1,c_2) * Q^S_{C_1=c_1}}{\sum_{c_1}\sum_{c_2}\sigma_{C_1,C_2}(c_1,c_2) * Q^S_{C_1=c_1}}$$

By using (5.10), we have:

$$Q^S_{C_1=c_1,C_2=c_2} = \frac{\frac{\sum_{C_1=c_1,C_2=c_2} t.M}{\sum t.M}}{\sum_{c_1}\sum_{c_2}\frac{\sum_{C_1=c_1,C_2=c_2} t.M}{\sum t.M}} = \frac{\sum_{C_1=c_1,C_2=c_2} t.M}{\sum t.M} = Q^I_{C_1=c_1,C_2=c_2} \quad \square$$

5.3.3 Analysis of the *measure* on k dimensions

We analyze in the case when $C_1, C_2, ..., C_k, A$ are independent for M. Moreover, we assume that $C_1, C_2, ..., C_k, A$ are independent variables.

Theorem 5.3.4. *Let S be the statistical model with the sequence of distributions $\sigma_A, \sigma_{A,C_i}, \sigma_{C_i,C_j}$ for the dimensions C_i. Then, $Q^S_{C_1=c_1,...,C_k=c_k}$ is equal to $Q^I_{C_1=c_1,...,C_k=c_k}$ where:*

$$Q^S_{C_1=c_1,...,C_k=c_k} = \frac{\sigma_{C_{k-1},C_k}(c_{k-1},c_k) * Q^S_{C_1=c_1,...,C_{k-1}=c_{k-1}}}{\sum_{c_1}....\sum_{c_k}\sigma_{C_{k-1},C_k}(c_{k-1},c_k) * Q^S_{C_1=c_1,...,C_{k-1}=c_{k-1}}}$$

Proof. We first study with $k = 3$. We have:

$$Q^S_{C_1=c_1,C_2=c_2,C_3=c_3} = \frac{\sigma_{C_2,C_3}(c_2,c_3) * Q^S_{C_1=c_1,C_2=c_2}}{\sum_{c_1}\sum_{c_2}\sum_{c_3}\sigma_{C_2,C_3}(c_2,c_3) * Q^S_{C_1=c_1,C_2=c_2}}$$
$$= \frac{\sigma_{C_2,C_3}(c_2,c_3) * \frac{\sigma_{C_1,C_2}(c_1,c_2) * Q^S_{C_1=c_1}}{\sum_{c_1}\sum_{c_2}\sigma_{C_1,C_2}(c_1,c_2) * Q^S_{C_1=c_1}}}{\sum_{c_1}\sum_{c_2}\sum_{c_3}\sigma_{C_2,C_3}(c_2,c_3) * \frac{\sigma_{C_1,C_2}(c_1,c_2) * Q^S_{C_1=c_1}}{\sum_{c_1}\sum_{c_2}\sigma_{C_1,C_2}(c_1,c_2) * Q^S_{C_1=c_1}}}$$

By using the definition $Q^S_{C_1=c_1,C_2=c_2}$, we have:

$$\sigma_{C_2,C_3}(c_2,c_3) * Q^S_{C_1=c_1,C_2=c_2} = \sigma_{C_2,C_3}(c_2,c_3) * \frac{\sigma_{C_1,C_2}(c_1,c_2) * Q^S_{C_1=c_1}}{\sum_{c_1}\sum_{c_2}\sigma_{C_1,C_2}(c_1,c_2) * Q^S_{C_1=c_1}}$$

$$= \sigma_{C_2,C_3}(c_2,c_3) * \frac{\sigma_{C_1,C_2}(c_1,c_2) * \frac{\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i)}{\sum_{c_1}\sum_{a_i}\sigma_{(C_1,A)}(c_,a_i) * Avg_\mu(a_i)}}{\sum_{c_1}\sum_{c_2}\sigma_{C_1,C_2}(c_1,c_2) * \frac{\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i)}{\sum_{c_1}\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i)}}$$

$$= \frac{\sigma_{C_1,C_2,C_3}(c_1,c_2,c_3) * \frac{\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i)}{\sum_{c_1}\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i)}}{\sum_{c_1}\sum_{c_2}\sigma_{C_1,C_2}(c_1,c_2) * \frac{\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i)}{\sum_{c_1}\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i)}}$$

$$= \frac{\frac{\sum_{a_i}\sigma_{C_1,C_2,C_3,A}(c_1,c_2,c_3,a_i) * Avg_\mu(a_i)}{\sum_{c_1}\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i)}}{\sum_{c_1}\sum_{c_2}\frac{\sum_{a_i}\sigma_{C_1,C_2,A}(c_1,c_2,a_i) * Avg_\mu(a_i)}{\sum_{c_1}\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i)}} \quad \text{(By using (5.7))}$$

$$= \frac{\frac{\sum_{a_i}\sigma_{C_1,C_2,C_3,A}(c_1,c_2,c_3,a_i) * Avg_\mu(a_i/C_1,C_2,C_3)}{\sum_{c_1}\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i/C_1)}}{\sum_{c_1}\sum_{c_2}\frac{\sum_{a_i}\sigma_{C_1,C_2,A}(c_1,c_2,a_i) * Avg_\mu(a_i/C_1,C_2)}{\sum_{c_1}\sum_{a_i}\sigma_{(C_1,A)}(c_1,a_i) * Avg_\mu(a_i/C_1)}} \quad \text{(By using (5.6))}$$

$$= \frac{\frac{\sum_{a_i}\frac{N_{C_1=c_1,C_2=c_2,C_3=c_3,A=a_i}}{N} * Avg_\mu(a_i/C_1,C_2,C_3)}{\sum_{c_1}\sum_{a_i}\frac{N_{C_1=c_1,A=a_i}}{N} * Avg_\mu(a_i/C_1)}}{\sum_{c_1}\sum_{c_2}\frac{\sum_{a_i}\frac{N_{C_1=c_1,C_2=c_2,A=a_i}}{N} * Avg_\mu(a_i/C_1,C_2)}{\sum_{c_1}\sum_{a_i}\frac{N_{C_1=c_1,A=a_i}}{N} * Avg_\mu(a_i/C_1)}} \quad \text{(By using (5.8))}$$

$$= \frac{\frac{\sum_{a_i}\frac{N_{C_1=c_1,C_2=c_2,C_3=c_3,A=a_i}}{N} * \frac{\sum_{C_1=c_1,C_2=c_2,C_3=c_3,A=a_i} t.M}{N_{C_1=c_1,C_2=c_2,C_3=c_3,A=a_i}}}{\sum_{c_1}\sum_{a_i}\frac{N_{C_1=c_1,A=a_i}}{N} * \frac{\sum_{C_1=c_1,A=a_i} t.M}{N_{C_1=c_1,A=a_i}}}}{\sum_{c_1}\sum_{c_2}\frac{\sum_{a_i}\frac{N_{C_1=c_1,C_2=c_2,A=a_i}}{N} * \frac{\sum_{C_1=c_1,C_2=c_2,A=a_i} t.M}{N_{C_1=c_1,C_2=c_2,A=a_i}}}{\sum_{c_1}\sum_{a_i}\frac{N_{C_1=c_1,A=a_i}}{N} * \frac{\sum_{C_1=c_1,A=a_i} t.M}{N_{C_1=c_1,A=a_i}}}} \quad \text{(By using (5.9))}$$

$$= \frac{\frac{\sum_{a_i} \frac{\sum_{C_1=c_1,C_2=c_2,C_3=c_3,A=a_i} t.M}{N}}{\sum_{c_1} \sum_{a_i} \frac{\sum_{C_1=c_1,A=a_i} t.M}{N}}}{\frac{\sum_{c_1} \sum_{c_2} \frac{\sum_{a_i} \frac{\sum_{C_1=c_1,C_2=c_2,A=a_i} t.M}{N}}{\sum_{c_1} \sum_{a_i} \frac{\sum_{C_1=c_1,A=a_i} t.M}{N}}}{\sum_{t.M}}}$$

$$= \frac{\sum_{C_1=c_1,C_2=c_2,C_3=c_3} t.M}{\sum_{t.M}} \tag{5.11}$$

By using (5.11),

$$Q^S_{C_1=c_1,C_2=c_2,C_3=c_3} = \frac{\sigma_{C_2,C_3}(c_2,...,c_3) * Q^S_{C_1=c_1,C_2=c_{(2}}}{\sum_{c_1} \sum_{c_2} \sum_{c_3} \sigma_{C_2,C_3}(c_2,...,c_3) * Q^S_{C_1=c_1,C_2=c_{(2}}}$$

$$= \frac{\frac{\sum_{C_1=c_1,C_2=c_2,C_3=c_3} t.M}{\sum_{t.M}}}{\sum_{c_1} \sum_{c_2} \sum_{c_3} \frac{\sum_{C_1=c_1,C_2=c_2,C_3=c_3} t.M}{\sum_{t.M}}}$$

$$= \frac{\sum_{C_1=c_1,C_2=c_2,C_3=c_3} t.M}{\sum_{t.M}} = Q^I_{C_1=c_1,C_2=c_2,C_3=c_3} \tag{5.12}$$

From (5.12), the theorem is hold with $k = 3$. We prove the theorem is hold with an arbitrary k. By using the development technique such the case $k = 3$, we have

$$\sigma_{C_{k-1},..,C_k}(c_{k-1},...,c_k) * Q^S_{C_1=c_1,...,C_{k-1}=c_{k-1}} =$$

$$= \sigma_{C_{k-1},..,C_{k-1}}(c_{k-1},...,c_k) * \frac{\sigma_{C_{k-2},..,C_{k-1}}(c_{k-2},\ldots,c_{k-1}) * Q^S_{C_1=c_1,...,C_k=c_{k-2}}}{\sum_{c_1} \cdots \sum_{c_{k-1}} \sigma_{C_{k-2},..,C_{k-1}}(c_{k-2},\ldots,c_{k-1}) * Q^S_{C_1=c_1,...,C_{k-2}=c_{k-2}}}$$

$$= \frac{\sum_{C_1=c_1,...,C_k=c_k} t.M}{\sum_{t.M}} \tag{5.13}$$

By using (5.13),

$$
\begin{aligned}
Q^S_{C_1=c_1,\ldots,C_k=c_k} &= \frac{\sigma_{C_{k-1},C_k}(c_{k-1},c_k) * Q^S_{C_1=c_1,\ldots,C_{k-1}=c_{k-1}}}{\sum_{c_1} \ldots \sum_{c_k} \sigma_{C_{k-1},C_k}(c_{k-1},c_k) * Q^S_{C_1=c_1,\ldots,C_{k-1}=c_{k-1}}} \\
&= \frac{\frac{\sum_{C_1=c_1,\ldots,C_k=c_k} t.M}{\sum_{t.M}}}{\sum_{c_1} \ldots \sum_{c_k} \frac{\sum_{C_1=c_1,\ldots,C_k=c_k} t.M}{\sum_{t.M}}} \\
&= \frac{\sum_{C_1=c_1,\ldots,C_k=c_k} t.M}{\sum_{t.M}} = Q^S_{C_1=c_1,\ldots,C_k=c_k} \qquad \square
\end{aligned}
$$

From the above theorem, we see that there is the binary relationship between $Q^S_{C_1=c_1,\ldots,C_k=c_k}$ and $Q^S_{C_1=c_1,\ldots,C_k=c_{k-1}}$. Moreover, we need only binary distributions to approximate answers. In Chapter 7, we will present the design for counters to evaluate the exact distributions.

5.4 Learning a statistical model by sampling

To learn the statistical model, we can sample with uniform distribution to find the approximation of distributions σ. We use the (ε, δ)-approximate algorithm where ε is the error and $1-\delta$ is the confidence. The number of samples must follows a function of ε and δ.

In the next lemma, the distribution $C.A$ can be approximated with the error ratio ε and the confidence $1-\delta$ by sampling m tuples.

Lemma 5.4.1. *We can ε-approximate the distribution over $C.A$ with probability $1-\delta$ if $m > \frac{1}{2}.(\frac{|C| * |A|}{\varepsilon})^2.\log\frac{1}{\delta}$ samples and N large enough.*

Proof. We approximate the distribution $\sigma_{C.A}(c_j, a_i)$ by the density of tuples for all values $c_j \in C$ and $a_i \in A$. We consider m uniform samples. Let

$$d_1(j,i) = \frac{|\{t \ : \ t.C = c_j \wedge t.A = a_i\}|}{m}$$

As $\sum_{j,i} d_1(j,i) = 1$, we interpret d_1 over the values $c_j \in C$ and $a_i \in A$ as $\sigma_{C.A}(c_j, a_i)$. Let us show that d_1 and $\sigma_{C.A}(c_j, a_i)$ are ε-close.

For each $c_j \in C$ and $a_i \in A$, let $X_k = 1$ if the k-th tuple of $\widehat{I}$ is such that $t.C = c_j$ and $t.A = a_i$. Otherwise, $X_k = 0$.

then

$$I\!E(X_k) = \sigma_{C.A}(c_j, a_i) \text{ and } d_1(j,i) = \frac{\sum_k X_k}{m}$$

and,

$$I\!E(d_1(j,i)) = I\!E(\frac{\sum_k X_k}{m}) = \sigma_{C.A}(c_j, a_i)$$

.

As the tuples are taken *independently*, we can apply a Chernoff-Hoeffding bound [16] with the absolute error form.

$$Pr[\mid d_1(j,i) - I\!E(d_1(j,i)) \mid \geq t] \leq e^{-2t^2.m}$$

In this form, t is the error and $\delta = e^{-2t^2.m}$ is the confidence. We set $t = \frac{\varepsilon}{\mid C \mid * \mid A \mid}$, and $\delta = e^{-2t^2.m}$ and conclude that if $m > \frac{1}{2}.(\frac{\mid C \mid * \mid A \mid}{\varepsilon})^2.\log\frac{1}{\delta}$ then:

$$Pr[\mid d_1(j,i) - \sigma_{C.A}(c_j, a_i) \mid \leq \frac{\varepsilon}{\mid C \mid * \mid A \mid}] \geq 1 - \delta.$$

With a union bound on $\forall c_j \in C$ and $\forall a_i \in A$, we conclude that:

$$Pr[(\sum_{c_j}\sum_{a_i} \mid d_1(j,i) - \sigma_{C.A}(c_j, a_i) \mid) \leq \varepsilon] \geq 1 - \delta \qquad \square$$

We now describe how to approximate OLAP queries, assuming $A \triangleleft M$ for some attribute A. To answer the OLAP query on dimension C, it is enough to keep the distribution over $C.A$. For the same reason, as in the previous lemma, we can approximate the distribution $C.A$.

Theorem 5.4.1. *If $A \triangleleft M$, we can ε-approximate Q_C^I by Q_C^S with probability $1 - \delta$ if $m > (\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ samples and N large enough.*

Proof. We consider each Q_C^I which is $\varepsilon/2$-approximated by $Q_C^{\widehat{I}}$ for $m > (\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$. But $I\!E(Q_C^{\widehat{I}}) = Q_C^S$, hence we can also apply the Hoeffding bound and conclude that

Q_C^S $\varepsilon/2$-approximates $Q_C^{\widehat{I}}$ for $m > (\frac{|C|}{\varepsilon})^2 . \log \frac{1}{\delta}$. By the triangular inequality, we get the theorem. □

5.5 Conclusion

We have just presented a new approximate method for OLAP query answering. With the statistical model, to find the close answers, we need only the informations about statistical dependencies. If the statistical dependencies exist in data warehouse, we showed also how to approximate the distributions of statistical dependencies and the distribution of some attributes. The advantage of this method is the compact structure. It requires only small space and small time to approximate. It makes the simpler analysis of approximation.

We can apply the method of statistical model when we had known the existence of statistical dependencies in data warehouse. So, how can we discover all statistical dependencies when we don't have their informations. The chapter 8 will give us the mining of this kind of dependencies.

The next chapter continues to study the approximation in the OLAP data exchange context where a data warehouse is also build from some different sources.

CHAPTER 6

OLAP Data Exchange

6.1 Context

In the context of OLAP data exchange in Figure 6.1 , we consider the situation where k different sources feed a data warehouse I. For example, the relation I_1 of source S_1 feeds the data from England, the relation I_2 of source S_2 feeds the data from France, etc. We want to select $\widehat{I}_i$ made of m_i samples from each source and define $\widehat{I}_e = \widehat{I}_1 \cup \widehat{I}_2 \cup ... \cup \widehat{I}_k$ where each $\widehat{I}_i$ follows the uniform distribution. We ask which m_i guarantee that any OLAP query Q on I will be well approximated by $\widehat{I}_e$. We first consider the uniform distribution, the *measure*-based distribution and the statistical model.

6.2 Approximation with the uniform distribution

If each source corresponds to a *unique* attribute value $b \in B$ (country for example), let r_B be the distribution which gives the density of tuples of each source. If $r_B(U.K) = 1/4$, then $1/4$ of the tuples of the target come the source "U.K" and each source is identified with a specific country. We say that I is the union on B of $I_1, I_2, ..., I_k$ if there are k distinct sources corresponding to k different countries.

We will select $\widehat{I}_i$ with $m_i = m * r_B(b_i)$ in I_i with the uniform distribution, and let $\widehat{I}_e = \widehat{I}_1 \cup \widehat{I}_2 \cup ... \cup \widehat{I}_k$.

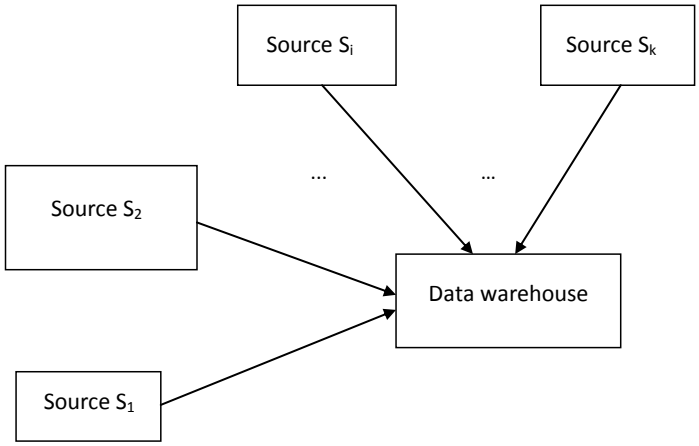


Figure 6.1: An OLAP data exchange context.

We want to guarantee the approximation of any OLAP query, using $\widehat{I_e}$ for any attribute.

Theorem 6.2.1. *If* $m \geq \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ *and* $m_i = m * r_B(b_i)$ *uniform samples are selected in each* I_i*, then the answer* $Q^{\widehat{I_e}}_{C_1,\ldots,C_p}$ *to any query* Q *on dimensions* $C_1,\ldots,C_p$*, is* ε*-close to* $Q^{I}_{C_1,\ldots,C_p}$ *with probability* $1-\delta$*.*

Proof. In N tuples of the target, we have $N * r_B(b_i)$ tuples of the source i. Notice that in $\widehat{I_e}$, we have exactly m_i tuples in I_i.

Consider a uniform sampling $\widehat{I_u}$ of m tuples on I. If we expect m_i tuples for each source, what is the probability to get m_i tuples in m tuples? We can show that the probability to get the number of tuples in the interval $[m_i(1-\varepsilon), m_i(1+\varepsilon)]$ is large by using Chernoff-Hoeffding bound.

So, any time we have m tuples on I, the number of tuples of each source i is ε-close to m_i. In the worst case, we can delete $\varepsilon * m_i$ tuples. For the relative distribution, it has the small impact because the *measure* is bounded.

Hence, the answers is close in the sense that the probability that we get m_i tuples in $[m_i(1-\varepsilon), m_i(1+\varepsilon)]$ is large. And from that, the error of making a query is proved as follow:

In the case of a single dimension C, we can show that,

$$Pr[|\ Q^{\widehat{I_u}}_{C} - Q^{\widehat{I_e}}_{C}\ | \leq \varepsilon] \geq 1-\delta$$

We use the Chernoff-Hoeffding bound [16] and a union bound (as in the case of sampling):

$$Pr[\mid Q_C^I - Q_C^{\widehat{I_u}} \mid \leq \varepsilon] \geq 1 - \delta$$

With the triangular inequality, we conclude that:

$$Pr[\mid Q_C^I - Q_C^{\widehat{I_e}} \mid \leq 2.\varepsilon] \geq 1 - \delta \qquad \square$$

6.3 Approximation with the *measure*-based distribution

In the case of the *measure*-based distribution $\widehat{I_M}$, we can't take m_i samples on each source I_i with the distribution $\widehat{I_{M,i}}$, because the union of such samples determines a distribution which may be far from $\widehat{I_M}$ on I. It is possible that the source 1 has very low measures and high density (3/4 for example), whereas, the source 2 has large measures and low density (1/4). In I_M the density of the tuples of source 2 will be higher than 1/4 because less tuples of source 1 will be selected. How could we combine the sources in this case?

If B follows the distribution r_B, let $\mu(b_i)$ the distribution of the measure for the value $B = b_i$, i.e. for the source i, and let $Avg_\mu(b_i)$ be the average of this distribution.

Let us define m'_i by:

$$m'_i = m * \frac{r_B(b_i) * Avg_\mu(b_i)}{\sum_i r_B(b_i) * Avg_\mu(b_i)}$$

A theorem similar to Theorem 6.2.1 could then be stated, where we replace the m_i uniform samples on each source I_i by m'_i samples with the *measure*-based distribution on each source. Let $\widehat{I_e} = \widehat{I_1} \cup \widehat{I_2} \cup ... \cup \widehat{I_k}$ where each $\widehat{I_i}$ is the set of samples of source i with the *measure*-based distribution.

Theorem 6.3.1. *If* $m \geq \frac{1}{2}.(\frac{|C|}{\varepsilon})^2.\log\frac{1}{\delta}$ *and* $m'_i = m * \frac{r_B(b_i) * Avg_\mu(b_i)}{\sum_i r_B(b_i) * Avg_\mu(b_i)}$ *samples are taken in each source* i*, then the answer* $Q^{\widehat{I_e}}_{C_1,...,C_p}$ *to any query* Q *on dimensions* $C_1, \ldots, C_p$*, is* ε*-close to* $Q^I_{C_1,...,C_p}$ *with probability* $1 - \delta$*.*

Proof. Consider a *measure*-based sampling $\widehat{I_M}$ of m tuples on I.

Let $N_{B=b_i}$ be the number of tuples such that $B = b_i$.

$$\frac{r_B(b_i) * Avg_\mu(b_i)}{\sum_i r_B(b_i) * Avg_\mu(b_i)} = \frac{\frac{N_{B=b_i}}{N} * \frac{\sum_{B=b_i} t.M}{N_{B=b_i}}}{\sum_i \frac{N_{B=b_i}}{N} * \frac{\sum_{B=b_i} t.M}{N_{B=b_i}}} = \frac{\sum_{B=b_i} t.M}{\sum t.M}$$

With the *measure*-based sampling, if a tuple is selected, its *measure* is replaced by 1. So, we have

$$\frac{r_B(b_i) * Avg_\mu(b_i)}{\sum_i r_B(b_i) * Avg_\mu(b_i)} = \frac{\sum_{B=b_i} t.M}{\sum t.M} = \frac{\sum_{B=b_i} 1}{m} = \frac{N_{B=b_i}}{m}$$

Hence, $\frac{r_B(b_i) * Avg_\mu(b_i)}{\sum_i r_B(b_i) * Avg_\mu(b_i)}$ is the density of tuples of each source.

The m tuples contain approximately $m'_i = m * \frac{r_B(b_i) * Avg_\mu(b_i)}{\sum_i r_B(b_i) * Avg_\mu(b_i)}$ tuples of the source i, with high probability, distributed with the *measure*-based distribution on I_i. The error of answers is relative to the error of number of tuples m_i. So, the difference for each component of query answer is small.

In the case of a single dimension C, we can show that:

$$Pr[\mid Q_C^{\widehat{I_M}} - Q_C^{\widehat{I_e}} \mid \leq \varepsilon] \geq 1 - \delta$$

We use the Chernoff-Hoeffding bound [16] and a union bound (as in the case of sampling):

$$Pr[\mid Q_C^{I} - Q_C^{\widehat{I_M}} \mid \leq \varepsilon] \geq 1 - \delta$$

With the triangular inequality, we conclude that:

$$Pr[\mid Q_C^{I} - Q_C^{\widehat{I_e}} \mid \leq 2.\varepsilon] \geq 1 - \delta$$ □

6.4 Approximation by the statistical model in data exchange

6.4.1 Conditions for the approximation

In a data exchange setting, we can apply the same analysis as in the previous section if we assume the *same type of statistical model* for each source. We need to combine both hypothesis:

- For each source, there is the same attribute A such that $A \triangleleft M$,
- Each source corresponds to a *distinct* attribute value $b \in B$ and B follows a fixed distribution in the data warehouse r_i

We can then combine the results of the preceding sections and determine:

- The approximate models of each source, i.e. the distributions $\sigma_{C,A}$ on $C.A$, for each source $j = 1, ..., k$.
- The weight m_i of each source computed as in theorem 6.2.1, such that $\sum_i m_i = m$ from the distribution σ_B. In this case, it is simpler to assume the uniform distribution on each source.

6.4.2 Different evaluation methods

We suppose that the data warehouse is built from three different sources. The density of tuples or the rate of each source in the target data warehouse is different. Assume that the rate of source 1 is $r_1 = \frac{m1}{m}$. The rate of sources 2 and 3 are $r_2 = \frac{m2}{m}$ and $r_3 = \frac{m3}{m}$.

Let $Q^k_{C=c_j}$ be the approximation of Q on $C = c_j$ by source k. Then, $Q^S_{C=c_j}$ the *approximation by the statistical model* is computed by one of three following methods:

1. The first way (described in Figure 6.2): the distributions $\sigma_{C.A}(c_j, a_i)$ are estimated by the target. The target does not stock the tuples. It stocks the distributions. This method is adapted to streaming data, described further in the next chapter. When new tuples arrive, we update $\sigma_{C.A}(c_j, a_i)$.
2. The second way (described in Figure 6.3): the target asks each source to estimate $\sigma_{C.A}(c_j, a_i)$. Then, these informations are sent and updated in the target when we need to approximate the answers.
3. The third way (described in Figure 6.4): the target asks each source its answer to the query. Then, these answers are sent to the target which then interpolate in order to approximate the global answer.

Theorem 6.4.1. *If $A \triangleleft M$ on each source, we can ε-approximate Q^I_C by $Q^S_C = \sum_k \frac{m_k}{m} * Q^k_C$ with probability $1 - \delta$ if $m > (\frac{|C|}{\varepsilon})^2 . \log \frac{1}{\delta}$ samples and N large enough.*

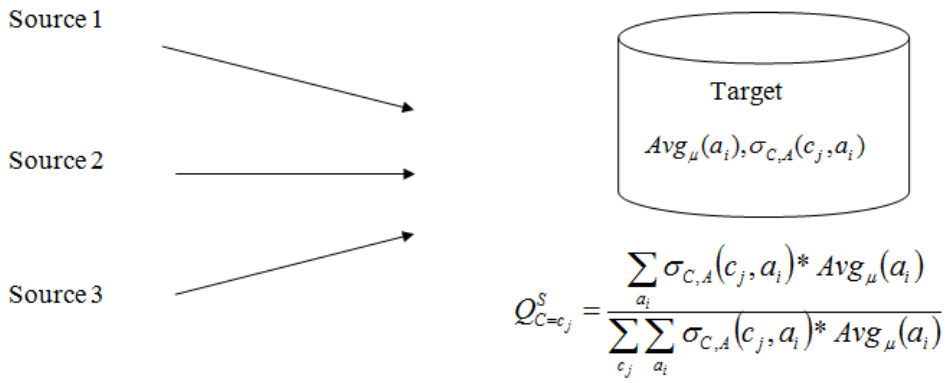


Figure 6.2: Statistical computation by the target.

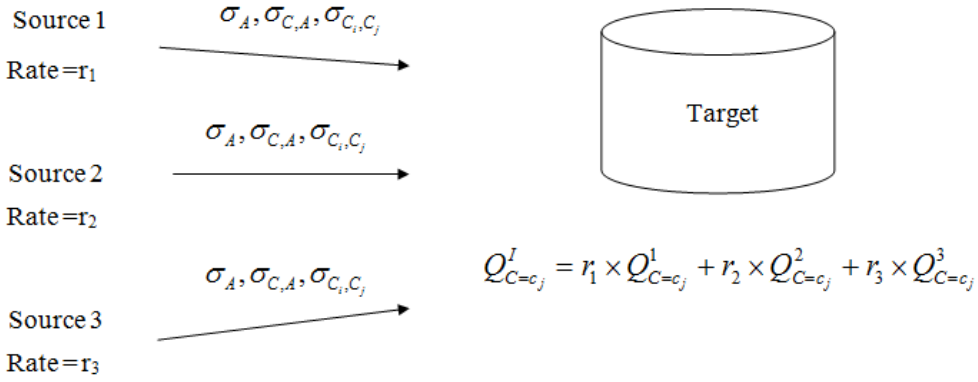


Figure 6.3: Statistical computation by the sources.

Proof. Consider a uniform sampling $\widehat{I_u}$ on I. It will contain approximately m_i tuples of each source, with high probability, distributed uniformly on I_i. Hence, Q^S_C is close to $Q^{\widehat{I_u}}_C$.

In the case of a single dimension C,

$$Pr[\mid Q^{\widehat{I_u}}_C - Q^S_C \mid \leq \varepsilon] \geq 1 - \delta$$

We use the Chernoff-Hoeffding bound [16] and a union bound (as in the case of sampling):

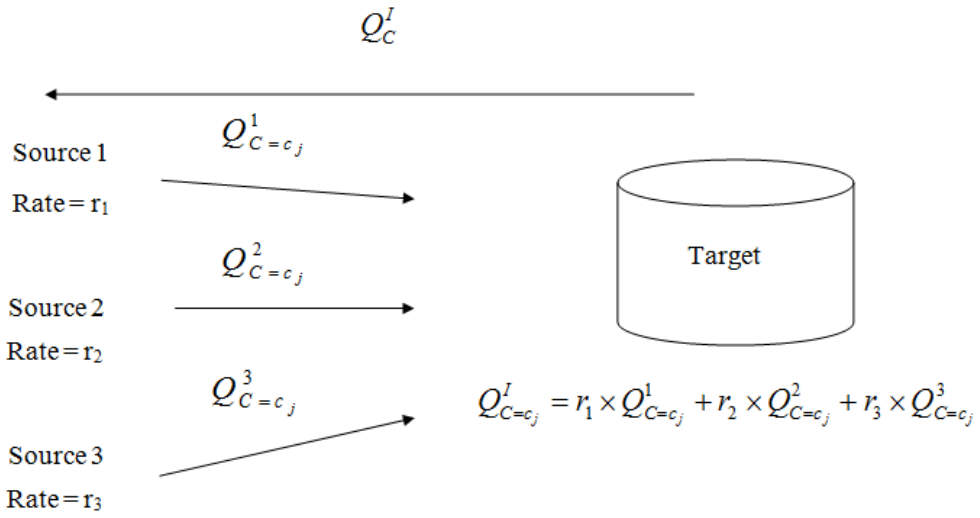


Figure 6.4: Approximate answer at each source.

$$Pr[\mid Q_C^I - Q_C^{\widehat{I_u}} \mid \leq \varepsilon] \geq 1 - \delta$$

With the triangular inequality, we conclude that:

$$Pr[\mid Q_C^I - Q_C^S \mid \leq 2.\varepsilon] \geq 1 - \delta \qquad \square$$

6.5 Conclusion

This chapter presents the techniques to approximate OLAP queries for a data warehouse, which combines different sources. The number of samples on each source depends on the sampling technique. Different answers can be used for the statistical model, on the union of the sources. They all give close answers with ε accuracy and $1 - \delta$ confidence.

CHAPTER 7

Streaming Data

7.1 Context

We now consider the construction of data warehouses. Where do the data come from? In the Data Exchange setting, several Sources send their tuples to a Target database. They may be modified by the Target and will end up in the Data Warehouse. This process is called ETL (Extract, Transform, Load) and is well automated.

There are many applications where the Sources continuously send their tuples to various clients. Sensors may send their data periodically to a central site and News Sources (BBC, CNN,..) send RSS feeds to their subscribers whenever new information arises. In both cases, these sources send XML data, which can be easily converted into a tuple of a relation. We view these tuples as Streams, which can be stored in a data warehouse.

In the Streaming model, data flows continuously and one of main questions is whether we need to store it or whether we can replace it by a much smaller memory. In precise terms, can we replace data of size $O(n)$ by some other data of size $O((\log n)^k)$, i.e. of size polylogarithmic in order to answer specific queries? In our situation, we want to approximately answer OLAP queries. We first consider a lower bound, directly obtained from Communication Complexity, and then proceed with approximate solutions, first with blocks of the stream and then with a learning method, when the data follow a statistical model.

The stream is the sequence of tuples $t_1, ..., t_n$ of the data warehouse I. In Figure 7.1

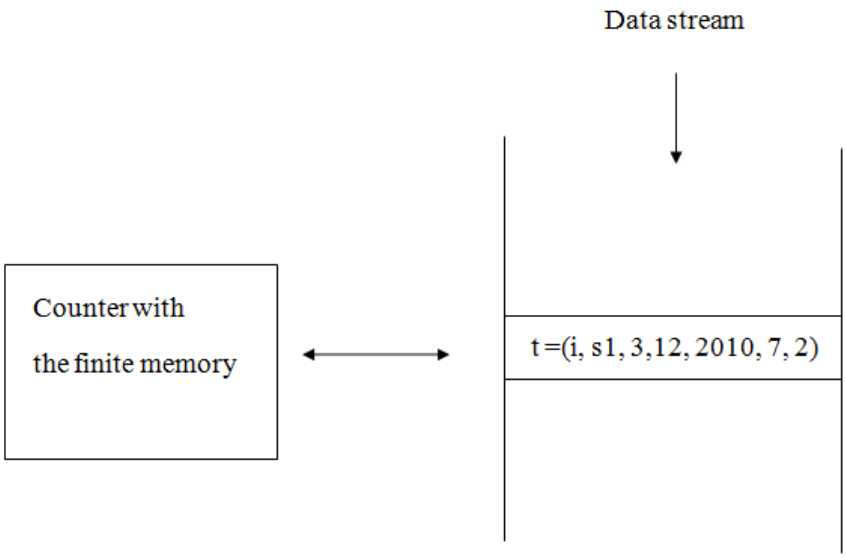


Figure 7.1: Streaming model.

for the schema 9.5, $t_i = (i, s1, 3, 12, 2010, 7, 2)$ stating that sensor $s1$ measures 7 hours of sunlight and 2 hours of rain on December 3rd 2010. The auxiliary tables such as *C(CITY,COUNTRY)* are fixed and independent of the stream.

7.2 Lower Bounds on the space complexity for unbounded domains

Communication Complexity [21] studies the number of bits Alice and Bob must exchange to compute a function $f(x, y)$ when Alice holds x and Bob holds y. In a *Protocol* $P(x, y)$ which combines the decisions of both Alice and Bob, the complexity of $P(x, y)$ is the number of bits $|P(x, y)|$ sent between Alice and Bob, i.e. $C(P) = Max_{x,y}|P(x, y)|$. Let $D(f)$ be the Minimum $C(P)$ over all deterministic protocols to compute the function, i.e. $D(f) = Min_P C(P)$ and $R_\varepsilon(f)$ be the Minimum $C(P)$ over randomized protocols with public coins and error ε, i.e. $Prob[P(x, y) \neq f(x, y)] \leq \varepsilon$, $R_\varepsilon(f) = Min_P C(P)$. In the *one-way model*, only Alice sends bits to Bob. In this case, we define the one-way Communication Complexity $\overrightarrow{D(f)}$ and $\overrightarrow{R_\varepsilon(f)}$ as before.

The memory M used by a deterministic (resp. randomized) streaming algorithm is always less then $D(f)$ (resp. $R_\varepsilon(f)$). Suppose a streaming algorithm computes $f(X)$ and let us write the stream $X = x|y$ as the concatenation of the input x with the input y. We can conceive the protocol where Alice transmits the Memory content

M to Bob who can then computes $f(X)$. Hence $\overrightarrow{D(f)} \leq M$.

Therefore a lower bound on the Communication Complexity provides a lower bound on the space of a streaming algorithm. If $x, y \subseteq \{1, 2, ..., n\}$, let $DISJ\ (x, y) = 1$ if $x \cap y = \emptyset$ and 0 otherwise. A classical result shows that $\overrightarrow{D(f)} = O(n)$ and $\overrightarrow{R_\varepsilon(f)} = O(n)$.

Consider the special case of an OLAP schema S where the data warehouse I_3 contains the tuples:

A	M
a_1	1
a_2	1
...	...
a_n	1

Table 7.1: Data warehouse I_3.

A typical situation is when m is known, but n is unknown The frequency $f_j = |\{i \in [1; n], a_i = j\}|, j \in [1; m]$. The moments $F_k = \sum_{j=1}^{m} (f_j)^k$ and $F_\infty = \max_j f_j$. Notice that F_0 is the number of distinct values j, whereas $F_1 = n$ and F_2 is the repeat rate.

Classical results show that F_0 can be estimated with randomized algorithms with a memory of $O(\log m)$, F_1 with a memory of $O(\log(\log n))$, and F_2 with a memory $O(\log n + \log m)$.

In this case, we have one dimension (the first attribute) and all tuples have the same *measure*, hence the analysis on A counts the number of occurrences of the a_i. In this case F^∞ is hard to approximate and we can reduce the approximation of F^∞ to the approximation of the OLAP query on A. Let $A[j]$ the density of the elements $a_i = j$.

Theorem 7.2.1. *The approximation of the OLAP query on dimension A requires a memory $O(n)$, i.e. proportional to the length of the stream. [26]*

Proof. We use the classical reduction from $DISJ\ (x, y)$ to F_∞. Let $x, y \in \{0, 1\}^n$ be the input to $DISJ$ and let $X_x = i_1, i_2, ..., i_k$ for $i_j \in \{1, 2, ..., n\}$ such that $i_j < i_{j+1}$ and $x_{i_j} = 1$. For example if $x = 011101$ then $X_x = \{2, 3, 4, 6\}$. If $x \cap y = \emptyset$ then $F^\infty = 1$ and if $x \cap y \neq \emptyset$ then $F^\infty = 2$. Therefore if we could approximate F^∞, we could decide $DISJ$. In our context, the approximation of the heaviest component is precisely F^∞, i.e. there is a j such that $A[j] * n = F^\infty$. As $DISJ\ (x, y)$ requires $\Omega(n)$ space for any randomized protocol, so does the approximation of the OLAP query on A. □

As the answer to OLAP queries requires $\Omega(n)$ space, we can only hope for gains of constant factors, in the general case.

7.3 Bounded domains

The previous lower bound implicitly uses the fact that $j \in [1; m]$ and m is arbitrarily large. In our situation, most of the dimension attributes C_i range over finite domains, except for the time attribute T.

T	C	B	A	M
t_1	c_1	b_1	a_1	10
t_2	c_2	b_2	a_2	20
...	...	...	...	...

Table 7.2: Data warehouse I_4.

In this example, the attribute T (time) has an an unbounded domain, whereas C, B, A range over a finite domain D. If we assume that $A \triangleleft M$, and that B, C are independent of A for M, we need the distributions:

- σ_A for the analysis on A
- $\sigma_{B,A}$ for the analysis on B
- $\sigma_{C,A}$ for the analysis on C
- $\sigma_{C,B}$ for the analysis on C, B

Each distribution is kept in matrices of size $|D|$ or $|D|^2$, used as counters.

- $A[i]$ stores $N * \sigma_A(a_i)$
- $BA[i, j]$ stores $N * \sigma_{B,A}(b_i, a_j)$
- $CA[i, j]$ stores $N * \sigma_{C,A(c_i,a_j)}$
- $CB[i, j]$ stores $N * \sigma_{C,B(c_i,b_j)}$

For each tuple t of the stream, we just update the counters:

- if $t.A = a_i$, we increase $A[i]$ of $+1$

- if $t.B = b_i \wedge t.A = a_j$, we increase $BA[i, j]$ of + 1
- if $t.C = c_i \wedge t.A = a_j$, we increase $CA[i, j]$ of +1
- if $t.C = c_i \wedge t.B = b_j$, we increase $CB[i, j]$ of +1

The design for counters in Table 7.2.

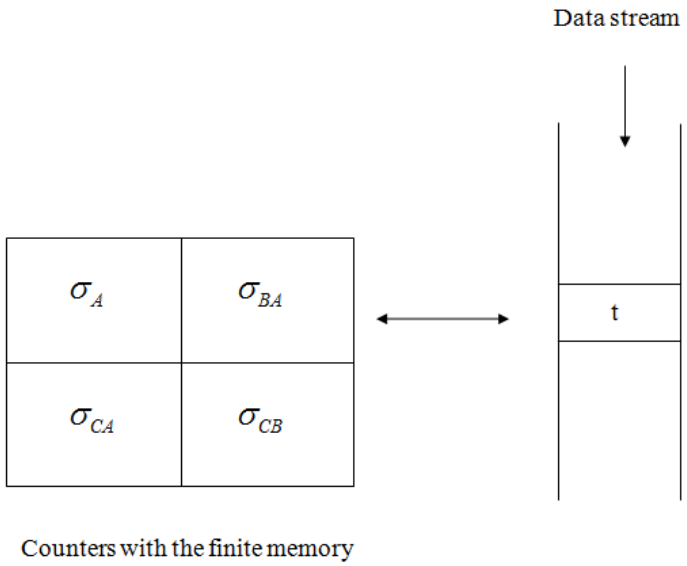


Figure 7.2: Design of counters.

7.4 Design of counters for the analysis on more than one dimension

In the case the analysis on more than one dimension, we need only the counter for the binary distributions: σ_A, $\sigma_{C_i,A}$ and σ_{C_i,C_j}. All these distributions are binary and suffice to answer OLAP queries on many dimensions. $\sigma_{C_i,\ldots,C_k,\ldots,C_j}$ which would require a lot more space.

The approach in this section is to organize the minimal number of counters, hence of space, to evaluate the exact distributions σ. Notice, that we compute exact distributions in this case.

7.5 Conclusion

In this chapter, we describe the context of streaming data where a data warehouse is built by different data streams. We showed lower bounds on the space complexity for unbounded domains. We introduce specific counters to evaluate the distributions of a statistical model. This approach optimizes the memory space.

CHAPTER 8

Mining Statistical Dependencies

8.1 Introduction

In this chapter, we describe a method to discover the statistical dependencies. In general, we may not know these dependencies. We are looking for them based on the decision tree.

The statistical dependencies generalize the classical functional dependencies as we explain in Chapter 5. In a statistical dependency, a set of attributes determines fixed distributions of the *measure*, with high probability.

Decision trees represent a functional dependency as they visualize how the values of few attributes such as A, B can predict the value of a target attribute. In another context, the *measure* M is the target attribute. The decision tree is generalized in Figure 8.1.

In class decision tree, we try select some attributes based on the information gain. In an exact decision tree, each leaf presents the value of the target attribute M. In this case, this value is predict with high probability. In the case of an approximate decision tree, each leaf predicts the distribution of the target.

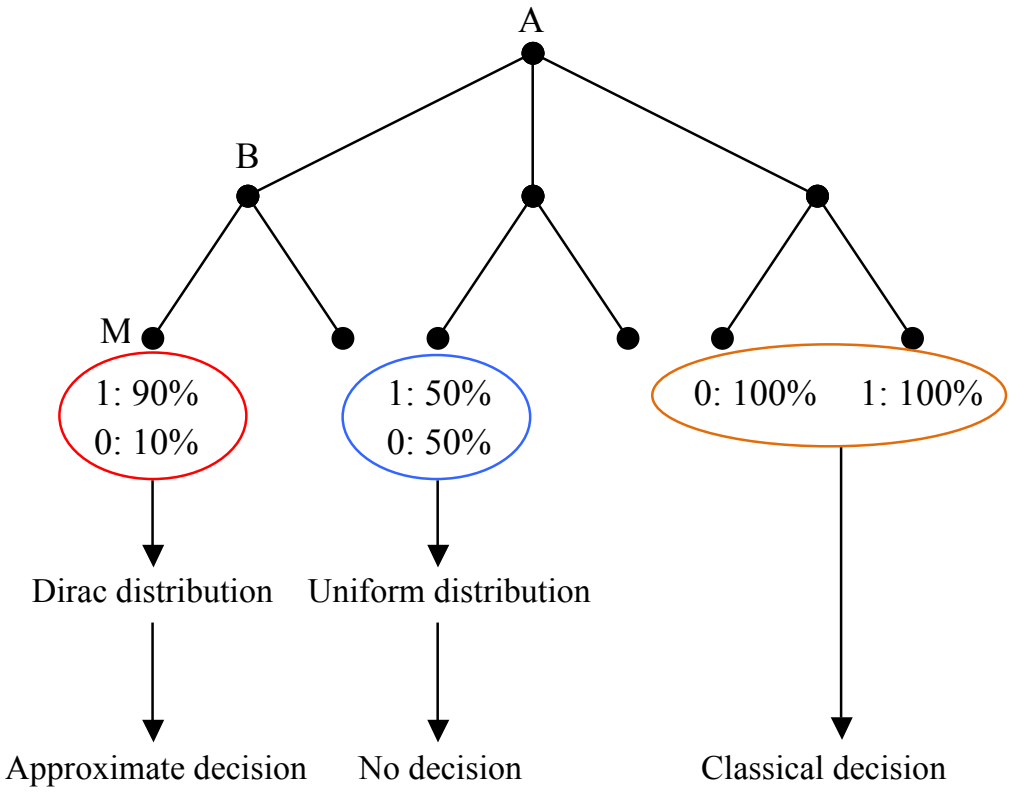


Figure 8.1: Generalize the decision tree

8.2 Entropy and information gain

The entropy of a source S, as a distribution over finite is defined as

$$Entropy(S) = \sum_i -p_i log_2 p_i$$

where p_i is the probability that S is equal i.

The information gain of the attribute A, $Gain(S, A)$, of the distribution S, is define as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{\mid S_v \mid}{\mid S \mid} Entropy(S_v)$$

where $Values(A)$ is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v (i.e., $S_v = s \in S \mid A(s) = v$).

Note the first term in the definition for Gain is just the entropy of the original source S. The second term is simply the sum of the entropies of each subset S_v, weighted

by the fraction of examples $\frac{\mid S_v \mid}{\mid S \mid}$ that belong to S_v. $Gain(S, A)$ is therefore the expected reduction in entropy caused by knowing the value of attribute A.

In classical data mining, we select attributes according to the maximal information gain. Given a set of m_0 samples for which we know all the attributes, we partition it randomly into a set L (learning set) and a set T (testing set) of equal size. Given L, we construct a decision tree, which provides a prediction of M for tuples. We can then compare for each tuple of T, the prediction with the real value. This determines the error rate, which we want to maintain at ε (about 10%).

8.3 Criteria of mining statistical dependencies

Given a set of m_0 tuples for which we know the *measure*, we randomly divide it into 3 sets of approximate equal sizes: two learning sets L_1 and L_2 and a testing set T. This is a fundamental difference with the classical method.

For each attribute A and each value a_i, we can estimate the distribution $\mu_1(a_i)$, i.e. the distribution of the target values for L_1 and similarly for L_2. We can then estimate the distances between the two distributions and define

$$d(A) = \sum_i |\mu_1(a_i) - \mu_2(a_i)|_1$$

Given the two learning sets L_1 and L_2, we can observe the following indicators:

1. The information gain of each attribute A for L_1,
2. The information gain of each attribute A for L_2,
3. For each attribute A, the distance $d(A)$.

In our context the main criteria is the distance and we want to find attributes such that $d(A)$ is small. If no attribute satisfies this criteria, we look for pairs A_i, A_j, and then triples and so on.

If there is a functional dependency $A- > M$, the information gain is maximal in L_1 and in L_2, and the distance between the two distributions is null. If we tolerate errors, the distance would be small.

However, there may be a small distance and a low information gain, and even a zero information gain. This is is achieved with the uniform distributions on the measure. In this case, the distance $d(A)$ remains small.

The small distance criteria is hence more general than the information gain and allows us to generalize the construction of decision trees. We can conceive the following algorithm.

8.4 Algorithm

The basic learning technique is to compute $d(A_i)$ for each attribute A_i, and eventually $d(A_i, A_j)$ for pairs of attributes, $d(A_i, A_j, A_k)$ for triples of attributes and so on.

We first test if $(d(A_i) < \varepsilon)$. If it is the case, A_i is a potential dependency $A_i \triangleleft M$ but we need to test it, using the set T. Let us define the learnt distribution $\mu(a_i)$ as:

$$\mu(a_i) = (\mu_1(a_i) + \mu_2(a_i))/2$$

It is indeed the distribution on all the learning tuples, in $L_1 \cup L_2$.

In order to test $\mu(a_i)$, we first compute the analogue distribution $\delta(a_i)$ on the test set T, i.e. the distribution of the measures of tuples in T for which $A = a_i$. We then test if $|\mu(a_i) - \delta(a_i)|_1 < \varepsilon$. This generalizes the classical test in a decision tree.

Data: Two training sets L_1, L_2, a test set T and a parameter ε
Result: A set of statistical dependencies S
/ * Mining for unique attributes */ ;
for *each* A_i **do**
  Compute $d(A_i)$ from L_1 and L_2: $d_1(A) = \sum_i |\mu_1(a_i) - \mu_2(a_i)|_1$;
  if $d_1(A) \leq \varepsilon$ **then**
    $\mu(a_i) = \dfrac{(\mu_1(a_i) + \mu_2(a_i))}{2}$;
    if $d_2(A) = \sum_i |\mu(a_i) - \delta(a_i)|_1 < \varepsilon$ **then**
      $S = S \cup \{A_i \triangleleft M\}$;
    end
  end
end
if *no unique attribute is found* **then**
  Look for pairs, triplets, ...;
end

Algorithm 3: Mining algorithm

8.5 Graphic representation of statistical dependencies

Suppose that the *measure* M has 10 possible values and we have discovered the statistical dependency $A, B \triangleleft M$. $\forall a_i \in A$, $\forall b_j \in B$, and $\forall m_k \in M$, each tuple (a_i, b_j) predicts the distribution of m_k when $A = a_i$ and $B = b_j$.

$$\mu_{(a_i,b_j)}(m_k) = Pr(t.M = m_k / t.A = a_i, t.B = b_j)$$

Assume that there are three types of distributions on leaves of this tree: μ_1, μ_2 and μ_3. Where $\mu_1 = \{\mu_{1(a_i,b_j)}(m_k)\}$, $\mu_2 = \{\mu_{2(a_i,b_j)}(m_k)\}$ and $\mu_3 = \{\mu_{3(a_i,b_j)}(m_k)\}$.The decision tree represents this dependency in Figure 8.2.

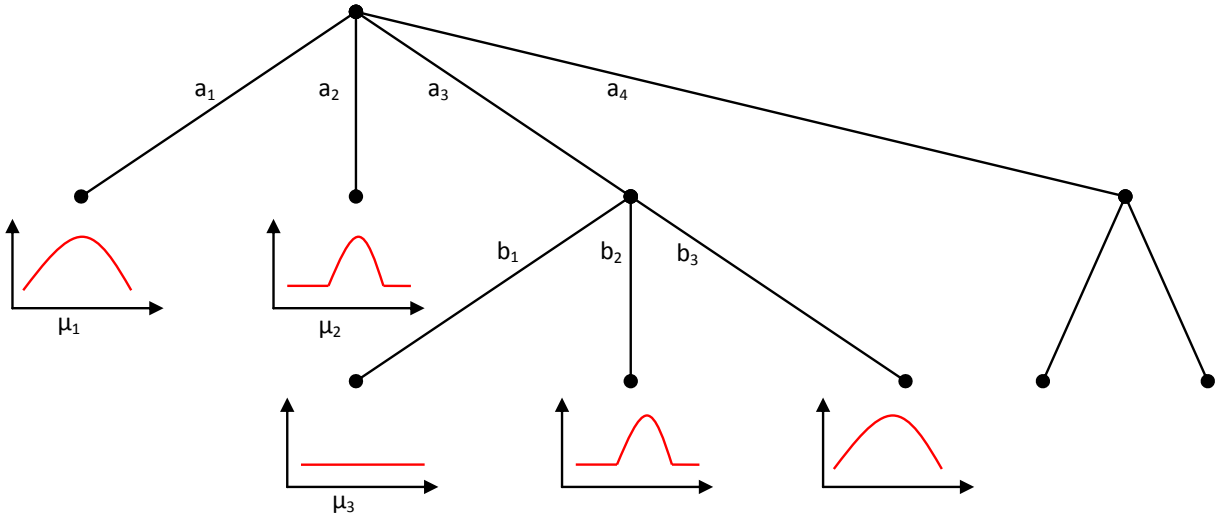


Figure 8.2: Statistical decision tree

8.6 Conclusion

In this chapter, we introduce an algorithm to discover statistical dependencies. We describe the graphic representation of statistical dependencies. In our statistical decision trees, some attributes predict the distribution of values of the target attribute with high probability.

CHAPTER 9

Implementation

In this chapter, we present the environment to test our algorithms, the programs to create data warehouses. Then, we apply our algorithms on these data bases to evaluate results of approximation and of mining of statistical dependencies.

9.1 Environment

We use MySQL for the relational data, Mondrian for OLAP engine and an improved version of JPivot where answers are graphically represented by multi-dimensional pie-charts.

9.1.1 Mondrian

9.1.1.1 Introduction

The decision-making system Mondrian [5, 17] is an OLAP engine written in Java. It allows to interactively analyze the very large datasets stored in the database management system. It reads data from a relational database and displays the results in a multidimensional format via a Java application programming interface (API).

9.1.1.2 Architecture

The Mondrian OLAP system consists of four layers [5, 17] from the end user to the data center. These layers in Figure 9.1 are: the presentation layer, the calculation layer, the aggregation layer and the storage layer.

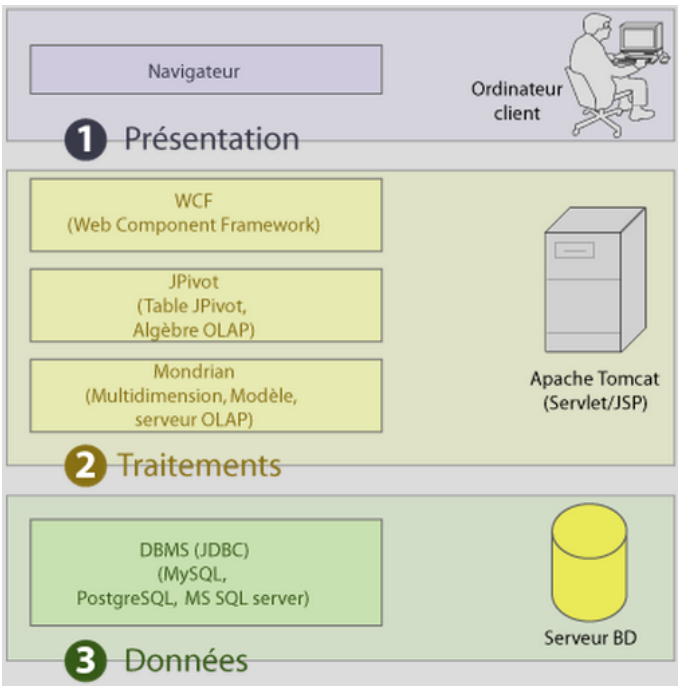


Figure 9.1: Mondrian architecture [5].

- "Presentation layer: the presentation layer determines what the user sees on the screen and how we can interact with it to make new requests. There are several ways to present sets of multidimensional data, including pivot tables, pie, line and bar charts, and advanced visualization tools such as clickable maps and dynamic graphics. They can be written in Swing or Java Script Pages (JSP), format tables outgoing Joint Photographic Experts Group (JPEG) or Graphics Interchange Fornat (GIF) or transmitted to a remote application via XML. That all these forms of presentation have in common is the "grammar" of dimensions, measures and cells in which the presentation layer asks the query and OLAP server returns an answer"[5, 17].

- "Calculation layer: the second layer parses, validates and executes MDX queries (MDX stands for 'multi-dimensional expressions'. It is the main query language implemented by Mondrian). A query is evaluated in multiple phases. The axes are computed first, then the values of the cells within the axes. For efficiency, the calculation layer sends cell-requests to the aggregation layer in batches. A query transformer allows the application to manipulate existing queries, rather than building an MDX statement from scratch for each request. And metadata describes the the dimensional model, and how it maps onto the relational model" [5, 17].

- "Aggregation layer: this layer is responsible for maintaining an aggregate cache. An aggregation is a set of measure values ('cells') in memory, qualified by a set of dimension column values. The calculation layer sends requests for sets of cells. If the requested cells are not in the cache, or derivable by rolling up an aggregation in the cache, the aggregation manager sends a request to the storage layer"[5, 17].
- "Storage layer: the storage layer is a Databases Management Systems (DBMS). It is responsible for providing aggregated cell data and members from dimension tables"[5, 17].

These components can all exist on the same machine, or can be distributed between machines. Layers 2 and 3, which comprise the Mondrian server, must be on the same machine (in Figure 9.1, layers 2 and 3 are situated in the second part of this structure). The storage layer could be on another machine, accessed via remote JDBC connection. In a multi-user system, the presentation layer would exist on each end-user's machine.

In the next sub section, we see how Mondrian displays the results of queries via the interface JPivot.

9.1.2 JPivot

9.1.2.1 Introduction

JPivot [4] is a graphic interface that allows users to perform OLAP queries and to see answers. The new version of JPivot in our implementation is improved by Tsuyoshi Sugibuchi (buchi@lri.fr). The interface adds new icons which allows users to select and upload schemas. Moreover, this interface implemented new styles of chart such as pie chart, bar chart line chart and hpie chart.

9.1.2.2 Interface

The interface JPivot is composed of a tool bar and two lists to create the queries. An answer is displayed via a data table and a chart. The Figure 9.2 provides an overview of the interface.

We can categorize all icons of the toolbar according to their functions as follows:

1. Group of icons to manage dimensions
2. Group of icons to select and upload schemas
3. Group of icons to choose modes of display of tables

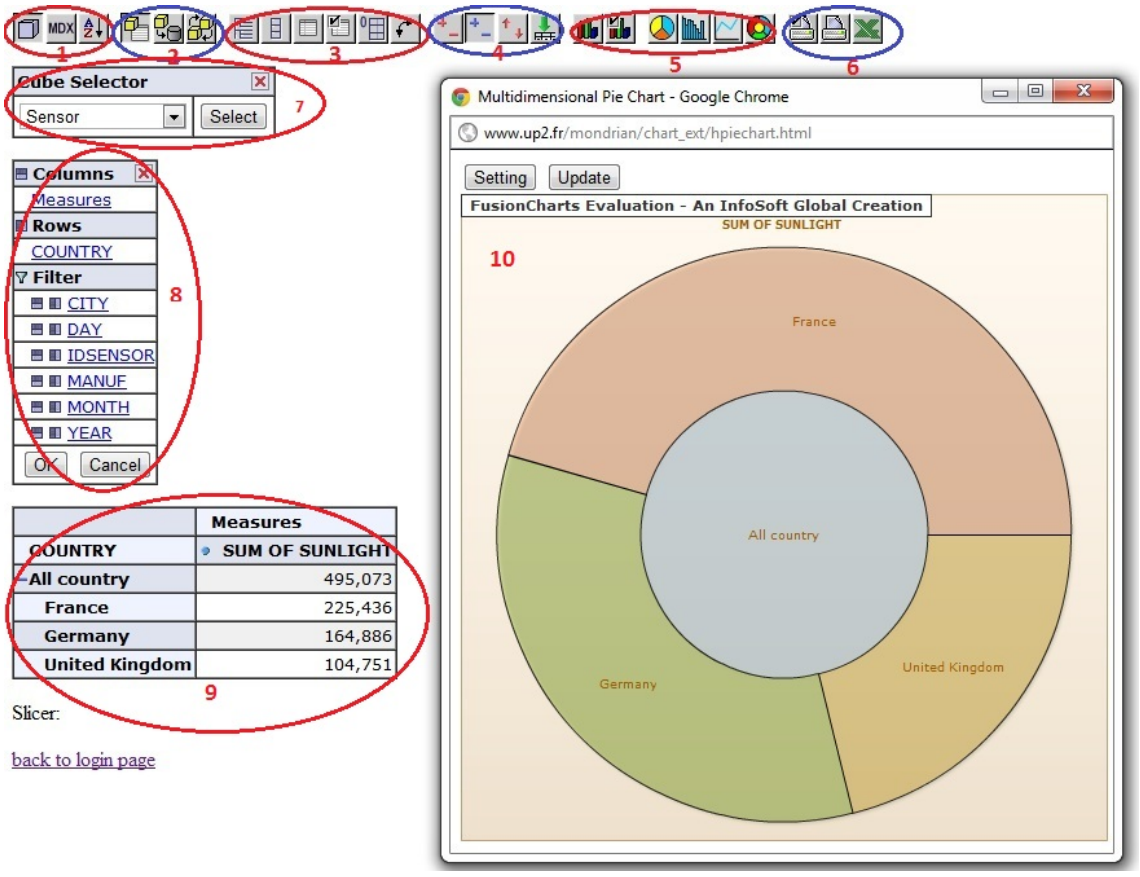


Figure 9.2: Jpivot interface.

4. Group of icons of navigation
5. Group of icons to manage charts
6. Group of icons to print or to export data
7. List to select one schema
8. List to choose measures, dimensions and filters for a query
9. Data table of query answering
10. Chart of query answering

9.1.2.3 Performance

We present the steps from creating a query to the answer of query.

- Step 1: choose an OLAP schema in the list of region 7
- Step 2: select all components for a query in region 8: measures, dimensions and filter.
- Step 3: by clicking in the button 'OK' of region 8 to fixe the query, Mondrian will return the answer by an interactive and clickable pivot table in region 9. In this pivot table, we can perform OLAP navigations. Besides of pivot table, the icons of group 5 allow to see chart of this answer. We present Figure 9.3 which is an example of the multi dimensional pie-charts available for the JPivot tool. The number of dimensions is arbitrary.

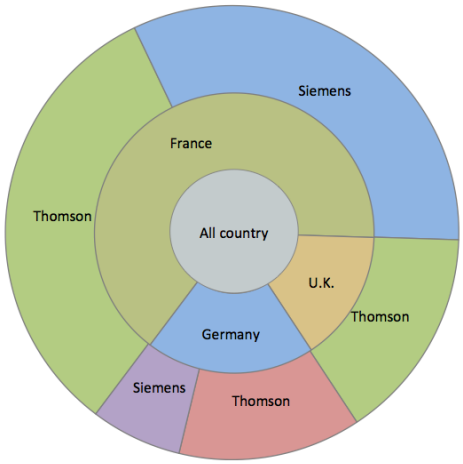


Figure 9.3: Analysis of *sun* over *country* and *manufacturer*.

To design cube schemas, we use Mondrian Schema Workbench, in the next sub section, we present its interface and its performance.

9.1.3 Mondrian Schema Workbench

9.1.3.1 Introduction

Mondrian Schema Workbench [28] helps you to create and test OLAP cube schemas visually. These cube schemas are XML metadata models that are created in a specific structure used by the Mondrian engine. These XML models can be considered cube-like structures which utilize existing FACT and DIMENSION tables found in the RDBMS. It does not require that an actual physical cube is built or maintained; only that the metadata model is created.

9.1.3.2 Interface

The interface of Mondrian Schema Workbench in Figure 9.4 provides the following functionality: set up the properties for connection to your cube database, create or edit elements (measures, dimensions, hierarchies) in the schema, display error messages or results, and saving your schema.

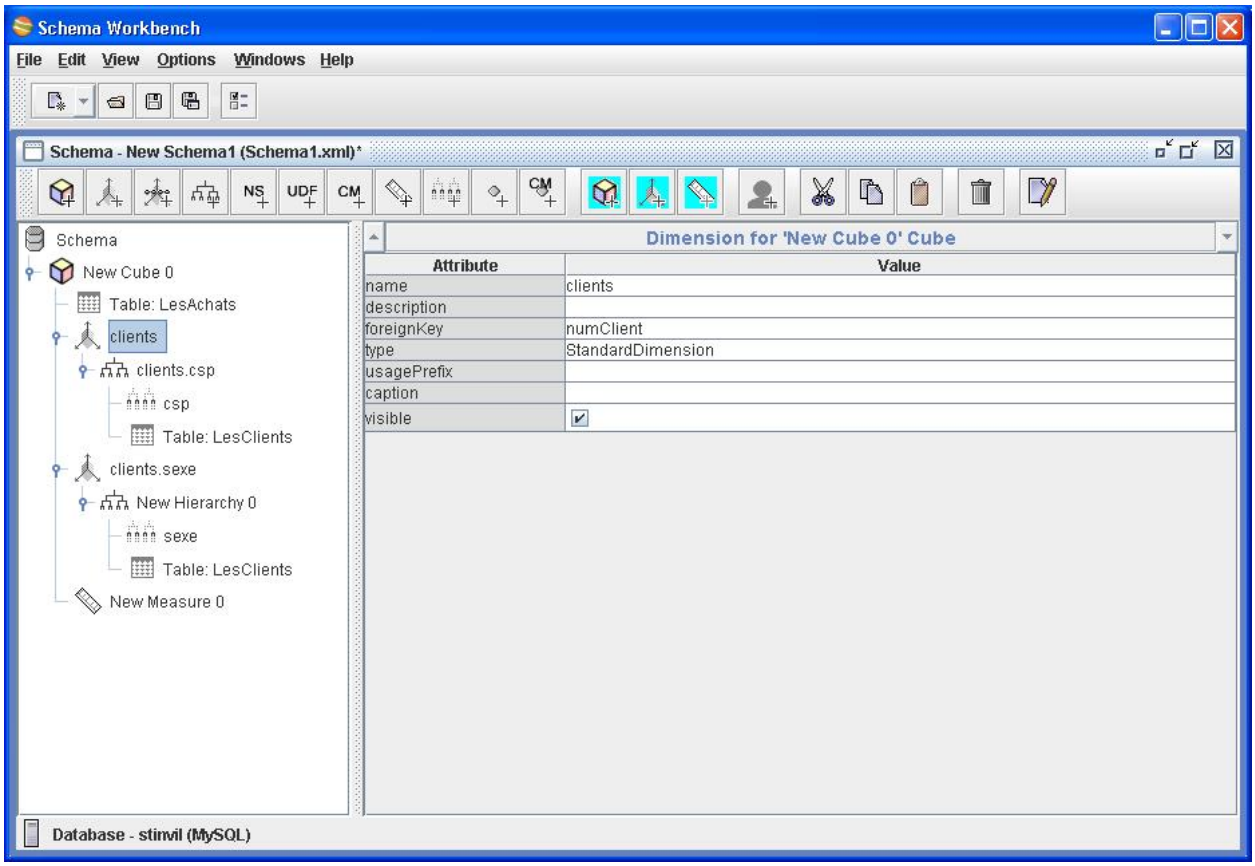


Figure 9.4: Interface of Mondrian Schema Workbench.

9.2 Sensors data warehouse

9.2.1 Data

In the next section, we have a set of experiments using two data warehouses to analyze the results. A data warehouse of sensors is simulated by our program. A data warehouse of RSS streams is collected from 4 web sites of news: www.cnn.com, www.rfi.fr, www.bbc.co.uk and www.voanews.com. We provide a summary about

these data warehouses in Table 9.1. The programs to create these data bases and to sample are coded in Java and PHP.

	Sensor data warehouse	RSS data warehouse
Number of samples	10^6	10^4
Duration of creation	5 days	3 months
Frequency of tuples	Continuous	Continuous
Source	Simulated by our algorithm	4 web sites on Internet

Table 9.1: Information of data warehouses.

We firstly consider sensors of the OLAP schema in Figure 9.5 which provide weather data, simulated by Algorithm 4 in page 78, such as hours of sun and hours of rain each day, in different cities.

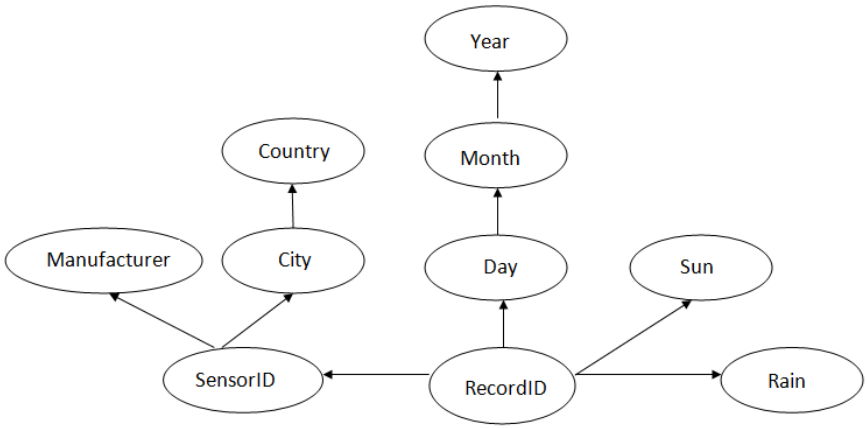


Figure 9.5: OLAP schema of sensors.

We assume an auxiliary table which provides for each sensor, its location *(city, country)*, manufacturer's name *(manuf)*, and a data warehouse DW lists the two *measures (sun, rain)* every day for each sensor, as in Example 1. For simplicity, there are only 2 manufacturers *Siemens* and *Thomson*.

In our experiment, we have 12 sensors: 6 in France, 3 in Germany and 3 in the U.K. These sources have different ratios of tuples in data warehouse. We simulate *(sun, rain)* values in the interval $[1, 10]$ with the following random generator. We associate a value $\tau_j \in [1, 10]$ for each city $j \in [1, 9]$ among the 9 cities, so that the distributions of *sun* are biased towards high values if $\tau_j \geq 5$ or low values if $\tau_j < 5$. The sensor 1 in London has $\tau_1 = 3$ and the sensor 9 in Marseille has $\tau_9 = 8$.

We set up τ_j for all cities in Table 9.3 and the location for each sensor in Table 9.2.

sensorID	manufacturer	city	country
1	Thomson	Paris	France
2	Siemens	Paris	France
3	Thomson	Lyon	France
4	Siemens	Lyon	France
5	Thomson	Marseille	France
6	Siemens	Marseille	France
7	Thomson	London	United Kingdom
8	Thomson	Birmingham	United Kingdom
9	Thomson	Manchester	United Kingdom
10	Siemens	Munich	Germany
11	Thomson	Berlin	Germany
12	Thomson	Hambourg	Germany

Table 9.2: Location of each sensor.

We describe our method of simulation of data in Algorithm 4, we simulate the data in N days. For each day, the data of 12 sensors are created. For each sensor, we select k in $[1, 10]$ with the uniform distribution (we denote $k \in_r [1, 10]$ in Algorithm 4). If the value of k is greater than τ_j, the value of *sun* is randomly selected in $[1, \tau_j]$. If the value of k is less than or equal to τ_j, the value of *sun* is randomly selected in $[\tau_j + 1, 10]$. The value of *rain* is equal to $(10 - sun)/2$.

Data: $\tau_1, \ldots, \tau_{12}, N$
Result: Table DW
`/* generate` $12 * N$ `tuples` *
for $i := 1$ **to** N **do**
  for $j := 1$ **to** 12 **do**
    $k \in_r [1, 10]$;
    `/* select` k `in` $[1, 10]$ `with uniform distribution` *
    if $k \geq \tau_j$ **then** $sun \in_r [1, \tau_j]$;
    else $sun \in_r [\tau_j + 1, 10]$;
    $rain := (10 - sun)/2$;
  end
  $date := date - 1$;
end

Algorithm 4: Generating data for DW

Notice that we satisfy the specific statistical constraints, as $city \triangleleft sun$, i.e. each city determines a fixed distribution of *sun* over the values $1, 2, ..., 10$. We give the values τ_j of cities and the distribution of *sun* on each *city* in Table 9.3. As an example, the distribution $\mu(Marseille)$ can be described as: the values $1, 2, ..., 8$ with probability $2/10$ and the values $9, 10$ with probability $8/10$.

city	τ_j	Distribution of sun on each city
Paris	6	If $k \geq 6$ then $sun \in_r [1, 6]$ else $sun \in_r [7, 10]$
Lyon	7	If $k \geq 7$ then $sun \in_r [1, 7]$ else $sun \in_r [8, 10]$
Marseille	8	If $k \geq 8$ then $sun \in_r [1, 8]$ else $sun \in_r [9, 10]$
London, Birmingham, Manchester	3	If $k \geq 3$ then $sun \in_r [1, 3]$ else $sun \in_r [4, 10]$
Munich, Berlin, Hambourg	5	If $k \geq 5$ then $sun \in_r [1, 5]$ else $sun \in_r [6, 10]$

Table 9.3: Distribution of *sun* on each *city*.

Figure 9.6 illustrates the distribution of *sun* on two cities: Marseillle and London. For Marseillle, the high values of *sun* have high probability. For London, the high values of *sun* have low probability.

9.2.2 Approximate answers for the different sampling methods

The sampling algorithms are analyzed in two cases: uniform distribution, *measure*-based. The real data usually introduce noise, for example undefined or incorrect values. It is important that these sampling methods are robust to noise.

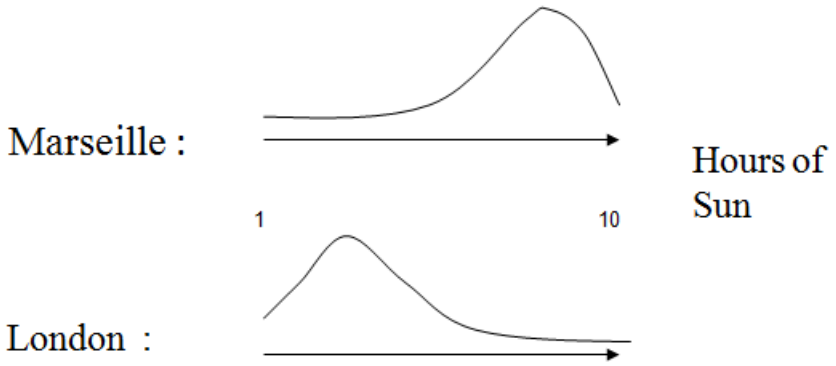


Figure 9.6: Distribution of *sun* on Marseillle and London.

We consider $Q2$: analysis of *sun* on *manufacturer* on the three sets: *Sensor* which defines the real data, *Uniform* which defines m samples with a uniform distribution, and *Measure-based* which defines m samples with the biased distribution. To visualize the quality of approximation, we observe the difference between pie charts of answers. The exact answer (Siemens, 0.3911), (Thomson, 0.6089) is represented in Figure 9.7.

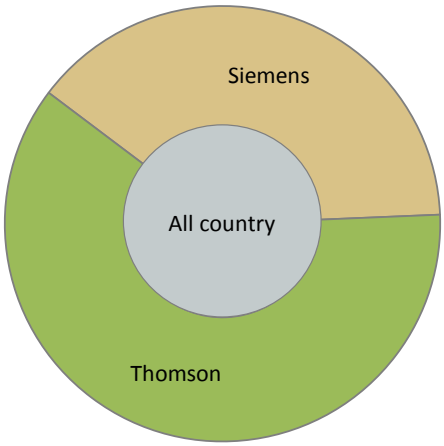
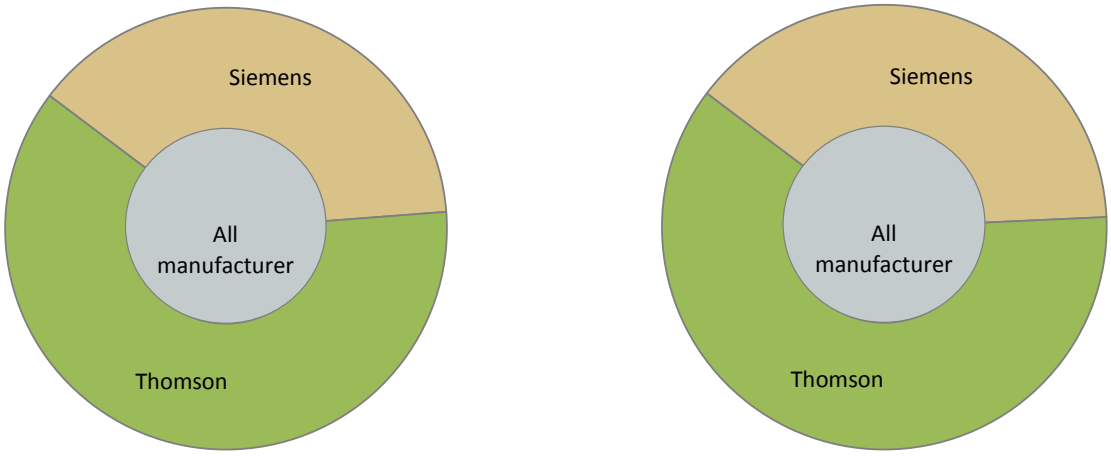


Figure 9.7: Q2: Analysis of *sun* on *manufacturer* on the set *Sensor* (Exact answer).

The two approximate answers of two sampling methods are represented by two pie charts in Figure 9.8. They show that the difference between the exact answer and the approximate answer is very small, as predicted by the theory. Notice that this holds for any query of the schema, provided the number of tuples is larger than N_0. If we apply a selection, the number of tuples after the selection may be less than N_0 and in this case the error will not be guaranteed.



(a) Approximate answer to Q2 on the set *Uniform*. The error is 1.2%.

(b) Approximate answer to Q2 on the set *Measure-based*. The error is 3.7%.

Figure 9.8: Analysis of *sun* on *manufacturer*.

9.2.3 Approximate answer from the statistical model

We denote Q^S_{manuf}, the approximate answer for the analysis of *sun* on *manufacturer* from the statistical model. From the theory discussed in chapter 5, for the *measure sun*, if there is a statistical dependency $city \triangleleft sun$, then the density of tuples over $manufacturer.city$ is enough to approximate the answer for the OLAP analysis of *sun* on *manufacturer*.

When we generated tuples for these data warehouses of sensors, we defined in advance the statistical dependency $city \triangleleft sun$ by the distribution μ. Moreover, from the distributions μ of *sun* on each *city*, we need to compute the average value of *sun* for each *city*: Avg_μ.

In our case, the data warehouse is built from different sources. The density of tuples (or the rate of each source) in the target data warehouse is different. For example, the rate of source 1 is $r_1 = 50\%$. The rate of sources 2 and 3 are $r_2 = r_3 = 25\%$. Then, Q^S_{manuf} is computed by one of three following ways

1. The first way (described in Figure 9.9): all Avg_μ and $\sigma($ *manuf. city*$)$ are estimated by a counter at the target. The target does not stock the tuples of sources. It stocks only the model of statistical dependencies computed by the counter. We assume that the counter always knows the number of tuples of the target. When there is a new tuples from the sources, the counter calculates and updates Avg_μ and $\sigma(manuf.city)$ in the target.

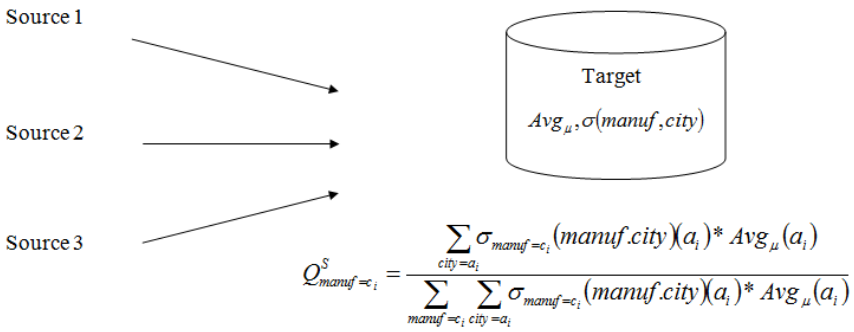


Figure 9.9: Statistical computation by the target.

2. The second way (described in Figure 9.10): the target asks each source to estimate Avg_μ and σ $(manuf. \quad city)$. Then, these informations are sent and updated in the target when we need to approximate the answers.

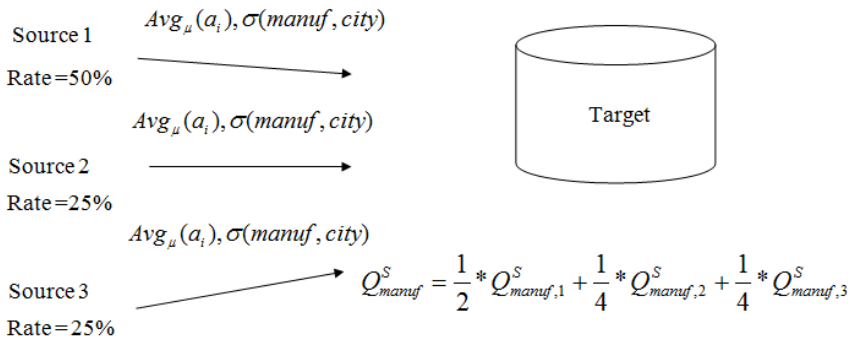


Figure 9.10: Statistical computation by the sources.

3. The third way (described in Figure 9.11): the target asks each source its answers. Then, these answers are sent the target to approximate for the global answer.

9.2.3.1 Statistical computation by the target

To estimate Avg_μ and $\sigma(manuf.city)$, we can use the sampling method with the uniform distribution on the whole data warehouse. The calculation of Avg_μ and $\sigma(manuf.city)$ is based on the generating way of tuples for these data warehouses.

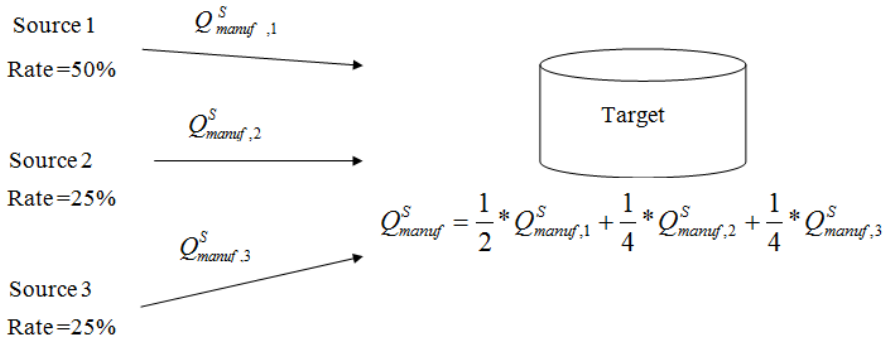


Figure 9.11: Approximate answer at each source.

1. From the distribution $\mu(Marseille)$ in Table 9.3: the values $1, 2, ..., 8$ with probability $2/10$ and the values $9, 10$ with probability $8/10$, we can calculate the average value of the *measure* with this distribution $Avg_\mu(Marseille)$ is equal to:

$$Avg_\mu(Marseille) = \frac{2}{10} * 4.5 + \frac{8}{10} * 9.5 = 8.5$$

Similarly for all the 9 cities, we provide the distribution μ and Avg_μ for each *city* in Table 9.4.

city	$Avg_\mu(city)$
Paris	$\frac{4}{10} * 3.5 + \frac{6}{10} * 8.5 = 6.5$
Lyon	$\frac{3}{10} * 4 + \frac{7}{10} * 9 = 7.5$
Marseille	$\frac{2}{10} * 4.5 + \frac{8}{10} * 9.5 = 8.5$
London, Birmingham, Manchester	$\frac{7}{10} * 2 + \frac{3}{10} * 7 = 3.5$
Munich, Berlin, Hambourg	$\frac{5}{10} * 3 + \frac{5}{10} * 8 = 5.5$

Table 9.4: Avg_μ for each city.

2. We calculate the density of tuples over $manufacturer.city$ in data warehouse: $\sigma(manuf.city)$.

In our case, the distribution $manufacturer.city$ is over 12 pairs: the probability $(Siemens, Paris)$ is $1/12$, as in Table 9.5.

3. From the informations of Table 9.4 and of Table 9.5, Q^S_{manuf} is calculated as follow:

   -The distribution of *sun* on Siemens:

$$Q^S_{manuf=Siemens} = \frac{\sum_{city=a_i} \sigma_{manuf=Siemens}(manuf.city)(a_i) * Avg_\mu(a_i)}{\sum_{manuf=c_i} \sum_{city=a_i} \sigma_{manuf=c_i}(manuf.city)(a_i) * Avg_\mu(a_i)}$$

$$= \frac{\frac{6.5+7.5+8.5+5.5}{12}}{\frac{6.5+7.5+8.5+5.5}{12} + \frac{6.5+7.5+8.5+(3.5*3)+(5.5*2)}{12}} = 0.389$$

manufacturer	city	Density of tuples
Siemens	Paris	$\frac{1}{12}$
Siemens	Lyon	$\frac{1}{12}$
Siemens	Marseille	$\frac{1}{12}$
Siemens	Munich	$\frac{1}{12}$
Thomson	Paris	$\frac{1}{12}$
Thomson	Lyon	$\frac{1}{12}$
Thomson	Marseille	$\frac{1}{12}$
Thomson	London	$\frac{1}{12}$
Thomson	Birmingham	$\frac{1}{12}$
Thomson	Manchester	$\frac{1}{12}$
Thomson	Berlin	$\frac{1}{12}$
Thomson	Hambourg	$\frac{1}{12}$

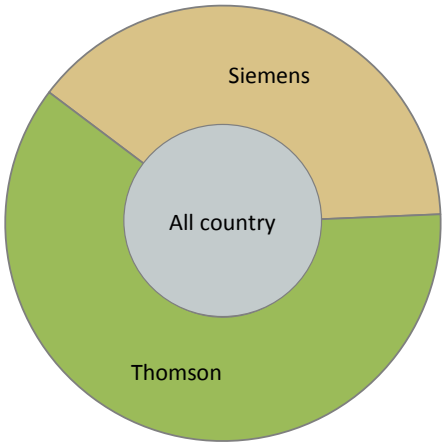
Table 9.5: Density of tuples.

- The distribution of *sun* on Thomson:

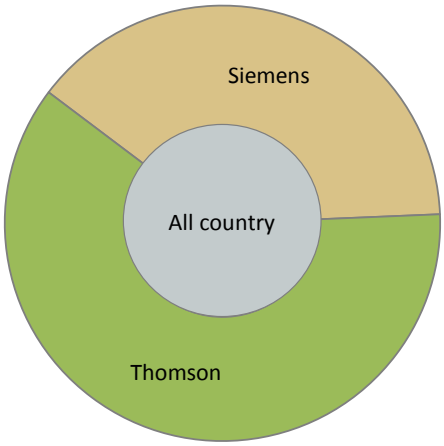
$$Q^S_{manuf=Thomson} = \frac{\sum_{city=a_i} \sigma_{manuf=Thomson}(manuf.city)(a_i) * Avg_\mu(a_i)}{\sum_{manuf=c_i} \sum_{city=a_i} \sigma_{manuf=c_i}(manuf.city)(a_i) * Avg_\mu(a_i)}$$

$$= \frac{\frac{6.5 + 7.5 + 8.5 + (3.5 * 3) + (5.5 * 2)}{12}}{\frac{6.5 + 7.5 + 8.5 + 5.5}{12} + \frac{6.5 + 7.5 + 8.5 + (3.5 * 3) + (5.5 * 2)}{12}} = 0.611$$

We can see that Q^S_C, the approximate answer from the statistical model, is close to the exact answer of Figure 9.12(a).



(a) Analysis of *sun* for each *manufacturer* on the schema *Sensor* (Exact answer).



(b) Q^S_C: Linear estimation by the data exchange. The error is 0.4%.

Figure 9.12: Exact answer and Q^S_C.

9.2.3.2 Statistical computation by the sources

In our case, the data warehouse is built from 3 sources in 3 countries: France, England and Germany. The rate of tuples of each source in the data warehouse are $m_1 = 1/2$ and $m_2 = m_3 = 1/4$. We can approximate the answer Q^S_{manuf} as follow:

Step 1: Each source i computes μ, Avg_μ and $\sigma(manuf.city)$. These informations are sent to the data warehouse.
Step 2: The data warehouse uses the informations of Avg_μ and $\sigma(manuf.city)$ to estimate $Q^S_{Siemens,i}$ and $Q^S_{Thomson,i}$ of each source.

Step 3: Because the rate of tuples of each source in the data warehouse is $m_1 = 1/2$ and $m_2 = m_3 = 1/4$, then the global approximation would be given by the formula:

$$Q^S_{manuf} = \frac{1}{2} * Q^S_{manuf,France} + \frac{1}{4} * Q^S_{manuf,England} + \frac{1}{4} * Q^S_{manuf,Germany}$$

In this case, the distribution of *sun* on Siemens is equal to:

$$Q^S_{manuf=Siemens} = \frac{1}{2} * Q^S_{Siemens,France} + \frac{1}{4} * Q^S_{Siemens,England} + \frac{1}{4} * Q^S_{Siemens,Germany}$$

The distribution of *sun* on Thomson is equal to:

$$Q^S_{manuf=Thomson} = \frac{1}{2} * Q^S_{Thomson,France} + \frac{1}{4} * Q^S_{Thomson,England} + \frac{1}{4} * Q^S_{Thomson,Germany}$$

9.2.3.3 Approximate answer at each source

If we follow the third way of approximation, at the step 2, each source not only estimates Avg_μ and $\sigma(manuf,city)$, but also computes $Q^S_{Siemens,i}$, $Q^S_{Thomson,i}$. Then, $Q^S_{Siemens,i}$, $Q^S_{Thomson,i}$ are sent to the data warehouse for the estimation of the global answer. The global answer is described in Figure 9.11.

The error analysis for the query Q2 is given by Table 9.6, for the three methods: uniform distribution, measure-based distribution and linear estimation by the data-exchange technique. This last estimation has the smallest error 0.4%.

Manufacturer	Figure 9.8(a) (Uniform sampling)	Figure 9.8(b) (Measure-based sampling)	Figure 9.12(b) (Q^S_{manuf}-Linear estimation by the data exchange)	Figure 9.7 (Exact answer)
Siemens	0.3851	0.4100	0.3890	0.3911
Thomson	0.6149	0.5900	0.6110	0.6089
TOTAL ERROR	0.0120	0.0378	0.0042	

Table 9.6: Quality of approximations on Q2.

9.2.4 Mining of statistical dependencies

In this section, we try to discover the statistical dependency $city \triangleleft sun$ on the data warehouse of sensors by just observing the distance between two distributions of *sun* on two sets of training.

For each value of *city*, we can estimate the distribution of values in $[1, 10]$ of *sun*. We firstly take 1000 tuples with the uniform distribution from the data warehouse. We then divide them in three sets having the same size: training 1, training 2 and test.

For each value a_i of *city*, we calculate the distributions of *measure sun* on training 1: $\mu_1(a_i)$, on training 2: $\mu_1(a_i)$, and on test: $\delta(a_i)$.

We then calculate d_1 and d_2, where

$$d_1 = \Sigma_i \mid \mu_1(a_i) - \mu_2(a_i) \mid_1$$

$$d_2 = \Sigma_i \mid (\mu_1(a_i) + \mu_2(a_i))/2) - \delta(a_i) \mid_1$$

We observe the values of d_1 and d_2 to analyze the error of mining. For each a_i, $\mid \mu_1(a_i) - \mu_2(a_i) \mid_1$ is in $[0, 2]$. We have the 9 values a_i for 9 cities. So, the values of d_1, d_2 are in $[0, 18]$. We replace d_1 by $d_1/18$, d_2 by $d_2/18$ to normalize the distance between $[0\%, 100\%]$.

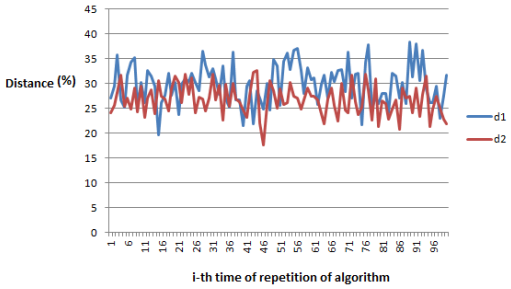
To have a better observation of results, we vary the numbers of total samples to divide in 3 sets: training 1, training 2 and test. We will have 4 graphs with 4 different numbers of total samples: 1000, 1500, 2000 and 2500.

For each graph, we fixed the number of total samples to divide in 3 subsets. We repeat the algorithm of mining with 100 times to observe the distances of distributions of the statistical dependency *city* $\triangleleft$ *sun*.

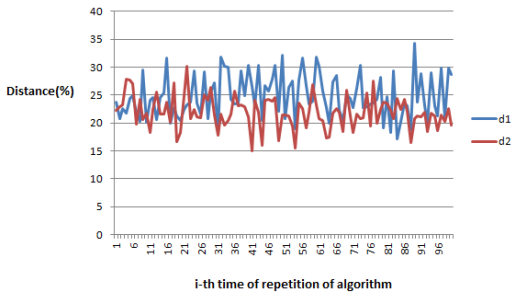
As depicted in Figure 9.13, with 1000 samples, the distance d_2 is from $17,5\%$. With 1500 samples, the distance d_2 is from 15%. With 2000 samples, the distance d_2 is from 14%. With 2500 samples, the distance d_2 is from 12%. In different cases of the total number of samples, the performance of algorithm behaves similarly. The distance d_2 is almost smaller than the distance d_1. Moreover, the larger the number of samples is, the smaller the error of mining is. With a small number of samples, we can approximate the statistical dependencies with high accuracy and high confidence.

9.3 RSS data warehouse

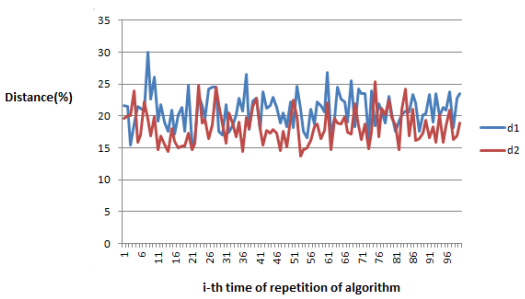
In this section, we continue with the experimental result of approximation and of mining on the data warehouse of RSS.



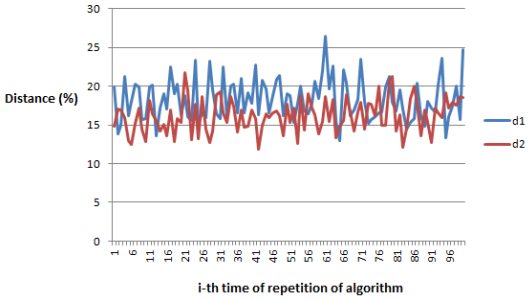
(a) Repeated Mining Algorithm with 1000 samples. The distance d_2 is from $17,5\%$.



(b) Repeated Mining Algorithm with 1500 samples. The distance d_2 is from 15%.



(c) Repeated Mining Algorithm with 2000 samples. The distance d_2 is from 14%.



(d) Repeated Mining Algorithm with 2500 samples. The distance d_2 is from 12%.

Figure 9.13: Quality of mining of $city \triangleleft sun$.

9.3.1 Context

RSS is a family of data formats based on XML and used for Web syndication. It is used for delivering the latest content from the sites. In this section, we use three PHP programs to daily collect RSS streams from four web sites www.cnn.com, www.rfi.fr, www.bbc.co.uk and www.voanews.com (Figure 9.14). Then these RSS streams are aggregated into our data warehouse 'stinvil' on the web site http://www.up2.fr/M1. You can download directly these programs from this address http://www.up2.fr/rss/download. In the next section, we analyze the experimental result of approximation and of mining on this data warehouse.

9.3.2 Data

We study another schema corresponding the collection of several sources of RSS. The OLAP schema of the RSS data warehouse is presented in Figure 9.15.

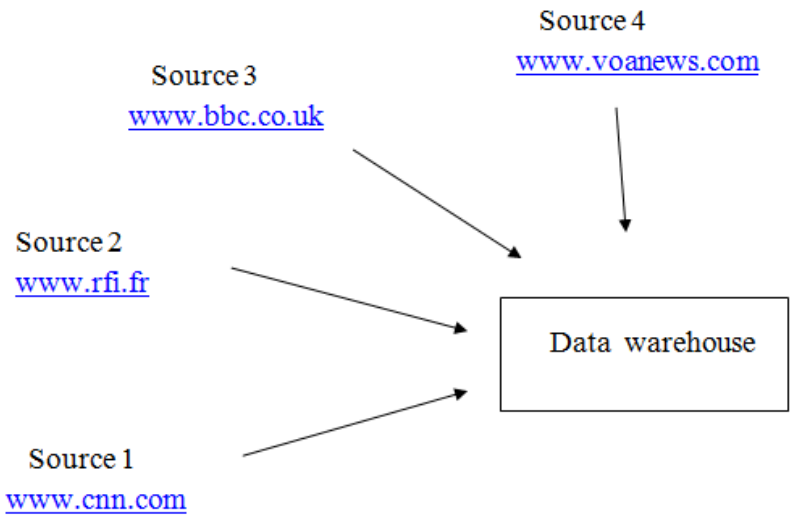


Figure 9.14: RSS streams.

The OLAP schema contains a table 'nrss' with 10 attributes : *nrss(recordID, link, day, month, year, region, domain, source, language, preference)*. In data warehouse, the measure *preference* represents the important level of information in each tuple. Its value varies between $[1, 10]$. We fixed the values of *preference* depending on two attributes *(region, domain)*. This defines the statistical dependency *(region, domain)* $\triangleleft$ *preference*. The value of *preference* depends on *region* and *domain* is illustrated in Table 9.7

9.3.3 Schemas of sets of samples

We produce distinct random sets of samples from the data base of RSS whose the OLAP schema of Figure 9.15 . The data warehouse of samples have been uploaded on the web site http://www.up2.fr/mondrian/uploader.html. You can access this page to test the query answering on these sets. We explain the significance of each set of samples:

- *rss* : the original data warehouse of RSS.
- *rssUdSamples1000*: the set of 1000 samples which are taken from the data warehouse by the uniform distribution.

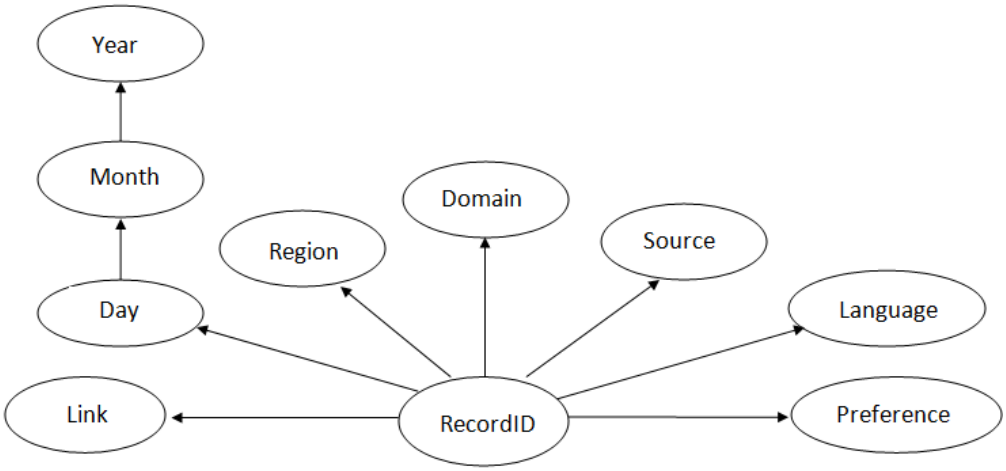


Figure 9.15: RSS OLAP schema

- *rssMeasureBasedSamples1000*: the set of 1000 samples which are taken from the data warehouse by the measure-based method on the *measure preference*.

9.3.4 Quality of approximation in the different methods

We estimate the approximate answers of different queries with the three methods: the uniform distribution sampling, the measure-based sampling and the statistical model. We calculate the distance between the approximate answers and the exact answer and compare the quality of approximation.

9.3.4.1 Analysis of the *measure preference* on *source*

We compare the query $Q4$: analysis of *preference* by *source* on the three schemas: *rss*, *rssUdSamples1000* and *rssMeasureBasedSamples1000*.

The approximate answers by using the uniform distribution sampling and the measure-based sampling are presented in Figures 9.16(a) and 9.16(b). The approximate answers from the statistical model and the exact answer are in Figures 9.16(c) and 9.16(d).

We denote Q^S_{source}, the approximate answer for the analysis of *preference* on *source* from the statistical model. There is a statistical dependency *(region, domain)* $\triangleleft$ *preference* in Table 9.7, then the density of tuples over *(source, region, domain)* is enough to approximate the answer Q^S_{source}. The density of tuples over *(source, region, domain)* is calculated by sampling the uniform distribution as in Table 9.8. So Q^S_{source} is estimated as follow:

region	domain	preference
asia	general	1
africa	general	1
americas	general	1
europe	general	1
middle east	general	1
FRANCE	business	6
FRANCE	entertainment	1
FRANCE	science	8
FRANCE	sport	1
UK	business	6
UK	entertainment	1
UK	science	8
UK	sport	1
US	business	4
US	entertainment	1
US	science	10
US	sport	1

Table 9.7: *(region, domain)* statistically imply *preference*.

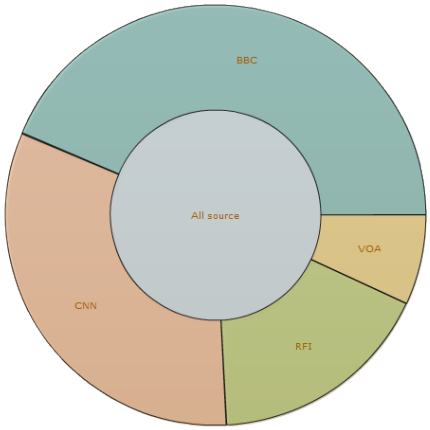
$$Q^{S}_{source=s_i} = \frac{\sum_{region=r_j,domain=d_k} \sigma_{source=s_i}(source.region.domain)(r_j,d_k) * preference(r_j,d_k)}{\sum_{source=s_i}\sum_{region=r_j,domain=d_k} \sigma_{source=s_i}(source.region.domain)(r_j,d_k) * preference_(r_j,d_k)}$$

The quality of approximation for the query $Q4$ is given as in Table 9.9, for the three methods: uniform distribution, measure-based distribution and statistical model.

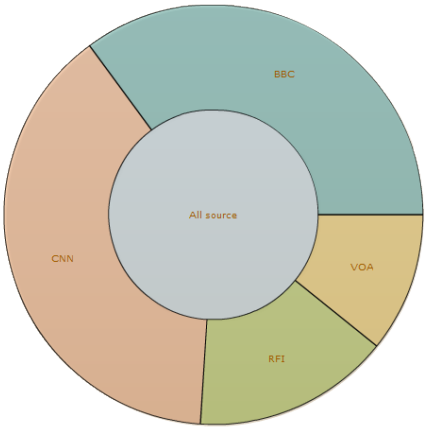
We show the answers for the query $Q5$: analysis on *domain* for *preference* on the three sets fo samples: *rss*, *rssUdSamples1000* and *rssMeasureBasedSamples1000*. Figure 9.17 illustrates the results.

The approximation for the query $Q5$ is given by Table 9.10, for the two methods: uniform distribution, measure-based distribution.

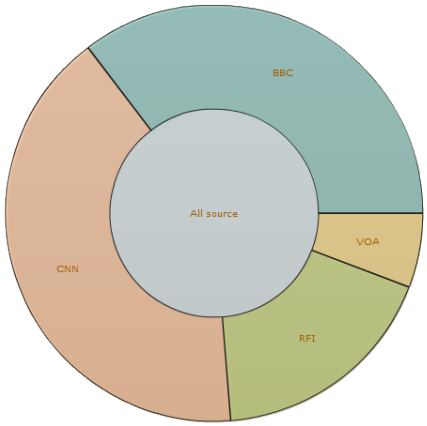
Because there are not enough the statistical dependencies, the query $Q5$ can not be estimated by the statistical model. If we want to approximate *preference* on *domain*, we need the statistical dependency *domain* $\triangleleft$ *preference*. But in the RSS



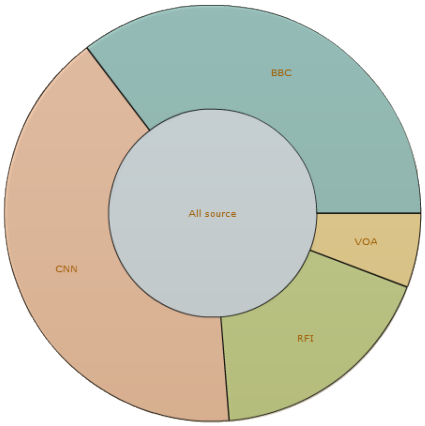
(a) Approximate answer to Q4 on schema *rssUdSamples1000*. The error is 4 %.



(b) Approximate answer to Q4 on schema *rssMeasureBasedSamples1000*. The error is 17.06%.



(c) Approximate answer to Q4 from the statistical model. The error is 7%



(d) Exact answer to Q4 on schema *rss*.

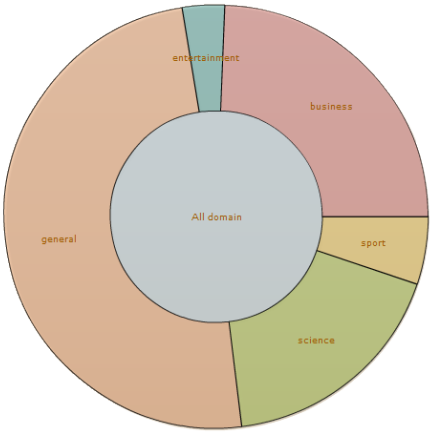
Figure 9.16: Analysis of *preference* on *source*.

data warehouse, there is not this dependency.

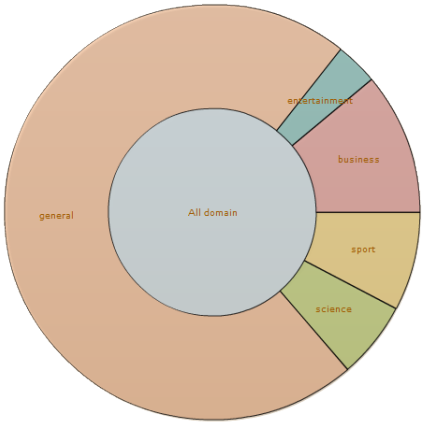
9.3.4.2 Analysis of *preference* on *region* and *domain*

We present the answers for the query *Q*6: analysis of *preference* over *region* and *domain* on the three schemas: *rss*, *rssUdSamples1000* and *rssMeasureBasedSamples1000*.

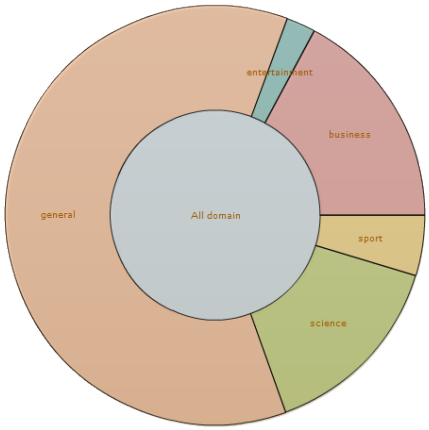
The approximate answers by using the uniform distribution sampling and the measure-based sampling are presented in Figures 9.18(a) and 9.18(b). With the different



(a) Approximate answer to Q5 on schema *rssUdSamples1000*. The error is 13%



(b) Approximate answer to Q5 on schema *rssMeasureBasedSamples1000*. The error is 10%



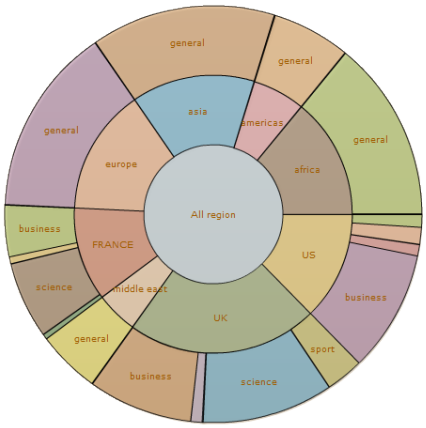
(c) Exact answer to Q5 on schema *rss*.

Figure 9.17: Analysis of *preference* on *domain*

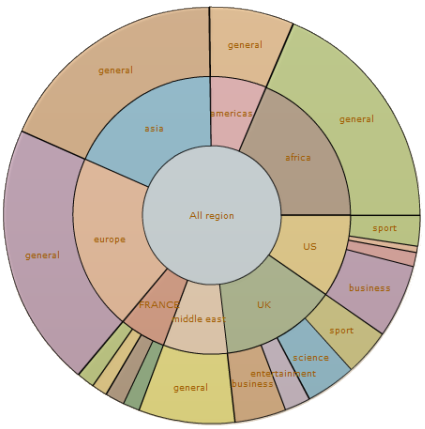
queries and with the different methods, the quality of approximation is good.

We consider $Q^S_{region,domain}$, the approximate answer for the analysis of *preference* on *region, domain* from the statistical model. The density of tuples over *(region, domain)* is calculated by sampling the uniform distribution as in Table 9.11. Then, Q^S_{source} is estimated as follow:

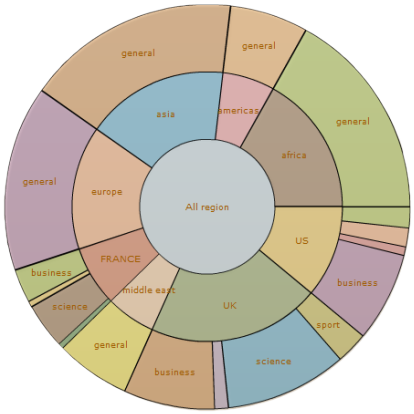
$$Q^S_{region=r_j,domain=d_k} = \frac{\sigma(region.domain)(r_j,d_k) * preference(r_j,d_k)}{\sum_{region=r_j,domain=d_k} \sigma(region.domain)(r_j,d_k) * preference_(r_j,d_k)}$$



(a) Approximate answer to Q6 on schema *rssUdSamples1000*.



(b) Approximate answer to Q6 on schema *rssMeasureBasedSamples1000*.



(c) Exact answer to Q6 on schema *rss*.

Figure 9.18: Analysis of *preference* on *region* and *domain*.

9.3.5 Mining of statistical dependencies

In this section, we show the experimental result of mining of statistical dependency *(region, domain)* $\triangleleft$ *preference* on the data warehouse of RSS. For each graph of test of performance, we fixed the number of samples of sets: training 1, training 2 and test. We also test the algorithm of mining with 100 times to observe the distances of distributions of the statistical dependency *(region, domain)* $\triangleleft$ *preference*.

In this case, for each pair of values (a_i, b_j) of *(region, domain)*, the distributions of *measure preference* on training 1: $\mu_1(a_i, b_j)$, on training 2: $\mu_1(a_i, b_j)$, and on test: $\delta(a_i, b_j)$.

We then calculate d_1 and d_2, where

$$d_1 = \Sigma_i \Sigma_j \mid \mu_1(a_i, b_j) - \mu_2(a_i, b_j) \mid_1$$

$$d_2 = \Sigma_i \Sigma_j \mid (\mu_1(a_i, b_j) + \mu_2(a_i, b_j))/2) - \delta(a_i, b_j) \mid_1$$

The values of d_1, d_2 are in $[0, 2]$. We replace d_1 by $d_1/2$, d_2 by $d_2/2$ to normalize the distance between $[0\%, 100\%]$.

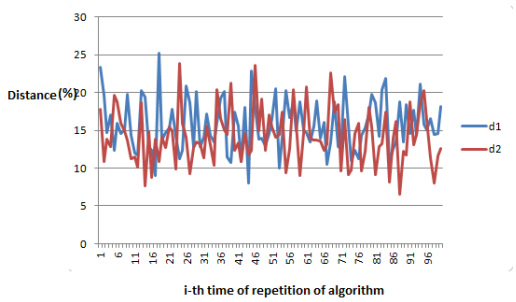
We test also with 4 different numbers of total samples: 1000, 1500, 2000 and 2500. We describe here 4 graphs obtained in Figure 9.19. The result is similar to the result on the data warehouse of sensors. The distance d_2 is almost smaller than the distance d_1. With a larger number of samples, we get the smaller error. The ratio of error is quite small. It is only about 10%.

Moreover, we can see that with the same number of samples, the error of mining on the data warehouse of RSS is much smaller than the error of mining on the data warehouse of sensors.

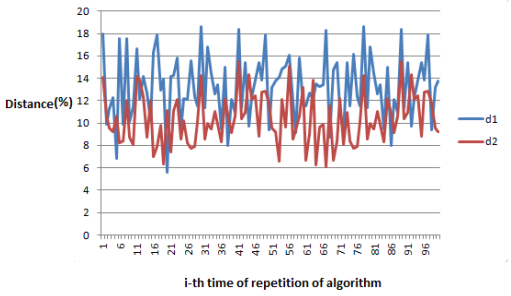
The cause of this difference is the number of elements in each distribution vector σ. With sensors, on each set of training, we have 9 distributions which correspond to 9 cities. Each distribution vector has 10 elements which correspond to 10 values of *sun*. We can consider that d_1 is the sum of distance between 90 elements. d_2 is similar to d_1. With RSS streams, d_1 is the sum of distances between 17 elements. So, the quality of approximation with RSS is better than in data warehouse of sensors.

9.4 Conclusion

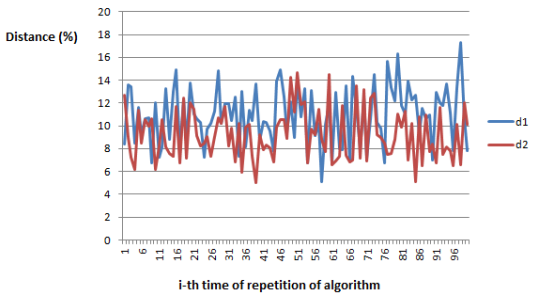
In this chapter, we used a data warehouse simulates sensors and a real data warehouse collecting RSS from the web sites on Internet. We tested our approximation methods: the approximation by uniform sampling, the approximation by *measure*-based sampling and by the statistical model. These algorithms have the good quality of approximation. They guarantee a small error with a high confidence.



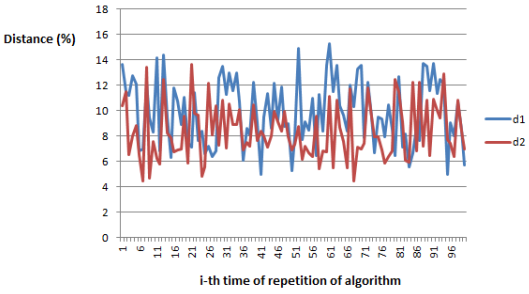
(a) Repeated Mining Algorithm with 1000 samples. The distance d_2 is from $7,5\%$.



(b) Repeated Mining Algorithm with 1500 samples. The distance d_2 is from 6%.



(c) Repeated Mining Algorithm with 2000 samples. The distance d_2 is from 5%.



(d) Repeated Mining Algorithm with 2500 samples. The distance d_2 is from 4.5%.

Figure 9.19: Quality of mining of *(region, domain)* $\triangleleft$*preference*.

source	region	domain	density of tuples
BBC	africa	general	0.0515
BBC	americas	general	0.0355
BBC	asia	general	0.0660
BBC	europe	general	0.0510
BBC	middle east	general	0.0185
BBC	UK	business	0.0200
BBC	UK	entertainment	0.0180
BBC	UK	science	0.0200
BBC	UK	sport	0.0420
CNN	africa	general	0.0745
CNN	americas	general	0.0295
CNN	asia	general	0.1040
CNN	europe	general	0.1125
CNN	middle east	general	0.0180
CNN	US	business	0.0295
CNN	US	entertainment	0.0115
CNN	US	science	0.0025
CNN	US	sport	0.0280
RFI	africa	general	0.0650
RFI	americas	general	0.0125
RFI	asia	general	0.0135
RFI	europe	general	0.0250
RFI	middle east	general	0.0180
RFI	FRANCE	business	0.0075
RFI	FRANCE	entertainment	0.0075
RFI	FRANCE	science	0.0075
RFI	FRANCE	sport	0.0075
VOA	africa	general	0.0250
VOA	americas	general	0.0135
VOA	asia	general	0.0250

Table 9.8: Density of tuples over *(source, region, domain)*.

Source	Figure 9.16(a) (Uniform sampling)	Figure 9.16(b) (Measure-based sampling)	Figure 9.16(c) (Statistical model)	Figure 9.16(d) (Exact answer)
BBC	0.4362	0.351	0.381	0.4143
CCN	0.3221	0.389	0.352	0.3356
RFI	0.1724	0.152	0.168	0.1738
VOA	0.0693	0.108	0.099	0.0763
TOTAL ERROR	0.0438	0.1706	0.0782	

Table 9.9: Quality of approximations on Q4.

Domain	Figure 9.17(a) (Uniform sampling)	Figure 9.17(b) (Measure-based sampling)	Figure 9.17(c) (Exact answer)
General	0.5421	0.714	0.6915
Business	0.2517	0.111	0.1716
Entertainment	0.0236	0.043	0.0224
Science	0.1753	0.059	0.1486
Sport	0.0431	0.073	0.0470
TOTAL ERROR	0.07965	0.1046	

Table 9.10: Quality of approximations on Q5.

region	domain	density of tuples
africa	general	0.2160
americas	general	0.0910
asia	general	0.2085
europe	general	0.2100
middle east	general	0.0730
FRANCE	business	0.0075
FRANCE	entertainment	0.0075
FRANCE	science	0.0075
FRANCE	sport	0.0075
UK	business	0.0200
UK	entertainment	0.0180
UK	science	0.0200
UK	sport	0.0420
US	business	0.0295
US	entertainment	0.0115
US	science	0.0025
US	sport	0.0280

Table 9.11: Density of tuples over *(region, domain)*.

CHAPTER 10

Conclusion and Future Work

We studied the approximation of OLAP queries, mostly when the aggregation operator is the Sum or the Average. Some of the results generalize to other operators. We presented two families of techniques:

- Sampling based methods: the uniform and the measure-based sampling which assume an existing data warehouse,
- Statistics based methods: when some statistical dependencies exist, we may rely on some finite statistics, and we don't assume any data warehouse.

The first technique is adapted to existing data warehouses, whereas the second technique is adapted to streaming data which do not store the entire data. These two situations seem typical of massive data, and point to different solutions.

The notion of a *statistical dependency* is inspired by graphs limits [6, 22] and applied to hypergraphs. It captures some statistical information which tends to a limit when the data warehouse grows and its size tends to infinity. In this case, we don't have to keep this statistics, and if we combine it with some other statistics which we estimate and store, we can extract approximate OLAP analysis, on any dimensions.

Besides the introduction of theory, we evaluated the algorithms by experiments. Our algorithms satisfy the important conditions. They guarantee a good accuracy and a high confidence and also have the small complexity of time and of memory space.

Our results point to the following research areas when classical dependencies can be generalized as statistical dependencies in the context of massive data. If we estimate some specific statistics on these data, which queries can we infer by combining the dependencies and the specific statistics? These approximate randomized techniques provide robust methods, insensitive to noise, to solve problems such as OLAP queries. We need to delimitate the class of queries which can be approximated, given specific statistical dependencies

CHAPTER 11

Approximation of OLAP queries on data warehouses SYNTHESE EN FRANCAISE

Nous étudions les réponses proches à des requêtes OLAP sur les entrepôts de données. Nous considérons les réponses relatives aux requêtes OLAP sur un schéma, comme les distributions avec la distance L1 et rapprocher les réponses sans stocker totalement l'entrepôt de données. Nous présentons d'abord trois méthodes spécifiques: l'échantillonnage uniforme, l'échantillonnage basé sur la mesure et le modèle statistique. Nous introduisons également une distance d'édition entre les entrepôts de données avec des opérations d'édition adaptées aux entrepôts de données. Puis, dans l'échange de données OLAP, nous étudions comment échantillonner chaque source et combiner les échantillons pour rapprocher toutes requêtes OLAP. Nous examinons ensuite un contexte streaming, où un entrepôt de données est construit par les flux de différentes sources. Nous montrons une borne inférieure de la taille de la mémoire nécessaire aux requêtes approximatives. Dans ce cas, nous avons les réponses pour les requêtes OLAP avec une mémoire finie. Nous décrivons également une méthode pour découvrir les dépendances statistique, une nouvelle notion que nous introduisons. Nous recherchons ces dépendances en basant sur l'arbre de décision. Nous appliquons la méthode à deux entrepôts de données. Le premier simule les données de capteurs, qui fournissent des paramètres météorologiques au fil du temps et de l'emplacement à partir de différentes sources. Le deuxième est la collecte de RSS à partir des sites web sur Internet.

I. INTRODUCTION Les entrepôts de données recueillent l'histoire de nombreux processus physiques, tels que les ventes d'articles, la mesure de capteurs, le trafic de données sur un réseau. Lorsque ces données sont recueillies sous une forme relationnelle, son analyse fait l'objet de traitement en ligne (OLAP). Un schéma OLAP corrige un ensemble de dépendances fonctionnelles entre les attributs, et définit les dimensions possibles. La réponse aux requêtes OLAP peut être considérée comme distributions, comme la répartition des ventes par pays ou la répartition de capteurs de mesure par ville. Dans la pratique, des réponses approximatives à des requêtes OLAP peut être suffisant, et pourrait être obtenue beaucoup plus efficace que des réponses exactes. La théorie de la complexité algorithmique étudie le compromis entre rapprochement et d'efficacité. Dans de nombreux cas, nous pouvons utiliser des algorithmes probabilistes qui permettent d'atteindre un epsilon rapprochement avec un degré de confiance élevé 1 - delta et sont beaucoup plus efficaces que les algorithmes déterministes. Dans le cas des grands entrepôts de données, nous exposons ces algorithmes pour le rapprochement des requêtes OLAP. Nous considérons deux méthodes d'échantillonnage: un échantillonnage uniforme et un échantillonnage mesure fondée sur qui les deux donnent de bonnes approximations. Les entrepôts de données sont construites par la collecte de données provenant de différentes sources et de les assembler. La théorie de la Data Exchange étudie comment décider efficacement les contraintes de cohérence, compte tenu des Sources. Nous étendons cette approche aux requêtes OLAP et demandons si nous pouvons déguster directement les sources, de collecter des données statistiques, et environ répondre aux requêtes OLAP à partir des statistiques, c'est à dire sans stocker l'ensemble des données. Des questions similaires concernent le modèle streaming, c'est à dire lorsque chaque source les flux de données vers un entrepôt de données. Pourrions-nous remplacer l'ensemble de l'entrepôt de données large par d'autres données en utilisant l'espace logarithmique? Nous répondons à ces deux questions, en généralisant la méthode d'échantillonnage. Les principaux résultats de la préoccupation de thèse le rapprochement des requêtes OLAP dans un cadre d'échange de données, et dans le cas particulier de statistique dépendances, une notion que nous introduisons. Il généralise la notion des dépendances fonctionnelles, comme certains attributs peuvent impliquer des distributions fixes sur la mesure. Dans ce cas, nous pouvons réduire les données à certaines statistiques limitées. Nous étudions également comment découvrir ces dépendances en généralisant la construction d'arbres de décision dans l'exploration de données.

II. ETAT DE L'ART La plupart des systèmes de base de données ont des modules qui offrent la possibilité d'analyse OLAP. Certains provenaient du modèle relationnel (Oracle, DB2, MS SQL / Server) et plus tard inclus des modules pour l'analyse, alors

que certains autres systèmes (SAS) ont commencé avec l'analyse et plus tard inclus le modèle relationnel classique. Dans ces systèmes, les données sont bien structurées dans un entrepôt de données et le schéma OLAP sont construits et modifiés. Les requêtes sont spécifiées par les filtres, les dimensions et les opérateurs d'agrégation, et les résultats doivent avoir de bonnes représentations graphiques. Les tableaux de bord intègrent les sorties graphiques. Pour les logiciels open source, Pentaho est un des systèmes les plus complets qui intègrent des modules pour ETL (Extract, Transform, Load), la génération de schémas et de l'interface graphique (JPivot). Nous allons utiliser ce système dans notre mise en œuvre. Outils OLAP pour les données semi-structurées sont moins développés, et l'analyse de données massives reste un domaine de recherche. Les systèmes classiques de maintenir un entrepôt de données physique qui peut être extrêmement large. Nous proposons des solutions approximatives qui peuvent s'appliquer à la technologie actuelle ou pouvons extrapoler à de nouveaux systèmes. Dans le premier cas, nous avons des échantillons de l'entrepôt de données et d'obtenir des réponses approximatives de manière plus efficace. Dans le deuxième cas, nous proposons un cadre général, où l'on peut remplacer l'entrepôt de données volumineux par quelques statistiques et de toujours garder l'analyse approximative OLAP. Cela peut s'appliquer à des données massives ou en streaming. La question générale que nous demandons, c'est: pouvons-nous remplacer ces données par quelques statistiques? Dans le cas de l'analyse OLAP, lorsque les attributs satisfont certaines hypothèses statistiques (indépendance), nous répondons positivement.

Cette thèse est structurée comme suit: Le Chapitre 1 donne un bref aperçu sur le contexte de notre recherche. Nous présentons l'objectif et les principaux résultats.
Dans le Chapitre 2, nous passons en revue les notions principales dans le système OLAP: l'entrepôt de données et le schéma. En outre, nous décrivons les composantes d'une requête et la définition de réponses relative que nous utilisons. Dans ce chapitre, nos nouvelles méthodes approximatives sont brièvement présentées. Ces méthodes sont basées sur les techniques d'échantillonnage. La qualité de nos algorithmes d'échantillonnage est garantie par l'utilisation les inégalités de probabilité de Hoeffding-Chernoff pour la somme de variables aléatoires bornées. A la fin de ce chapitre, nous présentons les différents contextes dans lesquels nous étudions les algorithmes d'approximation.
Nos principales contributions figurent dans le chapitre 3, Chapitre 4, Chapitre 5, Chapitre 6, chapitres 7 et 8. Le Chapitre 3 présente deux méthodes spécifiques: l'échantillonnage uniforme et l'échantillonnage en basant sur la mesure. Ils utilisent deux espaces probabilistes de rapprocher les réponses à la requête OLAP. Ces deux méthodes produisent un entrepôt de données beaucoup plus petit sur lequel nous pouvons requêtes approximatives OLAP. Nous allons prouver que ces deux algorithmes peuvent rapprocher les réponses avec une bonne précision epsilon de et avec

une haute probabilité 1 - delta. Dans le Chapitre 4, nous étudions une distance d'édition entre les entrepôts de données. Nous introduisons les opérations d'édition adaptés pour les entrepôts de données. Par ailleurs, nous définissons la distance entre deux entrepôts de données. Nous montrons que cette distance d'édition est une métrique, c'est à dire qu'il est toujours en [0,1], symétrique et a la propriété de l'inégalité triangulaire. La théorème de continuité indique que les proches des entrepôts de données impliquent que les réponses aux requêtes OLAP doivent également être proche. Dans le Chapitre 5, nous présentons le modèle statistique basé sur la dépendance statistique. Le noyau de cette méthode est la réduction de l'entrepôt de données en une structure plus compacte: quelques statistiques limitées. Ensuite, nous combinons ces statistiques limitées et les dépendances statistiques pour approcher la requête OLAP. Enfin, nous montrons comment apprendre les statistiques du modèle. Le Chapitre 6 couvre le rapprochement dans le cadre de l'échange de données OLAP. Dans ce contexte, nous présentons les différents algorithmes approximatifs: le rapprochement avec la distribution uniforme, le rapprochement avec la mesure fondée sur l'échantillonnage, et le rapprochement par le modèle statistique. Dans le Chapitre 7, nous considérons le streaming de données. Nous montrons tout d'abord quelques bornes inférieures sur la complexité de l'espace pour les domaines non bornés. Pour les domaines non bornés, nous concevons les compteurs pour la distribution exacte utilisée par le modèle statistique. Au Chapitre 8, nous décrivons une méthode pour découvrir les dépendances statistiques. Nous ne pouvons pas connaître ces dépendances. Nous généralisons la construction d'arbres de décision dans l'exploration de données. Dans un arbre de décision classique, certains attributs de prédire la valeur de l'attribut de M. cible Cette valeur est prédit avec une haute probabilité. Dans le cas de la dépendance statistique, nous avons construit l'arbre de décision dans laquelle certains attributs de prédire la distribution des valeurs de l'attribut cible avec une probabilité élevée. Dans le Chapitre 9, nous testons nos méthodes d'approximation: l'approximation par échantillonnage uniforme, l'approximation par échantillonnage mesure fondée et par le modèle statistique. Nous utilisons un entrepôt de données simule des capteurs et un véritable entrepôt de collecte de données RSS à partir des sites web sur Internet. Pour chaque méthode, nous nous intéressons à la qualité de l'approximation comme le taux d'erreur et le niveau confiant. A la fin, nous analysons les résultats et compare les méthodes. Dans le Chapitre 10, nous présentons la conclusion et les travaux futurs.

III. PRELIMINAIRES Ce chapitre présente les notions de base, des définitions et des résultats qui sont utilisés dans cette thèse. Nous décrivons les notations pour OLAP (traitement analytique en ligne) tels que des schémas, des requêtes, répondre à la requête. Nous présentons ensuite le modèle d'approximation en utilisant principalement la technique d'échantillonnage. Enfin, ces méthodes seront utilisées

dans le cadre de l'échange de données, de données distribuées et des données en continu.

III.1. OLAP Le traitement analytique en ligne ou OLAP pour faire court est la principale activité menée par les analystes et les décideurs. Les applications sont largement utilisées pour aider à la gestion de tableaux de bord, qui assemblent des représentations graphiques des requêtes OLAP. Nous présentons les notions et les définitions des schémas, des entrepôts de données, des requêtes et des réponses aux requêtes.

III.2 Schémas Le schéma OLAP est un arbre. La racine est recordID est l'ensemble de tous les attributs de l'entrepôt de données. De la racine, les nœuds en profondeur 1 sont les dimensions et les mesures. Une arête existe s'il y a une dépendance fonctionnelle entre les noeuds.

III.3 Requêtes OLAP La requête OLAP est une requête d'analyse. Pour exemple, analyser l La Somme du nombre de l'heure de Soleil sur Pays. La mesure est l'heure de Soleil est la mesure. La dimension est Pays. La somme est l'opérateur d'agrégation.

III.4 Réponses relatives D'autres auteurs considèrent la réponse absolue. J'étudie la réponse relative ou la distribution sous une forme de pie charte. Il y a des pie chart du 1 au n dimensions.

IV. ECHANTILLONNAGE DE L'ENTREPOT DE DONNEES

IV.1 Echantillonnage avec la distribution uniforme Dans ce cas, nous sélectionnons m échantillons distincts de l'entrepôt de données I, avec une distribution uniforme sur les N tuples. Pour mettre en œuvre la technique, nous utilisons un générateur de nombres aléatoires standard. Une fonction aléatoire(N) génère un élément i de 1 à N avec une probabilité 1/N. Il est une contrainte pour le nombre d'échantillons m, afin d'avoir un (epsilon, delta)-approximation. Nous allons montre que m ne dépend que de les paramètres epsilon et delta, mais pas sur N.

IV.2 Echantillonnage avec la distribution en basant sur la mesure Cette technique d'échantillon de l'entrepôt de données pour avoir m tuples distincts. Mais la probabilité de sélection de chaque tuple dépend de sa mesure. De plus, l'algorithme s'exécute en deux étapes: Nous sélectionnons d'abord un tuple t avec une distribution uniforme. Ensuite, nous gardons t avec une probabilité proportionnelle à sa mesure. Lorsque nous générons les échantillons, nous remplaçons la mesure de 1. Nous supposons que max est la valeur maximale de la mesure. Si max est faible par rapport à la taille de l'entrepôt de données, les deux techniques permettent de requêtes approximatives OLAP. Mais si max est grand (sans limite), seul le prélèvement mesure fondée peut être appliqué sur notre sens.

IV.3 Comparaisons Le contexte de l'application et la comparaison entre les deux

techniques sont introduits également

V. DISTANCE D'EDITION Nous étudie la distance d'édition pour les entrepôts de données. Nous décrivons les opérations d'édition: la suppression, l'insertion et la modification de tuple. Par ailleurs, nous définissons la distance entre deux entrepôts de données. Nous prouvons que cette distance d'édition est une métrique, c'est à dire qu'il est toujours dans [0,1], symétrique et a la propriété de l'inégalité triangulaire. Nous prouvons alors le théorème de continuité. La distance d'édition, nous avons introduit est adapté aux entrepôts de données, car elle garantit que près des entrepôts de données impliquent que les réponses aux requêtes OLAP doivent également être à proximité.

VI. MODELE STATISTIQUE Dans le modèle relationnel, les dépendances classiques, comme les dépendances fonctionnelles jouent un rôle important. Pour les entrepôts de données, il ya quelques autres dépendances importantes, en particulier statistique dépendances, que nous présentons. Dans ce cas, certains attributs impliquer distributions fixes de la mesure, et on peut alors rapprocher d'un entrepôt de données par un certain ensemble fixe de distributions. Dans ce contexte, nous étudions une autre méthode d'approximation pour répondre à la requête. C'est le modèle statistique. Dans notre modèle, un ensemble d'attributs détermine distributions fixes de la mesure M avec une forte probabilité. Dans cette section, la notion de dépendance statistique est présentée. Ensuite, nous montrons que la requête OLAP répondeur peut être obtenue à partir du modèle statistique. Enfin, nous montrons comment apprendre les distributions de dépendances statistiques du modèle.

VI.1 Relation entre le modèle statistique et l'approximation aux requêtes OLAP Nous utilisons le modèle statistique à rapprocher les réponses aux requêtes OLAP. Nous montrons que pour une mesure M, s'il y a un attribut A tel que A implique M, alors la distribution de plus de CA est suffisant pour environ une requête OLAP sur la dimension C. L'avantage de cette technique est que la structure compacte du modèle statistique rend l'analyse plus simple.

VI.2 Avantage Nous venons de présenter une nouvelle méthode approximative pour OLAP requête répondeur. Avec le modèle statistique, de trouver les réponses proches, nous avons besoin que les informations sur les dépendances statistiques. Si les dépendances statistiques existent dans l'entrepôt de données, nous avons montré également comment rapprocher les distributions de dépendances statistiques et la distribution de certains attributs. L'avantage de cette méthode est la structure compacte. Il ne nécessite que peu d'espace et peu de temps pour se rapprocher. Il rend l'analyse plus simple d'approximation.

VII. ECHANGE DE DONNEE OLAP Dans le cadre de l'échange de données OLAP dans la figure ci-dessous, nous considérons la situation où k différentes sources

alimentent un entrepôt de données I. Par exemple, la relation I^1 de la source S_1 alimente les données de l'Angleterre, la relation I^2 de la source S_2 alimente les données de la France, etc. Nous voulons sélectionner Îi en mi échantillons provenant de chaque source et de définir $\hat{I}^e = I^1 + I^2 + .. + I^k$ où chaque I^i suit une distribution uniforme. Nous demandons qui mi garantie que toute requête OLAP Q sur I sera bien approchée par I^e. Nous considérons d'abord la distribution uniforme, la mesure basée sur la distribution et le modèle statistique.

VIII. FLUX DE DONNEES

VIII.1 Contexte Nous considérons maintenant la construction d'entrepôts de données. D'où viennent les données proviennent-elles? Dans le cadre de l'échange de données, plusieurs sources envoient leurs tuples à une base de données cible. Ils peuvent être modifiés par la cible et finiront dans l'entrepôt de données. Ce processus est appelé ETL (Extract, Transform, Load) et est bien automatisé. Il existe de nombreuses applications où les sources envoient en permanence leurs tuples à divers clients. Les capteurs peuvent envoyer leurs données régulièrement à un site central et des nouvelles sources (BBC, CNN, ..) d'envoi des flux RSS à leurs abonnés dès que de nouvelles informations se fait sentir. Dans les deux cas, ces sources envoient des données XML, ce qui peut être facilement transformé en un tuple d'une relation. Nous considérons ces tuples sous forme de flux, qui peuvent être stockées dans un entrepôt de données. Dans le modèle de streaming, des flux de données en continu et l'une des questions principales est de savoir si nous avons besoin de les stocker ou si nous pouvons le remplacer par une mémoire beaucoup plus petit. En termes précis, nous pouvons remplacer les données de taille $O(n)$ par quelques autres données de taille $O((\ logn)^k)$, c'est à dire de taille polylogarithmique afin de répondre à des questions spécifiques? Dans notre situation, nous voulons répondre à environ requêtes OLAP. Nous considérons d'abord une borne inférieure, obtenu directement à partir de la complexité de la communication, puis procéder à des solutions approchées, d'abord avec les blocs du flux et ensuite à un procédé d'apprentissage, lorsque les données suivent un modèle statistique. Le flux est la séquence de tuples $t_1, ..., t_n$ de l'entrepôt de données I. $t_i = (i, s1, 3, 12, 2010, 7, 2)$ précisant que le capteur $s1$ mesures 7 heures d'ensoleillement et 2 heures de pluie sur Décembre 32010. Les tables auxiliaires tel que $C(ville, pays)$ sont fixes et indépendants du flux. Dans ce chapitre, nous présentons des bornes inférieures sur la complexité de l'espace pour les domaines non bornés et pour les domaines bornés. Théorème 1 : Le rapprochement de la requête OLAP sur la dimension A nécessite une mémoire $O(n)$, soit proportionnelle à la longueur du flux.

IX. DECOUVRIR DES DEPENDANCES STATISTIQUES Dans ce chapitre, nous décrivons une méthode pour découvrir les dépendances statistiques. En général, nous ne pouvons pas connaître ces dépendances. Nous recherchons pour eux en fonction

de l'arbre de décision. Les dépendances statistiques généralisent les dépendances fonctionnelles classiques comme nous l'expliquons dans le chapitre 5. Dans une dépendance statistique, un ensemble d'attributs détermine les distributions fixes d'une mesure, avec une forte probabilité. Les arbres de décision constituent une dépendance fonctionnelle comme ils visualisent comment les valeurs de quelques attributs tels que A, B peuvent prédire la valeur d'un attribut cible. Dans un autre contexte, le mesure M est l'attribut cible. Dans l'arbre de décision de classe, nous essayons de sélectionner des attributs basés sur le gain d'information. Dans un arbre exacte de décision, chaque feuille présente la valeur de l'attribut cible M. Dans ce cas, cette valeur est de prédire avec une forte probabilité. Dans le cas d'un arbre de décision approximatif, chaque feuille prédit la distribution de la cible. Dans cette distribution, les valeurs de M sont prévues avec la distribution de Dirac. Dans notre contexte, les principaux critères sont la distance et nous voulons trouver des attributs tels que $d(A)$ est petit. Si aucun attribut satisfait à ces critères, nous recherchons des paires A_i, A_j, puis triples et ainsi de suite. Si il ya une dépendance fonctionnelle $A- > M$, le gain d'information est maximale dans L_1 et L_2, et la distance entre les deux distributions est nulle. Si nous tolérons les erreurs, la distance serait faible. Cependant, il peut y avoir une petite distance et un gain d'information faible, et encore un gain d'information de zéro. Ceci est est réalisé avec les distributions uniformes sur la mesure. Dans ce cas, la distance $d(A)$ reste faible. Les petites critères de distance est donc plus général que le gain d'information et permet de généraliser la construction d'arbres de décision.

X. IMPLEMENTATION Nous testons nos méthodes d'approximation: l'approximation par échantillonnage uniforme, l'approximation par échantillonnage mesure fondée et par le modèle statistique. Nous utilisons un entrepôt de données simule des capteurs et un véritable entrepôt de collecte de données RSS à partir des sites web sur Internet. Pour chaque méthode, nous nous intéressons à la qualité de l'approximation comme le taux d'erreur et le niveau confiant. A la fin, nous analysons les résultats et compare les méthodes. Nous utilisons MySQL pour les données relationnelles, OLAP Mondrian pour le moteur et une version améliorée de JPivot où les réponses sont représentées graphiquement par pie chart multidimensionnels.

Bibliography

[1] Acharya, S., Gibbons, P. B., and Poosala, V. (2000). Congressional Samples for Approximate Answering of Group-By Queries. In *Proc. of SIGMOD Conference*, pages 487–498. 1, 24

[2] Akinde, M. O., Böhlen, M. H., Johnson, T., Lakshmanan, L. V. S., and Srivastava, D. (2003). Efficient OLAP query processing in distributed data warehouses. *Inf. Syst.*, 28(1-2):111–135. 1

[3] Alon, N., Matias, Y., and Szegedy, M. (1999). The Space Complexity of Approximating the Frequency Moments. *Journal of Computer and System Sciences*, 58(1):137–147. 14

[4] Aouiche, K. (2007a). Jpivot Documentation. *http://eric.univ-lyon2.fr/ kaouiche/inf9002/jpivot.html*. 73

[5] Aouiche, K. (2007b). Mondrian and Jpivot Course. *http://eric.univ-lyon2.fr/ kaouiche/inf9002/index.html*. vii, 71, 72, 73

[6] Borgs, C., Chayes, J., Lovász, L., Sós, V. T., and Vesztergombi, K. (2009). Limits of randomly grown graph sequences. *ArXiv e-prints*. 101

[7] Chernoff, H. (1952). A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *Annals of Mathematical Statistics*, 23(4):493–507. 11

[8] Cormode, G. (2009). Count-Min Sketch. In *Encyclopedia of Database Systems*, pages 511–516. 1

[9] Cormode, G., Muthukrishnan, S., and Srivastava, D. (2003). Finding hierarchical heavy hitters in data streams. In *Proc. of VLDB*, pages 464–475. 1

[10] Cormode, G., Muthukrishnan, S., Yi, K., and Zhang, Q. (2012). Continuous sampling from distributed streams. *J. ACM*, 59(2):10:1–10:25. 1

[11] Cuzzocrea, A. (2009). CAMS: OLAPing Multidimensional Data Streams Efficiently. In *Proc. of DaWaK*, pages 48–62. 1

[12] Cuzzocrea, A. and Gunopulos, D. (2010). Efficiently Computing and Querying Multidimensional OLAP Data Cubes over Probabilistic Relational Data. In *Proc. of ADBIS*, pages 132–148. 1

[13] de Rougemont, M. and Vieilleribière, A. (2007). Approximate Data Exchange. In *Proc. of ICDT*, pages 44–58. 14

[14] Fagin, R., Kimelfeld, B., and Kolaitis, P. G. (2010). Probabilistic data exchange. In *Proc. of ICDT*, pages 76–88. 14

[15] Fagin, R., Kolaitis, P., Miller, R., and Popa, L. (2003). Data Exchange: Semantics and Query Answering. In *Proc. of ICDT*, pages 207–224. 14

[16] Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(2):13–30. 13, 19, 22, 48, 53, 54, 56

[17] Hyde, J. (August 2006). Mondrian Documentation of Pentaho. *http://mondrian.pentaho.com/documentation/olap.php*. 2, 71, 72, 73

[18] Jermaine, C., Arumugam, S., Pol, A., and Dobra, A. (2007). Scalable approximate query processing with the DBO engine. In *Proc. of SIGMOD Conference*, pages 725–736. 1

[19] Joshi, S. and Jermaine, C. (2008). Materialized Sample Views for Database Approximation. *IEEE Trans. on Knowl. and Data Eng.*, 20(3):337–351. 1

[20] Kapp, R. M., Luby, M., and Madras, N. (1989). Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448. 11

[21] Kushilevitz, E. and Nisan, N. (1997). *Communication complexity*. Cambridge University Press, New York. 60

[22] Lovasz, L. (2009). Very large graphs. *Current Developments in Mathematics 2008*, pages 67–128. pdf. 36, 101

[23] Manku, G. S. and Motwani, R. (2002). Approximate Frequency Counts over Data Streams. In *Proc. of VLDB*, pages 346–357. 1

[24] Muthukrishnan, S. (2005). Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2). 14

[25] Palpanas, T., Koudas, N., and Mendelzon, A. (2005). Using Datacube Aggregates for Approximate Querying and Deviation Detection. *IEEE Trans. on Knowl. and Data Eng.*, 17(11):1465–1477. 1

[26] Rougemont, M. D. and Cao, P. T. (2012). Approximate Answer to OLAP Queries on Streaming Data Warehouses. In *Proc. of DOLAP Conference*, pages 121–128. 35, 61

[27] Spyratos, N. (2006). A Functional Model for Data Analysis. In *Proc. of FQAS*, pages 51–64. 5, 6, 7

[28] Wood, S. and JasperSoft. (April 2007). Schema Workbench Documentation. *http://mondrian.pentaho.com/documentation/workbench.php*. 75

[29] Wu, S., Ooi, B. C., and Tan, K.-L. (2010). Continuous sampling for online aggregation over multiple queries. In *Proc. of SIGMOD Conference*, pages 651–662. 1