Development and Implementation of the Quasi-Online Archive System for the Mixed Astrometrical and Photometrical Data

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Doktor rer.nat. Andrew Belikov
aus Moskau

Mannheim, 2005

Dekan: Professor Dr. Matthias Krause, Universität Mannheim
Referent: Professor Dr. Guido Moerkotte, Universität Mannheim
Koreferent: Professor Dr. Rainer Spurzem, Universität Heidelberg

Tag der mündlichen Prüfung: 30. Juni 2005

Abstract

The progress of the astronomy means an increase in the data volume which must be handled. It is impossible to use the astrometrical data obtained with a space satellite without intensive data processing made in a specially organized Data Processing Center.

In the presented study the Data Processing Center for a astrometric space mission was developed through conceptual, logical and physical design of the Center. We will review possibilities for the construction of the Center, hardware and software components and the design of the database for the mission.

The goal of the Thesis is to construct a Data Processing Center which will minimize the response time of time-critical requests. The distributed database in the shared-nothing environment is proposed for the data storage. A new approach to the data partitioning through nodes of the cluster is studied.

Acknowledgments

I would like to express my gratitude to Dr. Siegfried Röser and Dr. Elena Schilbach for the invitation to perform the presented study and for helping me with the study. I am very grateful to Dr. Sonja Hirte for the help with the work, for reading and spelling it and for instant advices to finish it finally.

This study would be impossible without financial support of the Klaus Tschira Stiftung. I would like to thank Dr.h.c. Klaus Tschira for his interest to this work and to the DIVA project.

I would like to acknowledge Prof. Dr. Guido Moerkotte for the supervision of this work and Dr. Sven Helmer for the help and advices during the work.

Zusammenfassung

Wie auf vielen anderen Gebieten ist die gegenwärtige Entwicklung der Astronomie durch einen enormen Zuwachs von Daten gekennzeichnet. Insbesondere trifft das auf astrometrische und photometrische Daten zu, die mit einem Weltraumsatelliten gewonnen werden. Eine effektive Bearbeitung solcher Daten ist ohne ein speziell angepasstes Datenverarbeitungszentrum nicht möglich.

Diese Promotionsarbeit ist der Entwicklung eines optimalen Datenverarbeitungszentrums für eine astrometrische Weltraummission gewidmet, und zwar von der Erstellung eines Konzepts bis zum logischen und physischen Design. Wir prüfen und vergleichen verschiedene Mögligkeiten für den Aufbau eines solchen Zentrums, diskutieren notwendige Hardware-, Softwarekomponenten und das mögliche Datenbankdesign für die Mission.

Eine der Hauptforderungen an das Konzept war die Gewährleistung minimaler Antwortzeiten bei zeitkritischen Anfragen an die Datenbank. Für die Datenspeicherung wird eine *distributed database in the shared-nothing* Umgebung vorgeschlagen. Ein neues Verfahren zur Verteilung von Daten auf Knoten eines PC-Clusters wird dabei untersucht.

In dieser Arbeit wurden zwei Hauptaufgaben untersucht, und die den Aufgaben entsprechenden Lösungen vorgeschlagen. Zum ersten soll ein Konzept des Datenverarbeitungszentrums (*DPC*) entwickelt werden, das die Forderungen einer astrometrischen Weltraummission erfüllt. Das *DPC* soll eine regelmässige Speicherung und pünktliche Verarbeitung aller Daten garantieren, die während einer Mission anfallen. Dabei sollen Kosten des *DPC* möglichst niedrig gehalten werden (ca. 5%-7% der Gesamtkosten der Mission).

Zum zweiten wurde untersucht, welche Möglichkeiten bestehen, die Daten so zu organisieren und zu verteilen, dass die Antwortzeiten bei der routinemässigen Datenverarbeitung minimal sind. Da täglich neue Beobachtungen anfallen, müssen die zeitkritischen Anwendungen, die umfangreiche Datenvolumen aus der Datenbank bearbeiten, optimal geplant und gewährleistet werden.

Ein Konzept des *DPC* für eine astrometrische Weltraummission wurde entwickelt und getestet. Der logische und physikalische Aufbau des *DPC* basiert auf Simulationen der routinemässigen Anwendungen, auf Evaluierung von Antwortzeiten bei umfangreichen Anfragen an die Datenbank, auf Untersuchung dazu geeigneter Hardware- and Softwarekomponenten.

Eine *Benchmark* für astronomische Datenbanken wurde entwickelt und getestet. Die vorgeschlagene *Benchmark* berücksichtigt das Datenverarbeitungschema, welches bei regelmäsigen astronomischen Beobachtungen mit Instrumenten mit großen Feld und mit CCD-Detektoren allgemein gültig ist.

Eine signifikante Verringerung der Antwortzeiten wurde durch eine optimale Verteilung (physische) von Daten erreicht. Als Lösung wurde ein horizontales *hash partitioning of the distributed database in a shared-nothing environment* entwickelt, wobei eine große Anfrage (mehr als 10^3 Zeilen pro Transaktion) vorausgesetzt wird. Es wurde festgestellt, dass in diesem Fall eine lineare Abhängigkeit der Antwortzeiten von der Zahl der Datensätze

angenommen werden kann. Das Szenario für die Datenverteilung folgt aus der Prüfung des aktuellen Status des Hardware/Software-Systems während der Datenverarbeitung. Dabei konnten die Antwortzeiten im Vergleich mit einer gleichmäßigen Datenverteilung um einen Faktor 10 verkleinert werden.

Das vorgeschlagene *DPC*-Konzept ist auch für erdgebundene Teleskope geeignet, wenn eine große Himmelsdurchmusterung mit CCD-Detektoren vorgenommen wird.

Die Promotionarbeit besteht aus 10 Kapiteln und 8 Anhängen. Das erste Kapitel gibt eine Einführung in die Problemstellung. Im Kapitel 2 wird eine Übersicht über die wissenschaftlichen Ziele der Astrometrie und über die neuesten Entwicklungen in der Weltraumastrometrie gegeben sowie die Hauptschritte bei der Datenreduktion während einer astrometrischen Mission beschrieben. Im Kapitel 3 werden die Anforderungen an ein *Data Processing Center* formuliert. Die Anforderungen ergeben sich aus einer Analyse der Art und des Umfangs der Daten, von notwendigen Speicherkapazitäten, von typischen Anforderungen seitens der Datenreduktion. Das entsprechende Konzept für *DPC* ist im Kapitel 4 beschrieben. Das logische und physische Design des *DPC* ist in den Kapiteln 5 und 6 dargestellt. Hier werden auch die Vorschläge zur Optimierung der Datenbankstruktur diskutiert, die Hardware- und Softwarekomponenten sowie der Zeitplan für Arbeiten des *DPC* beschrieben.

Bei der Entwicklung des in der Kapiteln 2–6 vorgestellten DPC wurden immer zwei Aspekte im Auge behalten: eine signifikante Verringerung von Antwortzeiten bei der Auswahl notwendiger Daten aus der Datenbank für die verschiedenen Anwendungen. Die Lösung hat als Randbedingung möglichst niedrige Kosten des *DPC*. Ein nächster Schritt zur Lösung dieses Problems wird im Kapitel 7 gemacht, wo der theoretische Hintergrund der Datenverteilung im Fall einer *shared-nothing*-Umgebung diskutiert wird. Es wurde festgestellt, dass bei groß en Datenanfragen eine lineare Abhängigkeit der Antwortzeiten von der Anzahl der angefordeten Datensätze angenommen werden kann. Dieses relativ einfache Herangehen stellt die wichtigsten Voraussetzungen für die Lösung des Problems einer optimalen Datenverteilung dar. Kapitel 8 beschreibt die Anwendung eines Systems, das die Lösung realisiert, und die erreichten Ergebnisse für die Datenverteilung. Eine für astronomische Datenbänke vorgeschlagenes *Benchmark* wird im Kapitel 9 dargestellt. Das *Benchmark*-System berücksichtigt das allgemeine Schema, welches die für eine Datenbank astrometrischen Beobachtungen durch scannende Satelliten üblich ist. Die verschiedenen Tests des *Benchmark*-Systems wurden ausgeführt und dargestellt. Im Kapitel 10 werden die Ergebnisse der Promotion zusammengefasst.

In den Anhängen findet sich eine detaillierte Beschreibung aller in der Arbeit durchgeführten Tests. Der Anhang A zeigt die Unterschiede zwischen der Anwendung der Java und C/C++ Programmiersprachen bei Datenanfragen auf. Der Anhang B stellt die Ergebnisse dar, die für die Antwortzeiten pro Transaktion in Abhängigkeit von der Zahl der Datensätze erreicht wurden. Im Anhang C werden die Tests für die Änderung von Antwortzeiten bei einer Vergrößerung der Anzahl der für die Datenverteilung genutzten Knoten dargestellt. Im Anhang D wird die Stabilität der Antwortzeiten für Datenbankanfragen untersucht. Wie gut eine lineare Form die Abhängigkeit der Antwortzeiten von Rekordsanzahl beschreibt, wird im Anhang E geprüft. Die Anhänge F und G untersuchen Abhängigkeiten der Antwortzeiten von der Größe der Tabellen. Anhang H zeigt die Tests zur Stabilität der *time cost function*. Im Anhang I werden die Unterschiede zwischen

Laden von Daten mit Hilfe von INSERT und der *DB2 utility db2split* beschrieben.

Contents

1 Introduction **1**

1.1 Goal of the Thesis 2

1.2 Summary of Contributions 3

1.3 Outline of the Thesis 3

2 Modern space astrometry **5**

2.1 Astrometry at a Glance 5

2.2 Space missions in astrometry 6

2.3 Astronomical & astrometric data processing 8

2.4 Summary 14

3 Requirements to the Data Processing Center **17**

3.1 Tasks of the DPC 17

3.2 Data Flows 19

3.2.1 Data Flow Scheme 19

3.2.2 Format of the Incoming Raw Data 20

3.3 Requirements from the Data Processing 21

3.3.1 Daily Schedule 21

3.3.2 Requirements from Applications 22

3.4 Requirements for the Data Storage 23

3.4.1 Estimation of the Scientific Raw Data Volume 23

3.4.2 Requirements to the Backup and Recovery System 24

3.5 Constraints 24

3.6 Additional Tasks of the Data Processing Center 25

3.6.1 Simulation of the Data 25

3.6.2 The Data Mining Abilities 27

3.7 Summary 27

4 Conceptual Design of the DPC **29**

4.1 The General Scheme of the Data Processing Center with Subcomponents . 29

4.2 The Model of Activity in the DPC 30

4.3 The Software Architecture 31

4.4 The Model of the Data Retrieval 34

4.5 The Model of the Data 36

4.6 Summary 38

5 The Logical Design of the DPC **41**

5.1 The Choice of the Data Storage Approach 41

5.2 The Logical Structure of the database 43

5.3 The Problems of the Logical Design. 44

5.4 Summary 44

6 The Physical Design of the DPC **45**

6.1 Attribute Domains the Data. The Required Precision 45

6.2 Optimization of the Database Structure 46

6.2.1 Theoretical Basis for the Design of the Optimum Structure 46

6.2.2 Queries from Applications to the Database 49

6.2.3 The Results of the Optimization of the Database Structure 51

6.3 The Physical Design of the Database 52

6.4 Requests to the Database from the Data Processing 57

6.4.1 Keys and Indices 61

6.5 The growth of the data with time. 62

6.6 Hardware & Software Components and Their Optimum Choice 64

6.6.1 A Shared-Nothing Linux Cluster as the Core Component of the Data Processing Center 64

6.6.2 RDBMS 66

6.6.3 Compilers 66

6.6.4 The Data Insert 67

6.6.5 Implementation of the Data Processing Center 68

6.6.6 Classification of Failures 70

6.6.7 The Backup and Recovery System 70

6.7 Summary 71

7 The Partitioning Problem: Theoretical Background 73

7.1 The Problem of Physical Data Placement 74

7.1.1 Data Partitioning Through Nodes. Tablespaces for Data and Indices 75

7.1.2 Requests to the Data. Time Cost Function 76

7.1.3 The form of the TCF 78

7.1.4 The Stability of the TCF 79

7.1.5 The Time Cost Function in the Form of Linear Matrix Equation . . 82

7.2 The Minimax Problem in the Case of the System of Linear Equations . . . 85

7.3 Estimation of the gain 86

7.4 The Use of the TCF 87

7.4.1 Optimum Number of Nodes 87

7.4.2 Optimum Control Node 88

7.4.3 The Status of the Cluster 88

7.4.4 Optimum Data Distribution 89

7.4.5 Data Redistribution 89

7.5 Minimization of the Size of the Test Database 90

7.6 Summary 93

8 The Partitioning Problem: Practical Implementation of the Solution and Results 95

8.1 Requirements to the DBSS 95

8.2 An Implementation of the DBSS for Astrometrical Databases. The Practical Solution of the Partitioning Problem. 96

8.2.1 Tests and the DBSS database 97

8.2.2 The Optimum Solution for the Data Placement 103

8.3 The Data Placement 103

8.4 The Implementation of the DBSS 103

8.4.1 The Tested Cluster 103

8.4.2 The DBSS Database 103

8.4.3 The Proof of the Results 104

8.5 Summary 106

9 A Benchmark for an Astrometrical Database 109

9.1 The Application Environment 109

9.2 The Logical Design of the Database 110

9.2.1 Attributes, Relations and the General Structure of the Database . . 110

9.2.2 The Transaction Profile . . . 110
9.3 Scaling Rules . . . 113
9.4 The Atomicity, Consistency, Isolation and Durability Properties . . . 114
9.5 Partitioning . . . 114
9.6 The Generation of the Input Data . . . 115
9.7 Response Time . . . 116
9.7.1 Specification of Control Times . . . 117
9.7.2 Computation of the Response Time . . . 117
9.7.3 Computation of rps . . . 117
9.8 System Under Test Definition . . . 117
9.8.1 Models of the Target System . . . 117
9.8.2 Hardware Definition . . . 117
9.8.3 Software Definition . . . 118
9.8.4 DBMS configuration . . . 118
9.9 Pricing . . . 118
9.10 Implementation and Results . . . 118
9.10.1 System Under Test . . . 118
9.10.2 Benchmark Database . . . 119
9.10.3 Control Times . . . 121
9.10.4 Pricing . . . 121
9.11 Summary . . . 121

10 Conclusions. 123

10.1 Summary of Results and Contributions . . . 123
10.2 Future Work . . . 124
10.3 Possible Use . . . 124

A The choice of the programming environment. C/C++ and Java 129

A.1 Test PC . . . 129
A.2 Test database . . . 129
A.3 Request . . . 130
A.4 Results . . . 130

B The single-row and multiple-row data retrieval **131**
B.1 Test PC . . . 131
B.2 Test database . . . 131
B.3 Request . . . 132
B.4 Results . . . 132

C The relation between the response time and the number of nodes **135**
C.1 Test PC . . . 135
C.2 Test database . . . 135
C.3 Request . . . 140
C.4 Results . . . 140

D The stability of the request **143**
D.1 Test PC . . . 143
D.2 Test database . . . 143
D.3 Request . . . 144
D.4 Results . . . 144

E The form of the TCF **147**
E.1 Test PC . . . 147
E.2 Test database . . . 147
E.3 Request . . . 148
E.4 Results . . . 149

F The dependence of the TCF on the size of the table in case of the local data retrieval **151**
F.1 Test PC . . . 151
F.2 Test database . . . 151
F.3 Request . . . 153
F.4 Results . . . 153

G The dependence of the TCF on the size of the table in case of the remote data retrieval **155**
G.1 Test PC . . . 155
G.2 Test database . . . 155
G.3 Request . . . 158
G.4 Results . . . 158

H The stability of the F+T matrix **161**

H.1 Test PC 161

H.2 Test database 161

H.3 Request 161

H.4 Results 161

I The choice of the data upload strategy. The direct insert and the use of DB2 utility **163**

I.1 Test PC 163

I.2 Test database 163

I.3 Request 164

I.4 Results 164

List of Figures

2.1 The principle of the parallax measurements 6
2.2 The growth of accuracy for the stellar position 7
2.3 The growth of accuracy and data volume for astrometry during the last decades 8
2.4 The orbits of DIVA/AMEX and GAIA 9
2.5 The measurement principles of scanning satellite (in the case of GAIA) . . 10
2.6 The focal plane of DIVA 11
2.7 The focal plane of GAIA 12
2.8 Illustration of the principle scheme of the focal plane 13
2.9 The principle scheme of the processing chain 15

3.1 The scheme for the development of the DPC 18
3.2 The general scheme of data flows 19

4.1 The main systems of the DPC 31
4.2 The Use Case of the activities in the DPC 31
4.3 The Use Case of the activities with the stored data 32
4.4 The subdivision of entities in the DPC 32
4.5 The first approach to the conceptual model of the raw data 33
4.6 The improved conceptual model of the raw data 33
4.7 The general conceptual data model for entities 34
4.8 Identifying entities 35
4.9 Java and C++ 35
4.10 The single-row and multiple-row data retrieval 36
4.11 The UML model for the data processing in the DPC. 39

5.1 The logical structure of the database 43

6.1 The UML scheme of the database 58
6.2 The growth of the data during the mission (DIVA/AMEX) 63

6.3 The relation between the number of nodes used in the query and the response time . . . 65
6.4 The comparison between the direct insert and the preprocessed insert . . . 67

7.1 The response time vs the CPU usage . . . 76
7.2 The distribution of the response time for different CPU usage . . . 77
7.3 The TCF for local and remote data retrieval . . . 79
7.4 The approximation of the TCF for a local data retrieval . . . 80
7.5 The approximation of the TCF for a remote data retrieval . . . 81
7.6 The response time and the quasi-stable solutions . . . 82
7.7 The distribution function of the response time . . . 83
7.8 The form of the TCF and the dependence of the TCF on the number of retrieved rows . . . 84
7.9 The matrix approach to the description of the cluster . . . 85
7.10 The optimum number of nodes . . . 87
7.11 The dependence of the response time from the number of rows retrieved and the size of the table in the case of the local data retrieval . . . 90
7.12 The approximation of the dependence of the response time from the size of the table in the case of the local data retrieval . . . 91
7.13 The dependence of the response time from the number of rows retrieved and the size of the table in the case of the remote data retrieval . . . 92
7.14 The approximation of the dependence of the response time from the size of the table in the case of the remote data retrieval . . . 93

8.1 Use case diagram for the DBSS . . . 96
8.2 Activity diagram for the DBSS . . . 97
8.3 Sequence diagram for the DBSS . . . 98
8.4 The principle scheme of the DBSS . . . 102
8.5 The response time for the uniform data distribution, the optimum data distribution and the single-node distribution . . . 106

9.1 The UML scheme of the Benchmark Database . . . 115
9.2 The records per seconds measure for RS1 statement . . . 119

E.1 The scheme of the test of the remote and local data retrieval . . . 148

List of Tables

2.1 Astrometric Space Missions 8

3.1 The data transmission rate 23
3.2 Main characteristics of the whole-sky catalogs 27

6.1 Characteristic time intervals for processes 46
6.2 Queries of the data processing 50
6.3 Attributes for the database with attributes' domains. 51
6.4 Relationships between attributes. 51
6.5 An optimum database scheme 52
6.6 The raw data tables 53
6.7 The processed data tables 54
6.8 The final catalog tables 56
6.9 The size of tables 62
6.10 GAIA and DIVA/AMEX clusters 68
6.11 The upgrade plan 69

8.1 The DBSS Database 101
8.2 The **F**+**T** matrix with dispersion 104
8.3 The response time for various data distributions 107

9.1 The benchmark database 111
9.2 The size of raw, processed and final data tables 114
9.3 Benchmark input data generation 116
9.4 The response time and rps for benchmark statements 122
9.5 The response time for RS1 122

A.1 ESQL/C and Java. 130

B.1 The difference between row by row data retrieval and cursor. 133

C.1 The request time for N nodes. 141

D.1 The quasi-stable solutions for the response time. 145
D.2 The distribution function of the response time. 145

E.1 The linear form of the TCF. 150

F.1 The dependence of the response time on the size of the table. 154
F.2 The dependence of the TCF on the tables' size. 154

G.1 The dependence of the TCF on the tables' size. 158
G.2 The response time for tables of various sizes 159

H.1 The **F**+**T** matrix with dispersion. 162

I.1 INSERT and LOAD. 164

Abbreviations and Terms used in the Thesis

There are some abbreviations and terms which must be described before the reader will start with the presented work.

Application is a completed functional unit of work which is performed as a detached program. Through the Thesis application means a unit of work which realizes the single stage of the pipeline or an external to the pipeline work (the application labeled as an "external application" in this case).

DBA, Database Administrator, a person who is responsible for the availability of the database to the users and programs that need it, the resposabilities include as well the making of backups and archiving data, on-going monitoring of the database and improvement of the database performence.

DPC, the Data Processing Center, the core element in the scientific data processing. The DPC receives the input raw data and produces the final scientific data at the end of the mission.

DBSS, the Database Statistic System, a software package, which was invented in the presented study.

DBMS, a Database Management System, a software system that facilitates the creation and maintenance and use of a database. **RDBMS** is a DBMS for the relational database, **ODBMS** is a DBMS for the object-oriented database.

DMS, in the DMS tablespace, Database Managed Space tablespace, the definition of the tablespace which stores data in files (or on devices) controlled by the he database manager.

mas, milliarcsecond, 10^{-3} of the arcsecond, $2.4(7)\,10^{-7}$ of the degree.

μ**as**, microarcsecond, 10^{-6} of the arcsecond, $2.4(7)\,10^{-10}$ of the degree.

NP-complete, a definition of a problem, which can be solved only by an algorithm whose run time is at least polynomial in the size of the input.

Pipeline is a sequence of functional units ("stages") which performs a task in several steps. Each functional unit takes inputs and produces outputs which are stored in its output buffer. One stage's output buffer is the next stage's input buffer. This arrangement allows all the stages to work in parallel thus giving greater throughput than if each input had to pass through the whole pipeline before the next input could enter.

rps, rows per second, a measure of the efficiency of the data retrieval from the database.

SMS, in the SMS tablespace, System Managed Space tablespace, the definition of the tablespace which stores data in operating system files.

SOC, the Space Operation Center, the operational center which provide an interface between scientists and the satellite. The Space Operation Center receives the data from the satellite, monitors the systems that keep the spacecraft functioning and issues commands to the satellite.

TCF, a time cost function, a time required to retrieve a single record from the database. This definition is used through the presented study.

TPC, the Transaction Processing Performance Council, a group which defines defines industry standard benchmarks that compare the ability of hardware and software platforms to perform database transactions. Also, the TPC usially designates the benchmark itself.

tps, transactions per second, a measure of the efficiency of the data retrieval from the database.

Chapter 1

Introduction

The collection of data has always been a major effort in astronomy. The increase of data and the use of collected data allow to solve a number of problems, from the calculation of orbits of planets of the solar system to the prediction of the fate of the Universe. A number of discoveries in physics, chemistry and even biology are based on astronomical data.

In the last years the enormous increase in the data volume was triggered by the installation and use of new astronomical techniques. This comprises both new large ground-based telescopes and successful space missions. Some multi terabyte databases are already in use in astronomy (for example, the Sloan Digital Sky Survey, `http://www.sdss.org`, which has already 10 TB data and is not completed yet), and in the nearest future they will become usual for astronomical archives. The data volume will increase with the development of next generation missions like SEGUE or GAIA.

But the huge data volume will remain useless without an effective data mining service provided by the owner of the data for the broad astronomical community. The storage and use of huge data volumes requires careful analysis of the strategy of data storage and software and hardware components which will be used. The problems of the data storage increase with a possible lack of financial resources available for a scientific project.

Historically the storage and processing of the large chunks of astronomical data were done in two ways: the construction of an astronomical data center or the creation of a mission data center (which includes ground-based instruments as well as astronomical satellites).

The development of Astronomical Data Centers has a history of almost 40 years. The first and leading Astronomical Data Center (*Centre de données astronomiques de Strasbourg, CDS*, `http://cdsweb.u-strasbg.fr`)) was established in 1972 in Strasbourg as a result of the cooperation between the Louis Pasteur University and Institut National d'Astronomie et de Géophysique. The goal of the newly created Data Center was not only the collection of useful astronomical information but also the distribution of this information to the astronomical community and supply of data mining abilities. Later, the CDS has initiated the foundation of astronomical data centers worldwide. One of the most successful is the Astronomical Data Center of NASA (ADS, `http://ads.harvard.edu`). The astronomical data centers have their own specific features: they possess a collection of data which is very wide in data types. The CDS data storage consists of a number of catalogs, from

tiny ones to huge all-sky surveys. Nevertheless, one of the principal results of the CDS's activity is the successful joining of a number of data sources under a single interface (SIMBAD, Vizier).

Data Centers of space missions have a more limited task: they have to work only for a restricted time during the space mission and produce a final catalog at the end of this mission. However, in practice most of the space missions data centers do not finish their work at the end of the mission but create a permanent archive of the mission. Sometimes, in case of an astronomical space mission, the data storage and handle requires to establish a new scientific institute like the Space Telescope Science Institute (`http://www.stsci.edu`) for the Hubble Space Telescope (HST).

The purposes of the space mission data center depend on the type of the mission and can be divided into two main branches: in case of the space telescope like HST, INTEGRAL (the gamma-ray space telescope, `http://isdc.unige.ch`), the Spitzer Space Telescope (the infra-red space telescope, `http://ssc.spitzer.caltech.edu`) and many others the mission data center must serve as an interface between observers and the Scientific Operation Center. This duty makes it necessary to collect requests for observations from an observer and compile an optimum plan of observations. The second purpose is to collect the observed data and reuse later to provide an access to these data for the wide astronomical community.

It is clear, that in case of the satellite which will scan the whole sky without observational program (DIVA, AMEX, GAIA) the first purpose is gone. Instead, we will have the need not only to collect data but to make all necessary scientific computations with the observed data (*the scientific data reduction* hereafter) and to provide the final catalog of the mission to astronomers. As a result the task of the data processing will be one of the primary goals for such a data center. In fact, we have to construct the data processing center as the basic element for the scientific operations with data during the mission.

We should note as well, that in case of an astrometric space mission like GAIA many astronomers will be interested in the mining of the original data. Although these applications will not be time critical and will come only at the very end of the mission the raw data volume must be stored online and must be available for tasks of data mining.

As a result we have a combination of an astronomical data center and a space mission data center, but they are separated in time: during the mission the main task is the data processing with the goal to complete the final catalog of the mission as soon as possible and after the end of the mission the data processing center can be turned into the online archive with both the raw data and the final catalog available.

1.1 Goal of the Thesis

Two problems will be solved in the present study. The first one is the problem of the concept of the Data Processing Center (*DPC*), which will satisfy the requirements from the astrometric space mission. The Data Processing Center has to supply the data storage, data processing for the mission and data mining for the external users at the end of the mission. Moreover, the budget of the DPC is limited and must not exceed some value

(usually the limit is 5%-7% of the cost of the mission). To combine the low cost for the DPC and the efficiency of the solution is the most important task of this Thesis.

The second, more technical problem is the decrease of the response time with which applications need to retrieve the data. As will be seen later, we will have a number of time-critical applications which deal with massive data chunks and these applications will have a limited time interval to retrieve data from the database.

1.2 Summary of Contributions

In the present Thesis the concept for the DPC for an astrometric space mission is proposed and tested.

The conceptual, logical and physical design of the Data Processing Center is based on the simulation of the applications at the DPC, the estimation of the response time for major requests from applications, the study of possible hardware and software components. The DPC concept will be useful not only in the case of a space astrometry mission, but also in wider fields of all-sky surveys made with ground-based telescopes. The general data processing schemes for all-sky surveys are similar due to the similarity of the observational technique (the use of the CCD, the reduction of the CCD image and the storage of the obtained data).

A benchmark for astronomical databases was developed and tested. The benchmark is based on the general scheme of the data processing in the case of periodical astronomical observations with wide field instruments and CCD detectors.

To decrease the response time for requests from applications an effective solution for the problem of the physical data placement was invented. The solution was designed for the case of a horizontal hash partitioning of the distributed database in a shared-nothing environment and the case of a massive data retrieval from tables (more than 10^3 rows in a single transaction). These circumstances allow us to use a linear approach for the dependence of the response time on the number of records retrieved. The data distribution scenario is a result of the estimation of the status of the hardware/software system at the moment of the data retrieval. The improvement of the response time is more than 10 times compared to a uniform data distribution.

1.3 Outline of the Thesis

The Thesis has the following structure: Chapter 2 is dedicated to the description of space astrometry and the typical astrometric scanning satellite. The chapter reviews the target of astrometry, the switch from ground-based astrometry to space missions and the data processing to be performed during the mission. In Chapter 3 we summarize the requirements for the Data Processing Center. The requirements are defined on the basis of the description of data flows throughout the mission, the description of the data storage capacities which are needed by the DPC and the demand from the data processing. The conceptual design of the DPC is defined by the requirements and is described in Chapter 4. The logical design is defined in Chapter 5, where problems of the logical

design are reviewed as well. The physical design of the DPC is described in Chapter 6. This includes the optimization of the structure of the database of the DPC, the description of hardware and software components of the DPC and the schedule for the work of the DPC. The development of the DPC described in Chapters 2-6 has one primary goal: to decrease the response time required for applications to retrieve data from the database of the DPC. The next step to solve this problem is made in Chapter 7 with the review of the theoretical background for the problem of data distribution in case of the shared-nothing environment (which was selected for the Data Processing Center in the previous chapter). For the massive data retrieval by the select statement, it is proved that we can use a linear form for the dependence of the response time from the number of records retrieved. This simple approach is the basis for the solution of the problem of data distribution. In Chapter 8 we describe the implementation of the system which realizes the solution as well as the results archived by the system of data distribution. Chapter 9 describes the proposed benchmark for astronomical databases. This benchmark is based on the general scheme of the database for scanning astronomical/astrometric missions and measures principle requests from applications to the database. The test of the benchmark was performed as well. Chapter 10 concludes the thesis and reviews the results of the study.

Appendices describe the tests performed during the study. Appendix A shows the difference between the use of the Java language and the C/C++ for the data retrieval. Appendix B shows the difference of the response time for data retrieval in a single transaction and the row by row data retrieval (one single row per transaction). Appendix C tests the change of the response time with the increase of the number of nodes used for the data distribution. In Appendix D we checked the stability of the response time for the data retrieval and in Appendix E we proved that the linear form suits quite well the description of the dependence of the response time from the number of rows retrieved for the massive data retrieval. Appendices F and G are dedicated to the search for the dependence of the response time from the size of the requested table. Appendix H tests the stability of the time cost function. Appendix I shows the difference between the data load with the use of the INSERT statement and the data load with the use of the DB2 utility db2split.

Chapter 2

Modern space astrometry

This chapter briefly reviews the basic principles of an astrometric space mission. The schematic layout of the satellite, the parameters of space astrometric missions and the data processing tasks during the mission will be shortly discussed. The main goal of this chapter is to show how the astrometric space satellite forms the scientific data.

2.1 Astrometry at a Glance

The topic of astrometry is the precise determination of stellar positions, proper motions and heliocentric parallaxes, which are the periodic (one year) parts of the time-derivatives of stellar positions caused by the motion of the Earth around the Sun. In general, astrometry deals with all coordinates and motions of stars on the sky. As the Earth revolves around the Sun, the direction to the star continually changes. At Figure 2.1 we can see, that the direction E_1S to a star S will change half a year later to the direction E_2S. Measuring the angles CE_1S and CE_2S and knowing the radius of the Earth's orbit a it is possible to calculate the stellar parallax π and the distance to the star d. The problem of the parallax measurements is that the distance d is very big, and as result, the angle π is very small (as well as the measured difference $CE_1S - CE_2S$). Even for the nearest star Proxima Centauri the change in the position does not exceed 1 arcsecond. The parallax for a star in the center of our Galaxy will not exceed 0.0001 arcsecond.

Another problem of the measurement of stellar parallaxes is a reference frame. The massive measurements of positions on the sky are possible for relative positions only. This means, that for each measured field on the sky some number of stars with known parallaxes and very precise positions located in this measured field must be available.

The first stellar catalog in the history of mankind was compiled by Hipparchus in the II century B.C. 200 years later it was recompiled by Ptolemaeus with a precision about 2 arcminutes. For 14 centuries the precision of observations remained the same until Tycho Brahe (16th century A.C.) improved the measurements of positions to 30 arcsec, which is the best value achievable without any technical equipment like a telescope. The first parallax was measured in 1837 by Bessel (61 Cyg, 0.3 arcsec). The improvement in astrometric accuracy over more than 2000 years of development is shown in Fig. 2.2.

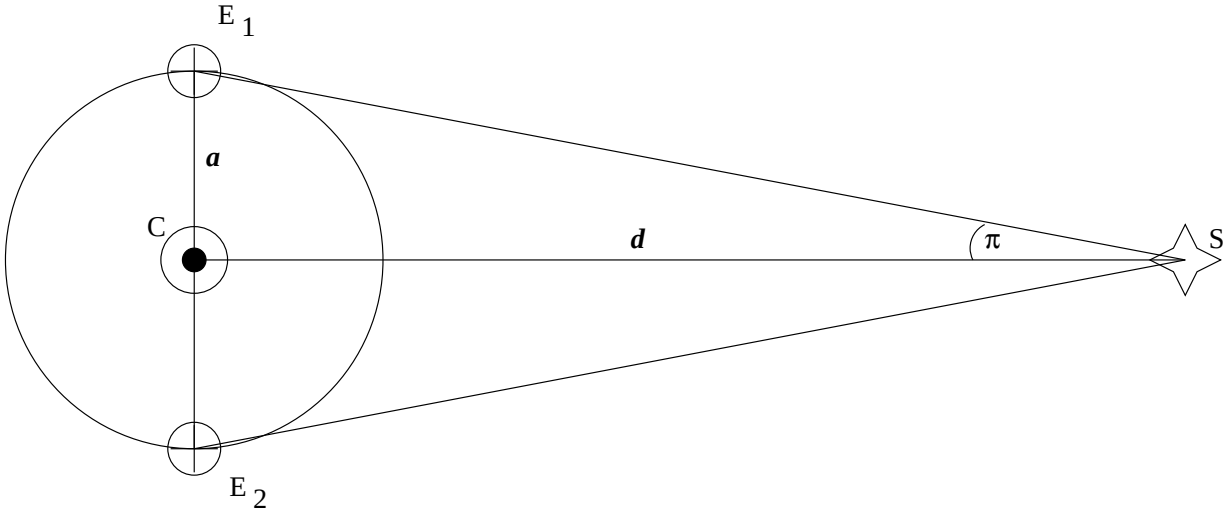

Figure 2.1: The principle of the parallax measurements. π is the stellar parallax, d is the distance to the star, a is the distance to the Sun. E_1 and E_2 are positions of the Earth with a half year time difference.

The accuracy of the ground-based measurements improved with time. The best measurements with photographic plates achieve 0.008 arcsec whereas it can be improved with use of CCD detectors to 0.002 arcsec ([van Leeuwen, 1997], for the mass measurements with wide field instruments). The best accuracy for stellar position reached with very small field astrometry is 1-2 mas ([Kovalevsky, 1995]). The new generation of ground-based telescopes was developed with the goal to provide an astrophysical research and to observe a single object. Although it is possible to use these instruments for small field astrometry and to improve the astrometric accuracy by a factor of 5-10 compared with the mass measurements with wide field instruments, the number of observations will be limited to less than 1000 objects per year. We can see, that ground-based astrometry practically reached its limits in precision. As well it is hard to create an uniform reference system for measurements of positions, parallaxes and proper motions over the whole sky with the pointed ground-based astrometry.

The importance of astrometric observations from space arises from the limitations of ground-based astrometry.

2.2 Space missions in astrometry

The first space mission for astrometry Hipparcos improved the accuracy of measurement of positions by a factor of 100 for 120,000 objects after 3 years of observation. This mission successfully ended more than 10 years ago, whereas planned astrometric missions will be able to reach 10-100 times better accuracy for millions of stars. According to the observation mode, future space missions for astrometry can be divided into two main groups: scanning instruments (FAME, DIVA, GAIA) and pointing instruments (SIM). Missions in the first group will be able to observe the complete sky and to measure positions, parallaxes and proper motions of stars down to a moderate limiting magnitude

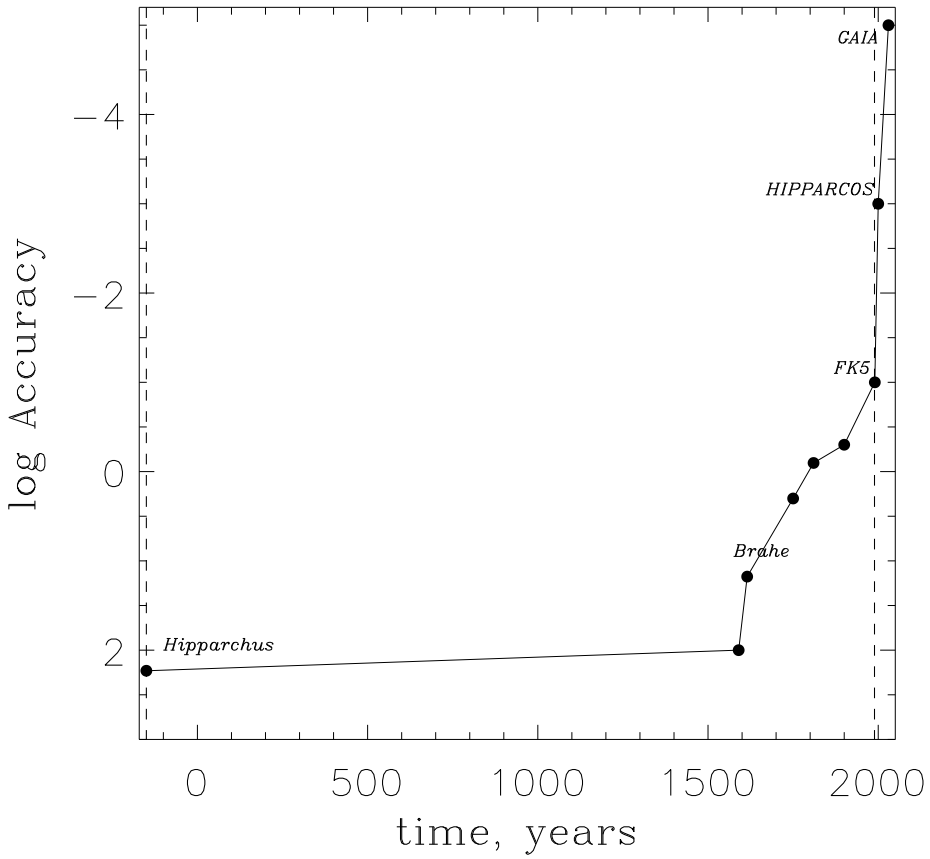

Figure 2.2: The growth of accuracy for the stellar position.

($V = 20^{\mathrm{m}}$ for GAIA), while SIM will provide an extremely high accuracy but for a limited list of objects only.

The precision of a single measurement is limited by a number of factors, the first one being the diffraction of the instrument. In the case of an observation with an ideal instrument with aperture D (the diameter of the objective of the telescope) the size of the diffraction image is $206264.8\,\lambda/D$ arcsec for monochromatic light with wavelength λ. Taking into account the photon noise of the flux observed by the instrument it is possible to evaluate, that the accuracy of a single measurement is $206264.8\,\lambda/(D\sqrt{N})$, where N is the number of photons of wavelength λ.

Let us review the main parameters of HIPPARCOS and ongoing astrometric missions to indicate the volume of the data during the mission. The Table 2.1

Simultaneously with the rise of the precision of astrometric measurements the data volume of measured positions arises as well (see Fig. 2.3).

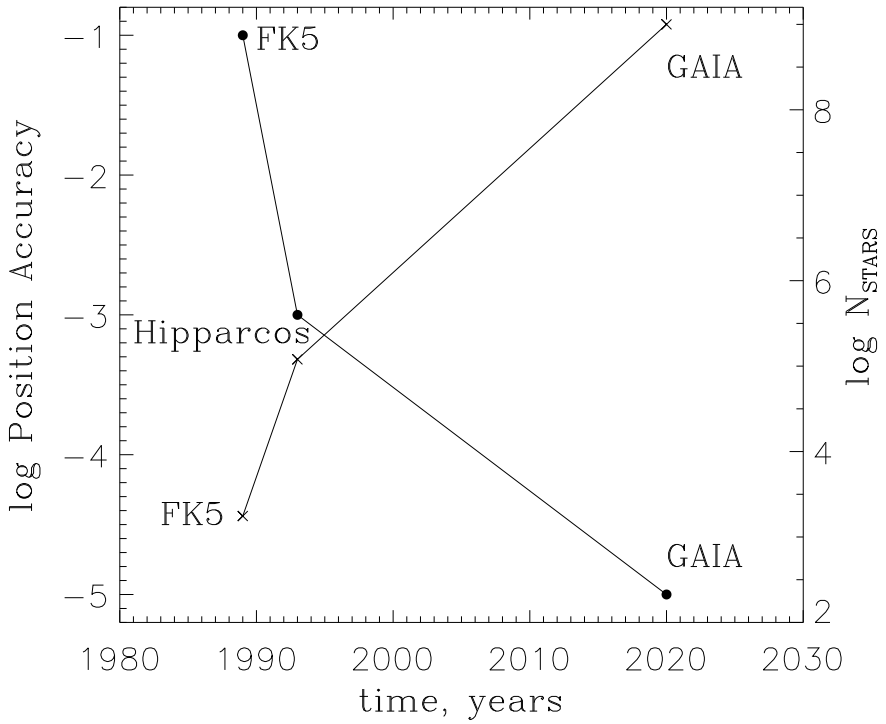

Figure 2.3: The growth of accuracy and data volume for astrometry during the last decades. The accuracy in the position measurements (circles, left axis, position's accuracy is in arcseconds) and the number of stars observed with this accuracy (crosses, right axis).

Table 2.1: Astrometric Space Missions

	HIPPARCOS	AMEX/DIVA	GAIA
Start of the mission, year	1989	canceled	2012(planned)
End of the mission, year	1993		2017(planned)
Final catalog produced, year	1997		2020
Type of observation strategy	input catalog	whole sky scanning	whole sky scanning
Limiting magnitude, mag	–	12	20
Position accuracy reached, mas	1 mas	0.19 mas (planned, $V = 12^{\rm m}$)	10 μas (planned, $V = 15^{\rm m}$)
Number of objects observed	120,000	$4\,10^7$ (planned)	10^9 (planned)

2.3 Astronomical & astrometric data processing

The main data flow during an astrometric space mission will be the data transfer from the satellite to the data processing center (**DPC**) via the science operations center (**SOC**).

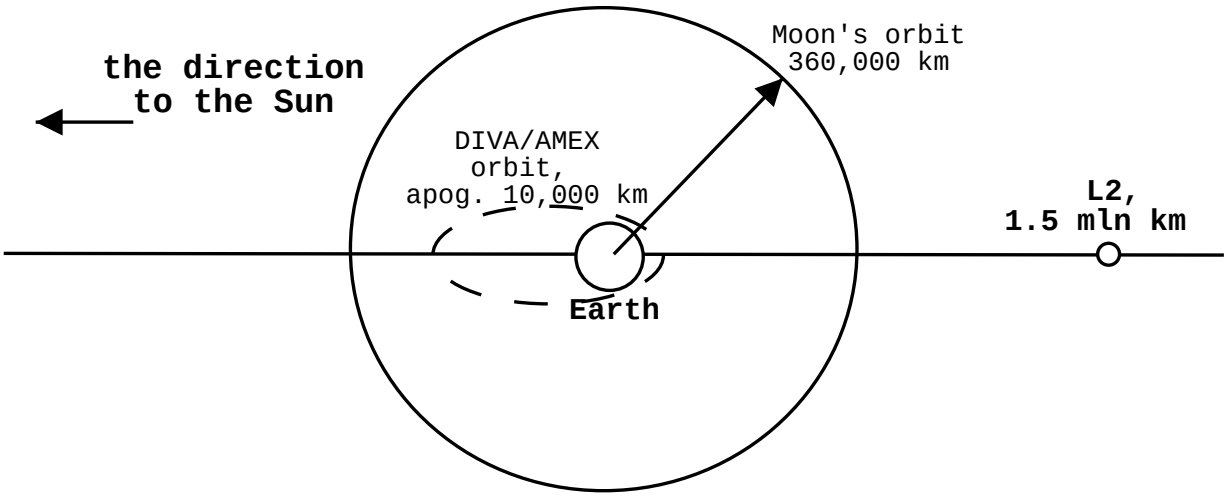

Figure 2.4: The orbits of DIVA/AMEX and GAIA. The picture is not scaled. Lagrange point L2 is shown.

The data flow from the satellite (telemetry) will include house-keeping data describing the status of the satellite (e.g., attitude, voltages, currents, temperatures, other parameters) on the one side and scientific measurements on the other side. The SOC receives telemetry data and uses it to monitor the satellite's behavior. The telemetry data will be send to the DPC for the scientific analysis. The telemetry capacity depends on the communication link between satellite and ground station (a large communication rate means a rise of the satellite's mass and therefore the price for the satellite and launch) and on the time interval during which the satellite will be observable from the ground station. In the case of DIVA the orbit was planned to be highly eccentrical so that the satellite would be observable for approx. 18 hours per day by a single ground station. The orbit of GAIA is quite different: GAIA will be placed in the L2 point of the Sun-Earth system (1.5 million km from Earth) and will be observable by a single ground station for 8 hours. The transmission rates are different as well (see Table 3.1).

The principle layout of the DIVA/AMEX satellite (as well as any scanning space satellite) includes two fields of view (*FOV*) combined at the focal plane of the telescope, where a CCD array is placed. In the case of DIVA/AMEX the beams would be combined in the single focal plane.

In the case of GAIA the two FOVs for astrometric measurements will be accompanied by a third instrument for spectroscopic and radial velocity measurements.

In case of the DIVA/AMEX on the focal plane two types of CCD mosaics are mounted. The first part consists of Sky Mappers (SM1 and SM2). The second part houses the so-called spectroscopic CCDs (SC1 and SC2). Outside this area, in undispersed light, three CCD mosaics, called Sky Mappers (SM3, SM1 and SM2), are mounted (see Fig. 2.6, SM3 is reserved and will be used in the case of the failure with SM1 or SM2). All mosaics are identical and each consists of 4 individual chips with 1024 $\times$ 2048 pixels of 13.5 micron $\times$ 13.5 micron. The CCDs are thinned, back-side illuminated for high quantum efficiency.

GAIA will have an astrometric star-mapper (ASM, with the same function as SM in case of DIVA), an astrometric field proper (AF1-11 in Fig. 2.7) and a broad-band photometer

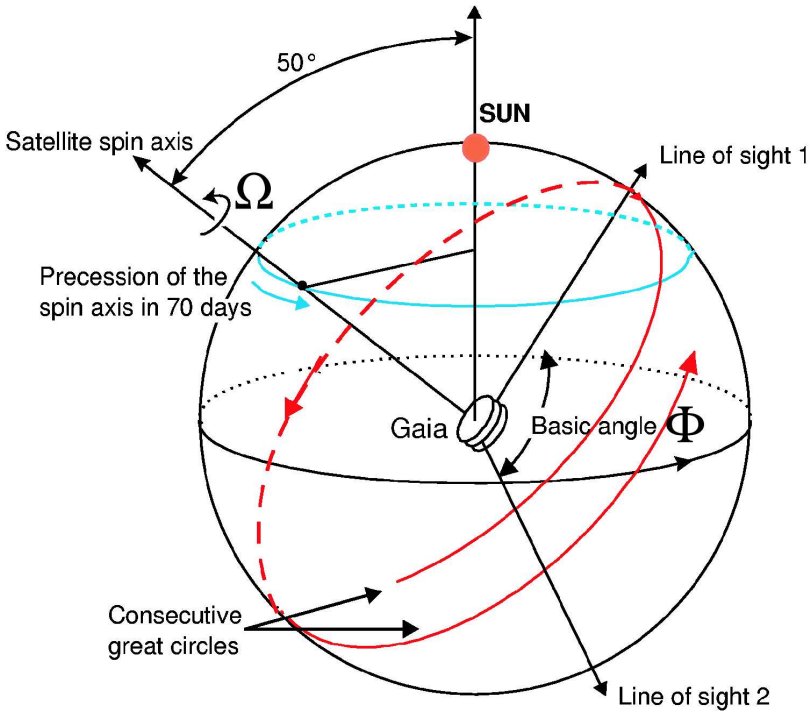

Figure 2.5: The measurement principles of scanning satellite (in the case of GAIA). The satellite rotates around the satellite's spin axis with the rotating speed Ω. The basic angle Φ is the angle between two lines of sight. Source: [de Bruijne J., 2003]

(BBP) which will measure stellar magnitudes in a number of bands.

The arrangement of the focal plane are similar for DIVA (see Fig. 2.6) and GAIA (see Fig. 2.7), and despite the differences in the construction of both satellites the general principles of observations will be the same:

- a stable angle between the two FOVs directions will be used to measure the angle between two stars,
- a CCD array will be placed in the combined focal plane of the two FOVs.

Due to the rotation of the satellite the stellar images are moving from left to right in the focal plane. The integrated exposure time per mosaic transit is 1.4 sec. All CCDs are clocked synchronously with the actual rotation of the satellite, i.e. they are operated in the so-called time-delayed integration (TDI) mode. The actual rotation rate, nominally 180 arcsec/s, is determined in real-time to an accuracy better than 0.1 arcsec/s from the crossings of individual stars through SM1 to SM2.

After detection in SM1 the onboard computer predicts windows surrounding the stars in the continuous pixel stream of the CCDs in the dispersed field. Only these windows will

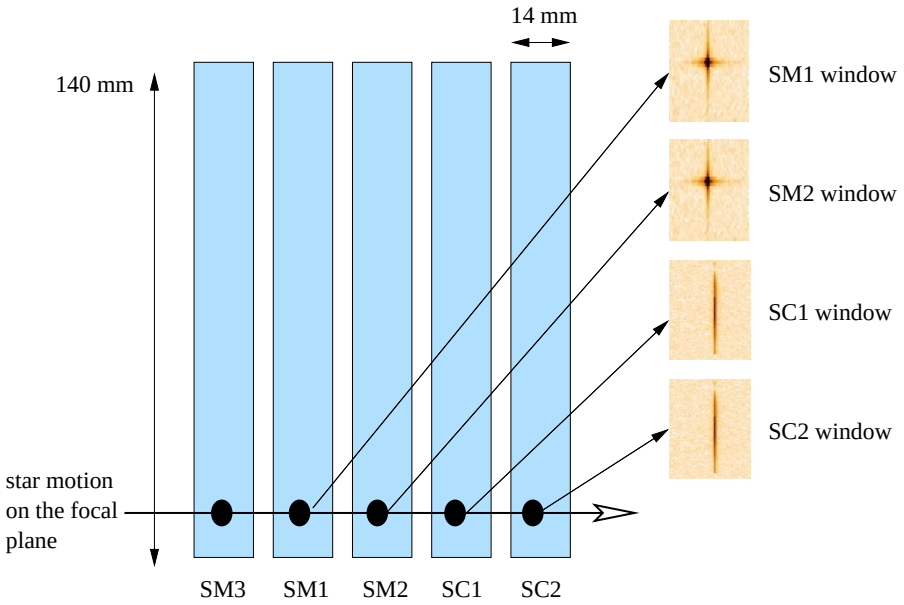

Figure 2.6: The focal plane of DIVA. The CCD mosaic SM3 is reserved one, windows were cut on mosaics SM2, SM1, SC1 and SC2.

be transferred to the ground. Additionally, smaller windows around the stars detected in the mosaics SM1 to SM2 will be transmitted to the ground.

In scan direction the full width of the central "Airy" fringe in both the SMs and the SCs is about 1.4 arcsec at a central wavelength of 750 nm. In cross-scan direction it is 2 times larger. Therefore and because the main astrometric measurements are done along scan a four times larger pixel size in this direction is used. In the present concept, this is achieved via an on-chip binning of four pixels in cross-scan direction. So, read-out noise as well as on-board data rates are reduced. On the focal plane the set of CCD is placed. During the observation, the star arises in the first Field of View (FOV, the first FOV is the first FOV on the direction of the rotation of satellite, see Fig. 2.5).

Being detected on the first CCD (sky mapper) the transition of the star will be passed on the other CCDs and only a limited window will be cut on other CCDs. After the time interval $\Delta t = \Phi/\Omega$, where Φ is an angle between two scanning directions and Ω is a rotation speed of the satellite the same star will appear in the second FOV and will be detected. The result of the single stellar detection (in one FOV) will be a set of windows from each CCD. This set will be accompanied with the clock counts, packed onboard and sent to the ground-based station with the corresponding telemetry.

A part of the CCD pixel array (a so called window) will be cut onboard and send to ground with the position of the center of the window on the CCD chip. This will be the primary information about the object which will be used to compute position of the object on the focal plane and to calculate the position of the object on the sky with the use of satellite coordinates. Like for Hipparcos, the data processing will be divided into following tasks:

- **Pixel Data Processing**. From the window (7×12 pixels in the case of DIVA/AMEX,

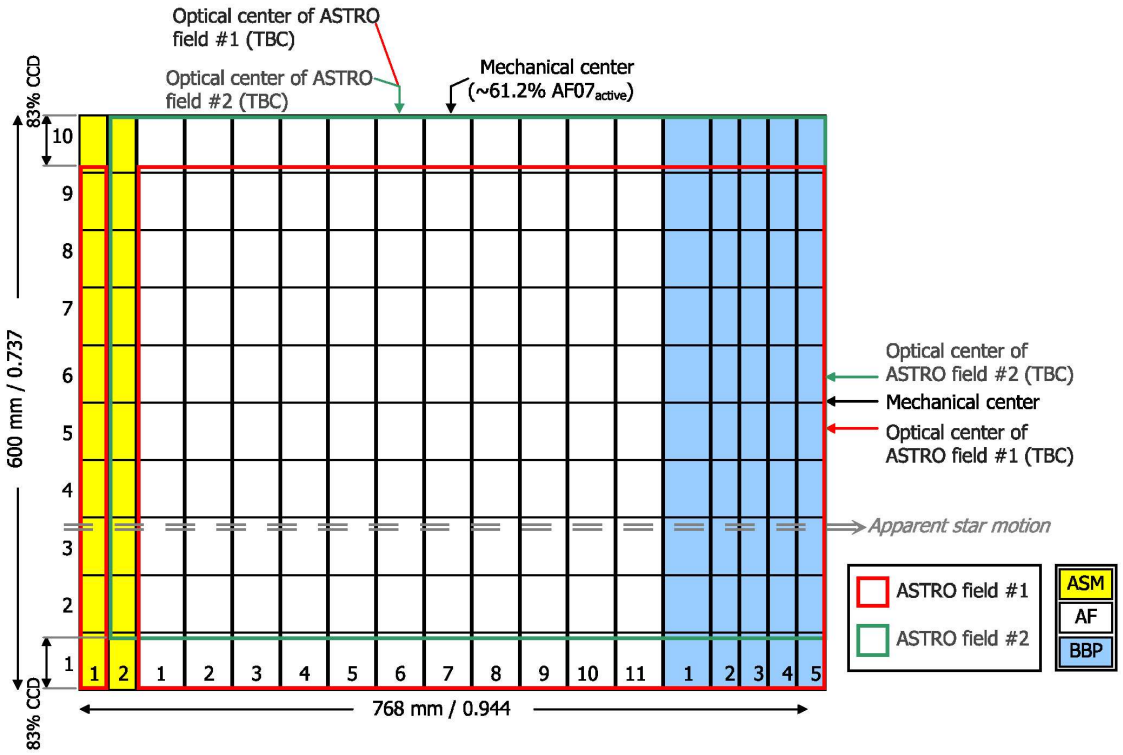

Figure 2.7: The focal plane of GAIA. Source: [Jordi C., 2003]

see [TD0284-01, 2002]) the centroid of the image will be determined as well as the brightness of the object.

- **Instrument Calibration**. Instrument calibration consists of photometric and astrometric calibrations. In the case of photometric calibration, we have to recalculate the brightness of the object to the adopted uniform photometrical system. Due to numerous effects this task will require to trace the performance of the CCD chips and to correct images. For example, CCD chip will degrade during the mission and as result the image of the same non-variable object will differ at the beginning and at the end of the mission. Astrometric calibration means the determination of all distortions which affect the position of the object on the focal plane.

- **Attitude Reconstruction**. The task of this step is to produce a relation between the satellite's internal coordinate system and the standard celestial coordinate system. To enable this information about the satellite's coordinate axis, rotation and onboard clock counts are sent to the ground periodically.

- **Great Circle Reduction (DIVA/AMEX only)**. This is the first step in the process of coordinate determination. Window centroids (from pixel data processing), instrument parameters (instrument calibration) and satellite's coordinate system (attitude reconstruction) will be combined to produce coordinates of objects on the basis of approximately 6 hours of satellite's observations. Due to the slow precession

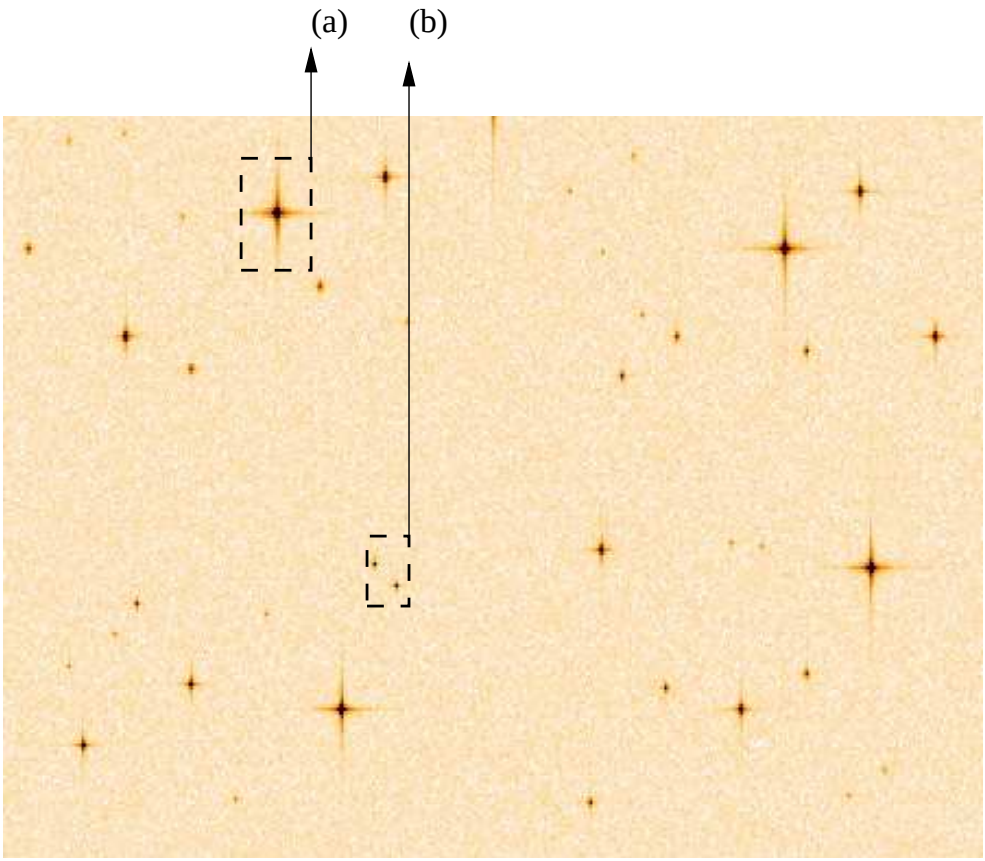

Figure 2.8: Illustration of the principle scheme of the focal plane. Two windows are shown, one with a single image (a) and one with two images (b). The window size in the case of DIVA/AMEX (7×13 pixels) is much smaller than shown here.

of the satellite's axis and the slow rotation (rotation period is approximately 2 hours) the 6 hours of observations could be interpreted as 3 scans of the same 360 degrees strips on sky and will be used to produce one-dimensional coordinates of all objects in this "strip". In the case of GAIA the approach of a Great Circle Reduction is not applicable. Any projection of the highly accurate one-dimensional single measurement onto a reference great circle other than the instantaneous one would immediately destroy the high accuracy due to the unknown cross-scan coordinate.

- **Sphere Reconstruction**. After half a year the satellite will cover the whole sky at least once with the the "strips" used in the **Great Circle Reduction**. As a result, the computation of coordinates of reference stars on the celestial sphere will be possible.

- **Astrometric Parameters Determination**. Positions of objects on the celestial sphere will be collected during the mission to produce stellar positions, parallaxes and proper motions.

- **Preliminary Identification**. To provide a link between the mission coordinate system and the standard coordinate systems we must cross-identify stars with the

most accurate positions with pre-existing sky surveys and catalogs.

- **Object Recognition**. The purpose of this task is to identify all stars and stellar objects observed by the satellite on the basis of their image coordinates. As a result the mission's system of stellar identification will be created and each image will be assigned to a star or stellar object.

From the simplified description above (see Fig. 2.9 for the principle scheme of the data processing, the **Astrophysic Parameter Determination** was omitted), one can realize, that the complete data reduction from the raw data to the final catalog is very complex task for the data reduction team. The Data Processing Center plays a central role within this process. The algorithm design and coding is an external task to the DPC and not a topic of this work.

2.4 Summary

The general description of the data processing scheme, instruments and the principle of astrometric and photometric data reduction can be summarized in the following:

- all missions targeted to the whole-sky scanning are based on the same principles. Despite some differences in instruments of DIVA and GAIA the same general description of observations can be made for both missions,
- the only important difference between DIVA and GAIA is the data volume, which will be described in the next chapter,
- the same data processing scheme can be used for both missions (except the absence of the Great Circle Reduction in the case of GAIA).

The next chapter will review the requirements to the DPC and will set problems which must be solved by the construction of the DPC.

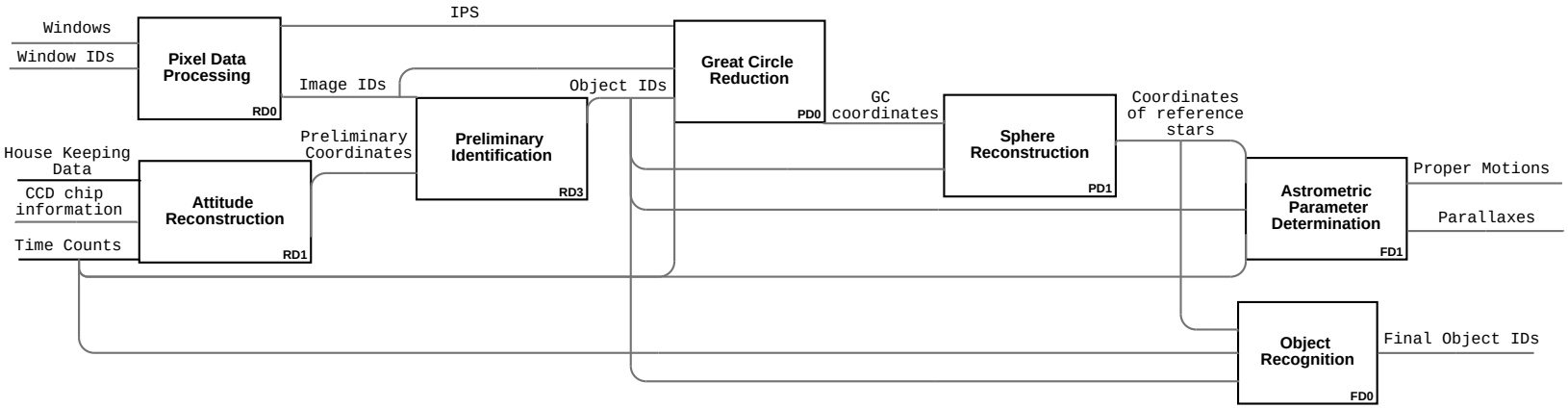

Figure 2.9: The principle scheme of the processing chain. The main data flows are shown.

Chapter 3

Requirements to the Data Processing Center

This chapter is dedicated to the review of requirements which the Data Processing Center of an astrometric space mission has to fulfill. The Data Processing Center has a number of tasks to be done during the mission. To describe the requirements to the DPC we start from the description of the data flows for the mission and from the estimation of the data volume to be stored. The description of the data processing made in the previous chapter serves as a basis for the more accurate and formal description of operations on the data in the DPC and will be combined with the requirements from time-critical applications. At the same time we have to take into account constraints which will limit possible ways of the solution. The most important ones come from the time critical applications but the limited resources which are available for the construction and support of the DPC plays a significant role as well. Fig. 3.1 shows the way which will be used to describe the development of the DPC.

The requirements to the DPC define the concept and the development of the DPC (which are subject of the next chapters).

3.1 Tasks of the DPC

The Data Processing Center is a core element for the data processing and serves as an "interface" between the data received from the satellite and the scientists.

The DPC has the following main tasks:

- to receive raw data from the Space Operation Center (SOC),
- to store raw data during the mission. This includes a regular backup and retrieval of data in case of possible failures,
- to provide an online access for users (pipeline, other applications of the data reduction process and external users at the end of the mission) to raw and processed data and the final catalog.

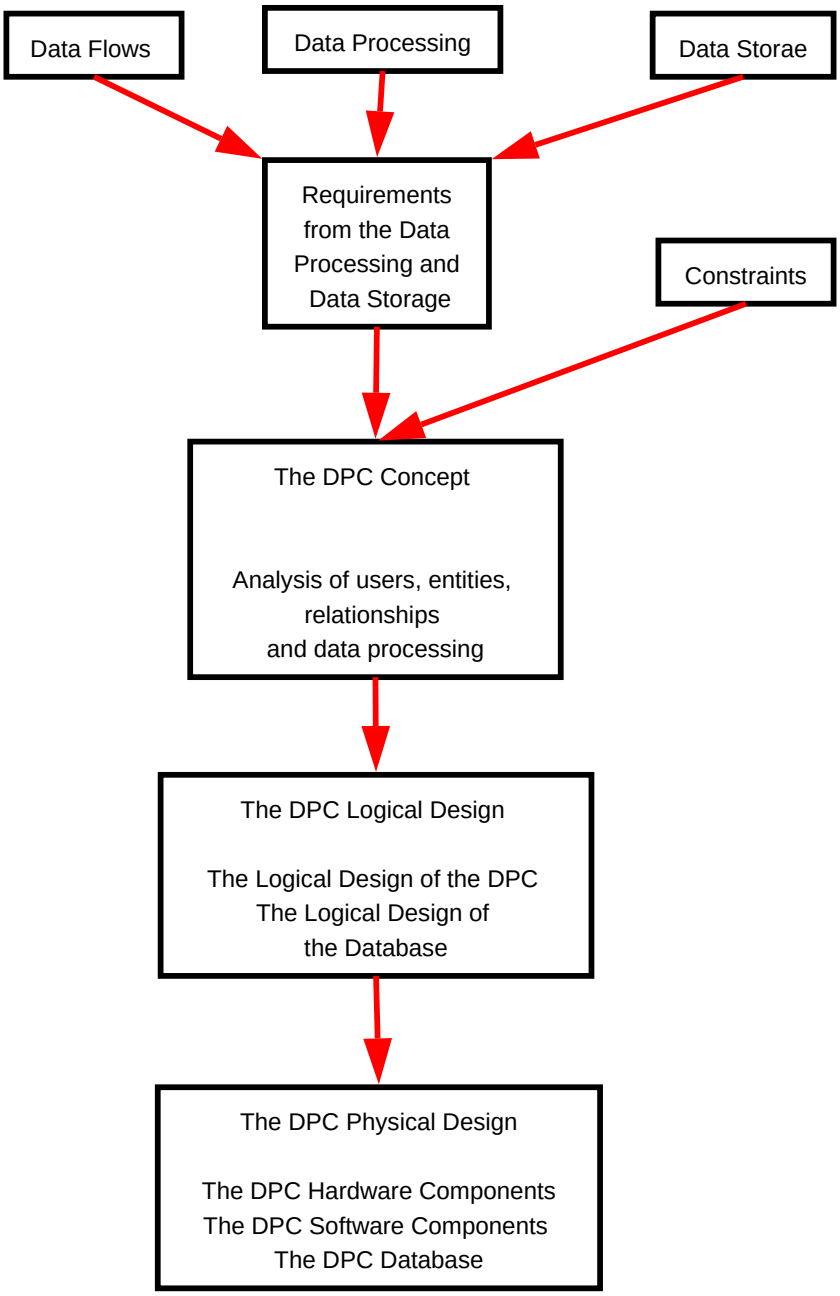

Figure 3.1: The scheme for the development of the DPC.

To define the requirements to the DPC, we have to analyse what kind of data will come to the DPC, which data processing operations will be made with the data and what is the desired final result of the mission. It is important to keep in mind that the time which can be spent for the data processing is limited and the final catalog of the mission must be supplied to astronomers as soon as possible after the end of the mission.

Some users of the DPC like First Look (FL, first and continuous inspection by GAIA scientists of science data produced with pipeline) and Scientific Quick Look (ScQL, technical

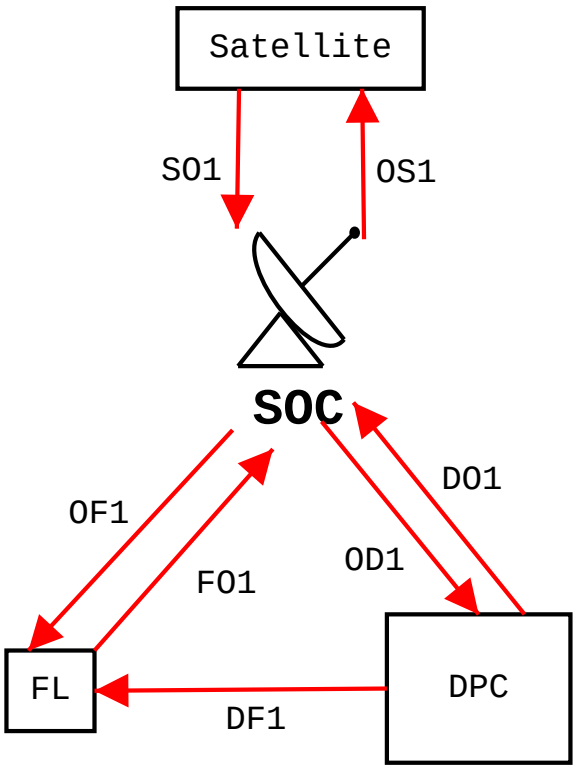

Figure 3.2: The general scheme of data flows.

inspection of satellite data at the SOC) have to access data stored in the DPC as well. Nevertheless, their design and work could be organized so that they will access incoming data in the SOC. In this work, we suppose that ScQL is installed in the SOC, whereas FL must share resources with the DPC (we will consider further FL and the DPC as two completely independent systems). In fact both ScQL and FL need only a few calibration information from the data stored in the DPC and the data flows from the DPC and FL and ScQL are not supposed to be significant.

We have to describe data flows through the mission as well as the data processing requirements and the data storage capacities which must be deployed in the DPC.

3.2 Data Flows

Data Flows include the description of the general scheme of data flows and the description of data formats of raw data.

3.2.1 Data Flow Scheme

Fig. 3.2 describes the data transmission between the subsystems of the mission.

S01 - the data flow from satellite to the SOC includes scientific data (time counts and windows) and house keeping data (position of the satellite and other parameters describing

the satellite state). The transmission rate is described in Table 3.1. The data for this data flow will be stored on the satellite, packed and transfered to the SOC. SOC will unpack the data flow and use for the First Look.

OS1 - data flow from the SOC to satellite. Commands from the SOC (for example, to change rotation rate).

DO1 - data flow from the SOC to the DPC. This is SO1 data which flows from the satellite to the DPC via the SOC.

OD1 - data flow from the DPC to the SOC containing information on the detailed study of the satellite performance (the study of the CCD chips, for example).

OF1 - data flow from the SOC to First Look (or/and Scientific Quick Look). This data flow will support First Look with the information about the satellite status.

FO1 - data flow from First Look to the SOC. This data flow is a result of the analysis of the satellite's orbit, orientation and instruments.

DF1 - data flow from the DPC to First Look. The data flow consists of the data required for the First Look to perform an analysis of the satellite state.

The DPC will receive and decode the data stream and will save the data in the database. Numerous applications will retrieve data from the database and will save results of the data processing into the database which we have to develop.

The task of the data processing is to produce astrometric and astrophysical parameters of stars from measured parameters. The DPC will receive CCD frames, angles which describe the satellite's position, onboard time counts and other parameters.

3.2.2 Format of the Incoming Raw Data

The incoming raw data are represented by three entities:

Time counts. These data store the on-board time at the moment of the object observation. TDI clock counts will increase with a 1 ms tact rate during the mission. The maximum TDI clock count will depend on the duration of the mission and will go up to 10^{11} for 3 years of the satellite's life. This will require a 34-bit integer. In theory it is possible to use 64-bit but the use of this data type will create mutual problems for applications, for example, with the portability of applications between hosts with 64- and 32-bit architectures. It is better to divide the TDI clock count on two variables: the running TDI clock count k_1 and the number of resets since the beginning of the mission k_2 (the data structure mirrors the onboard TDI timing information). Both values will depend on instrument features: we can reserve two 32-bit integers for them.

House-Keeping data. This data describe the satellite's position and performance (these include the onboard time, the position of satellite axis and the temperature regime of the satellite). The final set of parameters depends on details of the satellite construction. In any case, for the description of the satellite's orbit and positions of satellite's axes, we have to use the following parameters: the onboard time and the referenced time in UTC system (down to ms, E20.14 values were used in case of Hipparcos), Keplerian parameters of the orbit (5 values of E20.14 were used in case of Hipparcos), 6 values for the rotation axis of the satellite – 3 coordinates of the axis and 3 components of the velocity vector.

CCD chip information. This is the CCD chip number for the CCD chip in which the object was observed. To describe the position of the object on the CCD frame it is enough to use two values: they fix the corner of the window on the CCD chip in both coordinates (column of the CCD chip, "Y" coordinate and row of the CCD chip, "X" coordinate). The type of the window will be attached as well. The type of the window depends on the brightness of the detected object and defines the size of the pixel array which is cut onboard and sent to the SOC. 16-bit integers are sufficient to store these values.

Window. This array of integer values contains the main information about the object. The window size usually contains 7×13 pixels, only for a few bright stars the window will slightly larger. Also, for calibration purposes, full chip windows (524288 pixels) will be cut sometimes. Fortunately, the number of full-chip windows is limited and can be stored in a separate table. The value of each pixel is an integer and can vary from 0 to 65535. As a result, we need 2 characters (8-bit) to store each pixel value. Among the *object* windows (a window used to observe an object on the sky, but not a full chip window) the largest has $84 \times 13 - 1092$ pixels ([TD0284-01, 2002]). GAIA will have even smaller windows for most of objects ($3 \times 3 - 9$ pixels, [de Boer et al., 2000]).

Three entities (**Time Counts**, **CCD chip info**, and **Window**) define the raw data information about the observed object. Each CCD mosaics on the focal plane of the satellite (SM1, SM2, SC1 and SC2) sends to the ground these three entities to describe the object passing the focal plane. Fig. 2.6 shows the transit of the star through the focal plane. The object will be *detected* on SM1 and will be *predicted* on SM2, SC1, SC2. As a result the window which is cut on SM1 serves as a parent window for windows on SM2, SC1 and SC2. The window on SC1 is described by a set of entities **{Time Counts, CCD Chip info, Window}** , whereas windows on SM2, SC1, SC2 are described by a set of entities **{Time Counts, CCD Chip info, Window, Time Counts for the Parent Window on SM1, CCD chip info for the Parent Window on SM1}** .

We have described the raw data and requirements to the DPC from the raw data. During the data processing in the mission more entities will be created. These new entities and requirements to their precision will be observed later in the following chapters.

3.3 Requirements from the Data Processing

The data will be transferred to the DPC from the SOC as a set of files containing raw data from the satellite and telemetry commands to the satellite. The work with the data will be described below on the basis of the principle cycles of the DPC's activity.

3.3.1 Daily Schedule

The Incoming Data. The data of one day of observation will be transferred from the SOC to the DPC. The data transfer rate will depend on the quality of the link between the SOC and the data center. In the case of GAIA (85 GB of the daily data input, see Table 3.1) and an average supposed speed of 5 KBps this operation will take up to 5 hours (with the typical present day network).

The Reorganization of Incoming Data. The reorganization will include subdivision of the data into the House-Keeping Data and the scientific raw data. The generated Window Identifiers will be added to the chunk of data and used later as a partitioning key for the data. The algorithm for the generation of the Window Identifiers will be described in the next chapters (see Chapters 7 and 8).

The data load and archiving. The House-Keeping Data and the raw data will be loaded into the database. The daily data load for the scientific raw data will be 85 GB (in the case of GAIA, [EF5/FR/PC/038.02]). The daily load of the processed data will be different in case of the DIVA/AMEX and GAIA. As we will see later (see Section 6.5), the volume of the processed data for the DIVA/AMEX will be comparable with the volume of raw data whereas for GAIA the processed data volume will be 10 times bigger than the raw data volume.

The daily data processing. The daily data processing comprises Pixel Data Processing, Attitude Reconstruction, Preliminary Identification and Great Circle Reduction. All these tasks except the Great Circle Reduction work with the data chunk for the single day of observations.

The half-year data processing. Although this work does not belong to the daily tasks it will sometimes influence the daily schedule of the DPC. As soon as the data for a half-year period of observations will be processed, Sphere Reconstruction, Astrometric Parameter Determination and Astrophysic Parameter Determination will be started. These programs will run more than one day and will need to retrieve a huge data volume.

3.3.2 Requirements from Applications

The data processing chain was described in the previous chapter. Each process in the data processing chain requires not only the ability to retrieve data from the database but also to make necessary computations with these data as well. We must give some general requirements from the data processing to the computational capacities of the DPC.

Data Processing Capacity. In the case of Hipparcos 10^{10} FLOP were required for each observed star (see [O'Flaherty et al., 1997]). In the case of DIVA/AMEX, we would observe 4×10^7 stars for half a year. Assuming 19 hours of calculation each day (the rest of the time is reserved for backup/archiving duties), we will have 1.25×10^7 sec for the data processing. As a result we must provide at least 32 GFLOPS computational capacities for the DIVA/AMEX data processing. For GAIA with 10^{10} objects observed in a half-year period, we will need 8000 GFLOPS.

Time Critical Requests. The work with the data will be organized in daily cycles. This means, that every day an incoming portion of data (85 GB in the case of GAIA) must be processed. To prevent a delay in the data processing it is important to store and process this data portion before the next data portion will arrive.

The detailed description of the data processing will be done in Chapter 5.

Table 3.1: The data transmission rate. The data for Hipparcos are from [O'Flaherty et al., 1997], for DIVA from [DI-AED-RS-0001], for GAIA from [EF5/FR/PC/038.02].

	Hipparcos	DIVA/AMEX	GAIA
Data transmission rate, Mbps	0.023	0.7	1.2
Raw data per day, GB	0.24	5.25	85
onboard storage, GB	0.01	1	25

3.4 Requirements for the Data Storage

Requirements for the data storage are:

- safe storage of the data volume and the capability to quickly restore data in the case of hardware failure;
- online access to the complete data volume;
- quick response to time critical requests (esp. from the pipeline);
- the necessity to minimize the storage place.

3.4.1 Estimation of the Scientific Raw Data Volume

To provide an estimate of the data volume based on the number of observed stars, we take into account the following parameters:

- a limiting magnitude derived for the chosen S/N (Signal-to-Noise) ratio, which will determine the number of objects that can be observed by the satellite,
- an assumed distribution function $f(m)$ which describes the number of stars per unit of the sky surface in the selected range of visual magnitudes $[\mathrm{m}, \mathrm{m} + \mathrm{dm}]$,
- a multiplication factor r_M, which determines the number of images produced by the satellite. This parameter depends on the time interval of the whole sky scanning P_S and the mission duration T_M,
- a record length for the object l_r (7×13 pixels for the most of cases, see [TD0284-01, 2002]),
- the brightest stellar magnitude m_{min} and the faintest stellar magnitude m_{max} which will be observed.

Finally, the estimation of the data volume is

$$V(S/N) = \int_{m_{min}}^{m_{max}} r_M(m)\, f(m)\, l_r(m) dm.$$

The value V(S/N) is an estimate of raw data. To find the volume of processed data we have to multiply V(S/N) by the scaling factor of applications which produce processed data from raw data (pipeline). The estimation of the data volume stored into the database can be done only after the description of the data structure and will be made later after the description of the physical structure of the database (see Section 6.5).

The estimation done above are correct for the satellite with unlimited transfer capacity from the satellite to the ground. The real data volume is limited by the transfer rate for the data exchange between satellite and the SOC. Nevertheless, the precise number of records transmitted from the satellite to the SOC in each cycle of observation will depend on the brightness function of the observed sky region and the scanning law of the satellite. The parameter Signal-to-Noise plays the main role in the increase or decrease of the number of objects observed by the satellite.

The estimation for DIVA satellite was made on the basis of brightness function compiled by [Kharchenko et al., 1997] (see [TD0251-03, 2001] for results). The predicted number of stars observed down to $V = 16^{\mathrm{m}}$ is 4×10^7. The same estimations for GAIA give 10^9 stars. The number of observations for each star exceeds 170 (in the case of GAIA, [EF5/FR/PC/038.02]). Taking into account the limited transfer capacity the raw data volume in the case of GAIA will approximately the 100 TB ([EF5/FR/PC/038.02]).

3.4.2 Requirements to the Backup and Recovery System

We have to choose a backup and recovery strategy which will allow us to manage data in case of failures, and to restore the work of the Data Center as soon as possible. We have to take into account the stability of the data and data structure. Data processing will work with big chunks of data (typically a data input for one day of observation). In the processing chain, each process will receive the data from the previous process and can not be started until the previous process has finished its work. Processes select data from the database but do not update any data record. As a result, each update of the database means a huge data insert.

3.5 Constraints

The construction of the Data Processing Center for an astrometric space mission faces some constraints, where one of them – not the least important – is the financial budget. The cost of the construction of the satellite and for the launch will take more than 90 % of the project's budget. In case of DIVA, for which this work was initiated, the amount of money available for the data reduction was not specified at the beginning, but the least expensive solution had to be sought.

The cost of the deployment and support of the database center consists of the cost of hardware/software components, staff and training, downtime and outsourcing. The cost of hardware/software components usually takes only 30 % of the database center cost ([Perry, 2002]).

Hardware/software components. The cost of the hardware/software components must be reduced as well as the cost of the support of the system.

Limited manpower. The personal of the DPC has to be formed by scientists who are already working for the project. Only a few positions for programming/engineering personal are available.

Outsourcing. There are some examples of the successful outsourcing solutions in astronomy (SDSS, HST). The use of an external company for the development of the software (pipeline) will require very close cooperation between this company and the astronomical staff of the DPC, and may be not affordable by a low-cost mission.

Limited finances means: a careful balance must be kept between the cost and the effectiveness of the DPC.

3.6 Additional Tasks of the Data Processing Center

Despite the DPC will not be involved in the development of individual algorithms for the data processing, and must supply only interfaces for applications to the database we have to complete a task which concerns the development of the data processing chain. Major simulations of the raw and processed data must serve as an input during the testing of the pipeline.

3.6.1 Simulation of the Data

To simulate the raw data as they are delivered by the satellite, we have the following options ranging from the almost trivial case **A** to the realistic, but very complicated case **C**:

A) the simulation of "dummy" data, produced with the help of DB2 utilities or an external simulation program. These data will reproduce the structure of the real data only and the volume of data.

B) the simulation of the data based on the simulation of the satellite (the scanning law, the focal plane of the satellite and the output of the CCD chips in the focal plate will be simulated). A distribution of stars on the sky is assumed from some model of the Galaxy (see, for example, [Kharchenko et al., 1997]).

C) the same as **B** but with the real distribution of stars on the sky derived from one of the deep stellar catalogs.

These three methods to simulate raw data are needed for different purposes. The first method is sufficient to simulate data for benchmarks and database tests, whereas to analyse the work of the pipeline and the correctness of all processes, we have to use methods B or C. The second task does not the duty of the DPC itself, but must be done by the team of scientists and programmers which will create the pipeline running in the DPC. Nevertheless the simulated data must be stored at the DPC in the database and the DPC must be able to provide an access to these data to developers of the pipeline.

Simulated data A

The "dummy" data for the database tables are simulated by a program which will create artificial data. It is not necessary that the contents of these data set are observations of real stars. The only real requirement is the correspondence of these data to the database schema which will be developed in the next chapters.

Simulated data B

To simulate observations of an astrometric scanning satellite we have to describe the way the satellite scans the sky with its two fields of view (the so called scanning law). The satellite orbits around the Earth, moves with the Earth around the Sun and rotates around one axis (see Fig. 2.5 in the case of GAIA). This complicated motion can be described as a rotation around three axes. The first one is the direction to the Sun, the second is the satellite's axis of precession and the last one is the satellite's axis of rotation (which is in fact the axis of symmetry of the satellite as a solid body).

Data simulated by the method B and C have one common feature: we have to simulate two functions : the path of the scanning window(windows) of the satellite on the sky – the coordinate of the center of the axis of the field of view (FOV) on the sky with time $(l, b)[time]$ and the number of stars per magnitude interval for this coordinate $N(m)$ given by the brightness function. The brightness function describes the number of stars in a selected direction on the sky. The description for the simulation of the brightness function can be found in [Kharchenko et al., 1997]. The method described in this paper was developed for the DIVA satellite but can be used for GAIA as well.

Simulated data C

The only difference of this method from the method B is the fact that the number of stars is estimated not by the use of a brightness function but is taken directly from a suitable stellar catalog (like the Guide Star Catalog – [Lasker et al., 1990], for example). In this case, we can produce simulated data with the use of the database: from a stored catalog we can select data for any time interval in the scanning law. The main problem is the incompleteness of the catalog: there is no catalog complete enough to satisfy all of GAIA's requirements down to the completeness limit. Some catalogs are incomplete for bright stars as well. Nevertheless there are three catalogs which can be used at least for DIVA/AMEX data simulation:

GSC. The Guide Star Catalog was created as a master-catalog for the Hubble Space Telescope. Later it was updated and increased from the initial 18 millions stars to the present-day 998 millions (GSC II version). The GSC is incomplete in the bright end of the magnitude scale, a number of bright stars were removed from the catalog to provide an uniform distribution of bright stars over the sky which is important for the guidance of the telescope.

2MASS. The 2MASS catalog gives deep J, H, K photometry for 471 millions stars. The photometry of stars is inaccurate for close pairs. GAIA's ability to resolve close pairs will be much higher than in this catalog, yet it can be used for simulations.

Table 3.2: Main characteristics of the whole-sky catalogs.

	GSC-II	2MASS	USNO-B1.0
Band	V,J,R	J,H,K	B,R
Completeness, mag	19.5(J)	18(K)	20(B)

USNO-B1.0. The USNO-B1.0 catalog includes photometry and proper motions for 1.046 billion stars.

Table 3.2 shows the completeness limit for each catalog. The description of massive stellar whole-sky catalogs can be found in [Kharchenko et al., 2004], where the characteristics of each catalog are studied.

3.6.2 The Data Mining Abilities

The DPC will store all data volumes: raw data, processed data and the final catalog. After the end of the mission the DPC must supply the astronomical community not only with the final catalog but with the ability to access the raw data as well. To fulfill this requirement the DPC must support all data volumes online at the end of the mission.

3.7 Summary

The requirements to the DPC can be divided on the requirements from the data storage, the requirements from applications (to the computational facilities of the DPC) and some specific requirements (the astronomical data simulation ability).

The basic requirements are:

- the input raw data volume is 100 TB (in the case of GAIA). The data volume will increase with the work of pipeline and other applications (up to 1,000 TB as it is stated by GAIA),
- the computational facilities is at least 8,000 GFLOPS (in the case of GAIA),
- the ability to support the development of the pipeline with the data simulated in the DPC.

Chapter 4

Conceptual Design of the DPC

The conceptual design of the DPC describes the most general schemes used to find an appropriate, if not optimum solution for the requirements described in the previous chapter. We do not specify any hardware or software components here. Nevertheless, even at the beginning of the design of the DPC we have to concentrate on three main features which the properly designed DPC must fulfill: scalability, flexibility and availability. In our case scalability means that the DPC must be able to collect the increasing volume of the data through the mission (with daily data input of 85 GB in the case of GAIA). Flexibility is not so critical because applications for the DPC will be developed before the start of the mission. They will be improved during the mission but the interface between the application and the database must be created only once and has to be kept during the mission. Availability supposes that in the case of failure the work in the DPC must be restored before the next data input.

The conceptual design of the DPC reviews the general scheme for the DPC, activities in the DPC and the work with the data.

4.1 The General Scheme of the Data Processing Center with Subcomponents

We can divide the DPC into the following systems (see Fig. 4.1):

- The data storage system. The system must fulfill the requirements to the Data Storage described in the previous chapter.
- The software development system. This part of the DPC has to provide resources for the development and test of applications running on the DPC. The software development system allows to develop and test components of the pipeline without accessing the database of the mission. We need this subsystem for security reasons: an improper work of the application can damage the database as well as the operational system and other applications. It is important to note, that the development of the pipeline will not be finished at the start of the mission, but the data processing algorithms and software has to be improved once the actual data arrive, and do not exactly correspond to the simulation.

- The archiving system. The system partly shares resources with the data storage system.
- The data processing system. The system must fulfill the requirements from the applications described in the previous chapter.

4.2 The Model of Activity in the DPC

We have already described the data flows which will supply the DPC with the data, and we also described the requirements from applications which will be run in the DPC. Next, we have to decide how the activity of these applications will be structured inside the DPC. Therefore, we first describe the actors (users) in the DPC which will operate with the data and influence the DPC. Then we start with the modeling of activities inside the DPC. There are the following groups of users of the DPC:

- the system administrator, who is responsible for the support of the system,
- the database administrator (DBA), who is responsible for the support of the database, archiving of the data and restoring the database in the case of a failure,
- the pipeline, which includes a number of applications working with data and inserting new data into the database,
- Space Operation Center (SOC), which will have to access some data chunks to retrieve data for the Scientific Quick Look,
- Scientific Data Reduction (SDR), which will deal with the full data reduction and also with the retrieval of the new information from available data (data mining during the mission),
- External Users, who will appear at the very end of the mission and will access the database with final results, and possibly the whole data volume.

Possible activities of these users are (see Fig. 4.2):

- management of the DPC. This encapsulates all duties of the system administrator: support of the users, installation of new hardware/software and so on,
- management of the database, which comprises the DBA activities for the creation, configuration and the support of the database,
- operations with the database (data retrieval, insert and update).

Figure 4.2 illustrates all the activities in the DPC, whereas Fig. 4.3 shows the interactions of the users with the database.

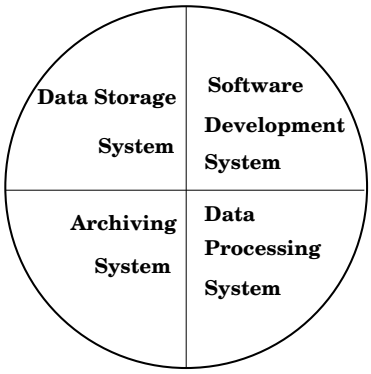

Figure 4.1: The main systems of the DPC.

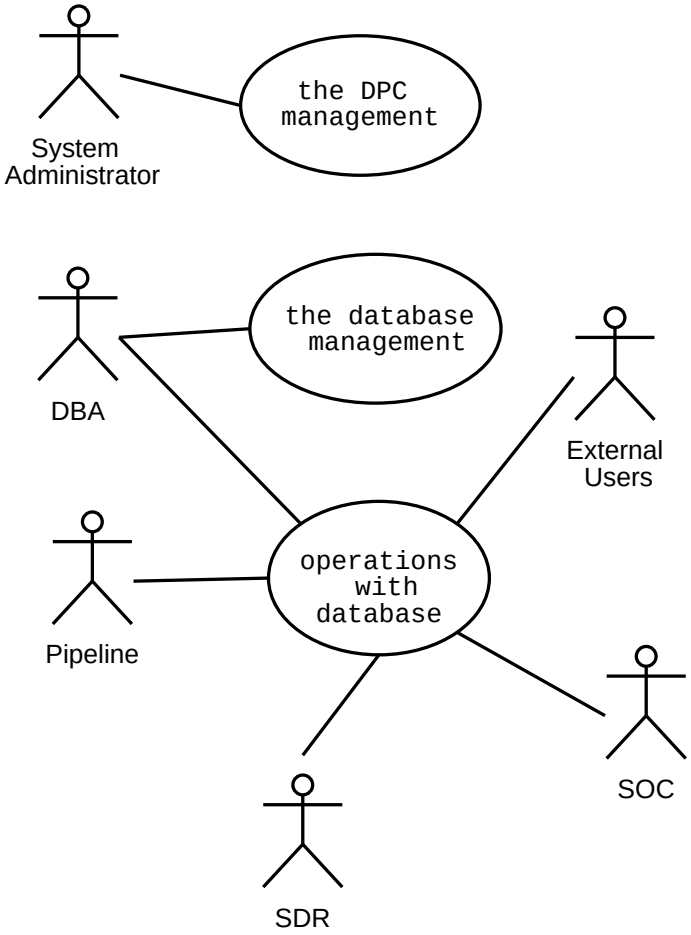

Figure 4.2: The Use Case of the activities in the DPC.

4.3 The Software Architecture

To provide an access to the data we have to supply the applications which will make scientific data reduction with an interface to the database. The access to the data can be

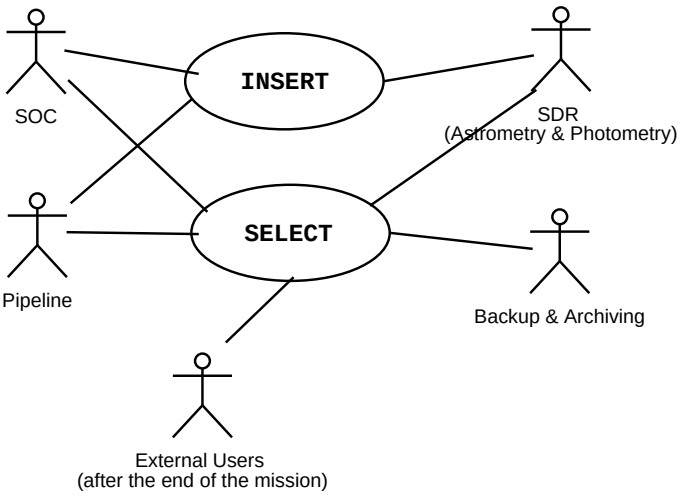

Figure 4.3: The Use Case of the activities with the stored data.

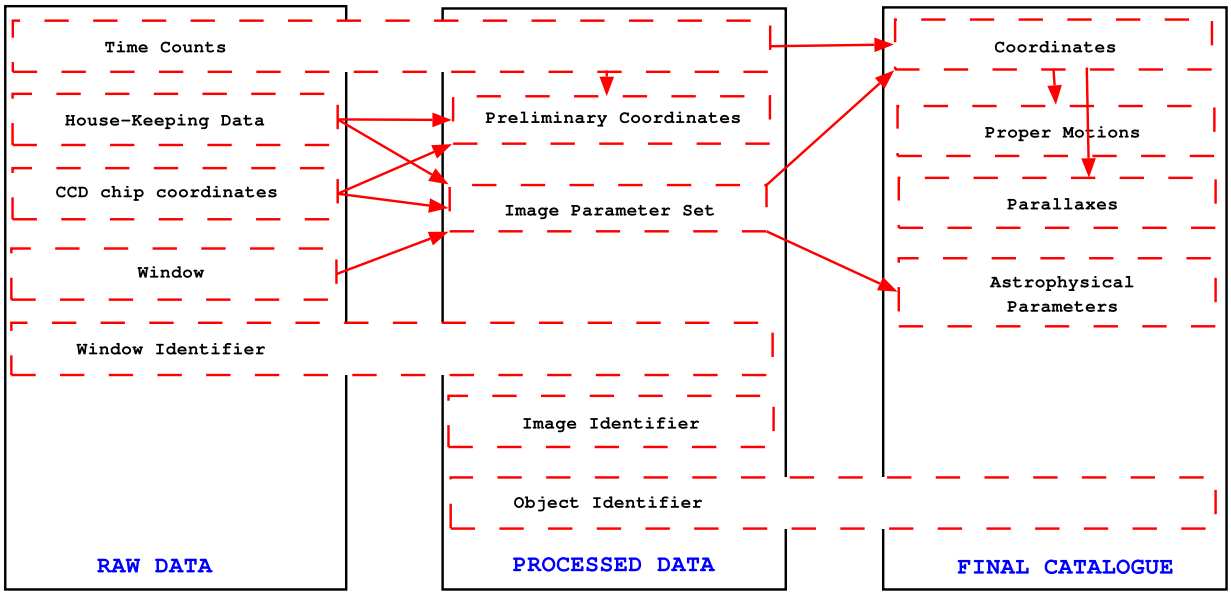

Figure 4.4: The subdivision of entities in the DPC.

organized by following methods:

- with the use of n-tier technology. The application has to use a middleware as an interface to the data (see [Chavez, 2000] for an example of the use of the n-tier in astronomy);
- with the use of the direct access to the database with the interface to the data implemented in the application.

The most important advantage of the n-tier technology compared to the direct access in the case of the DPC is a better flexibility (a number of applications can use the same interface, applications are independent from the database structure). The most important

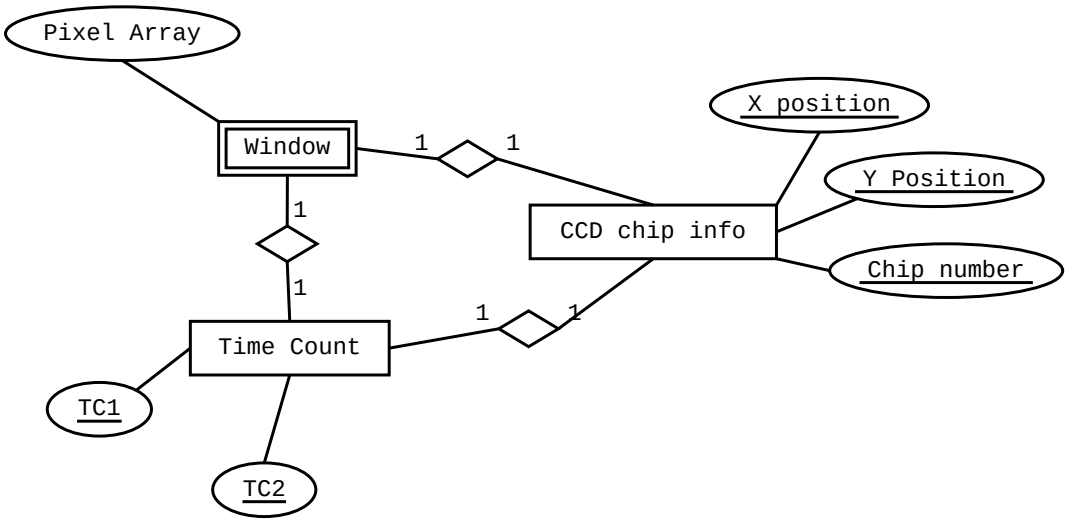

Figure 4.5: The first approach to the conceptual model of the raw data.

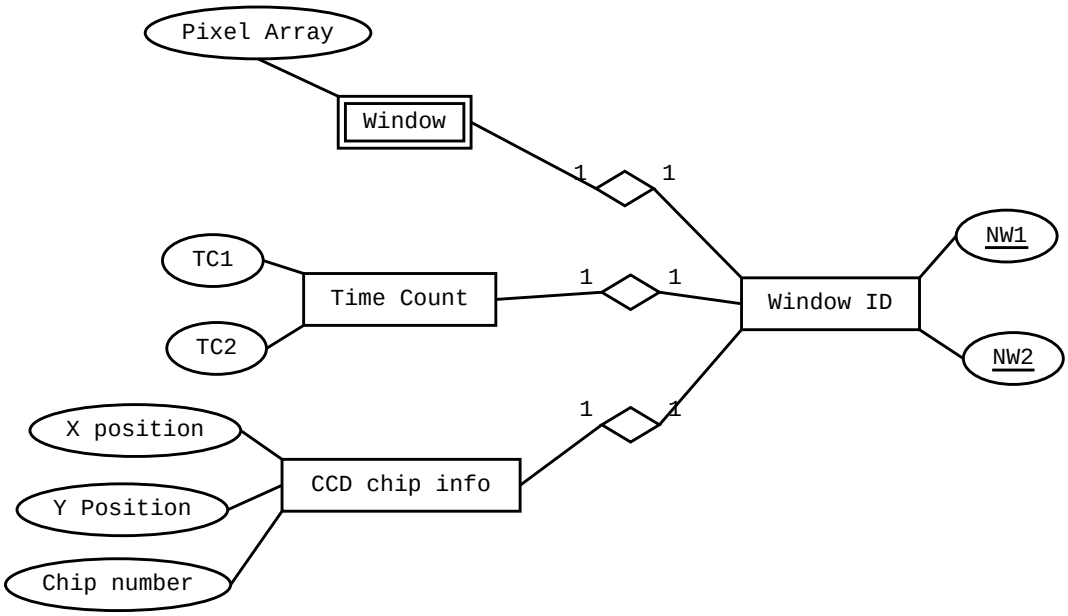

Figure 4.6: The improved conceptual model of the raw data.

disadvantage is a delay in the data transfer from the database to the application which occurs at each new layer of n-tier architecture.

We can use any language-interpreter (like Java, perl or python) to simulate the delay in the data transfer in the case of the n-tier architecture. In Appendix A we measured the response time for the data retrieval from the same database made at the same moment of time with the use of two applications: the first one was written in C++ (no middleware) and the second one was written in Java (JVM as a middleware). Fig. 4.9 shows, that the use of n-tier architecture produces a delay in the data transfer.

The result does not mean, that we reject the n-tier architecture for the software of the DPC. There are a number of non-time-critical applications where the n-tier architecture can be used successfully and can reduce efforts for the development and use of these

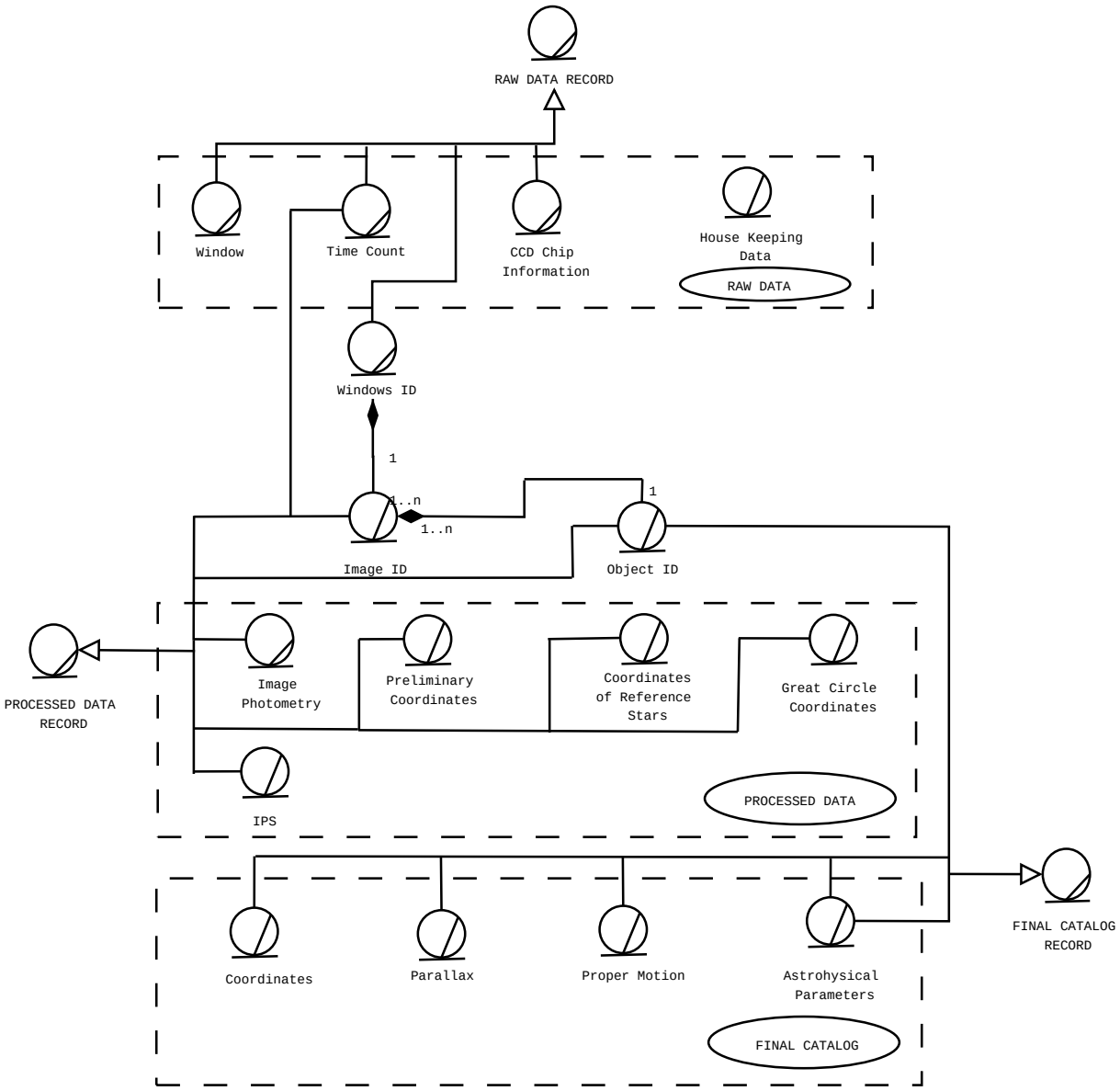

Figure 4.7: The general conceptual data model for entities. The attributes of the entities are omitted. All entities can be attributed to the three "general" entities: raw data record, processed data record and final catalog record. Window ID, Image ID and Object ID serve as the key attributes ("identifying entities") for "general" entities.

applications, but we have to avoid the n-tier architecture for time-critical applications.

4.4 The Model of the Data Retrieval

The data retrieval from the database can be made by two methods:

- a request from the application will retrieve the single data row in the single transaction (single-row data retrieval method),

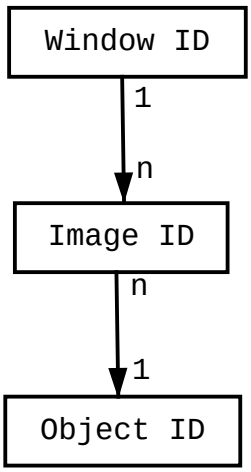

Figure 4.8: Identifying entities: Window ID, Image ID and Object ID.

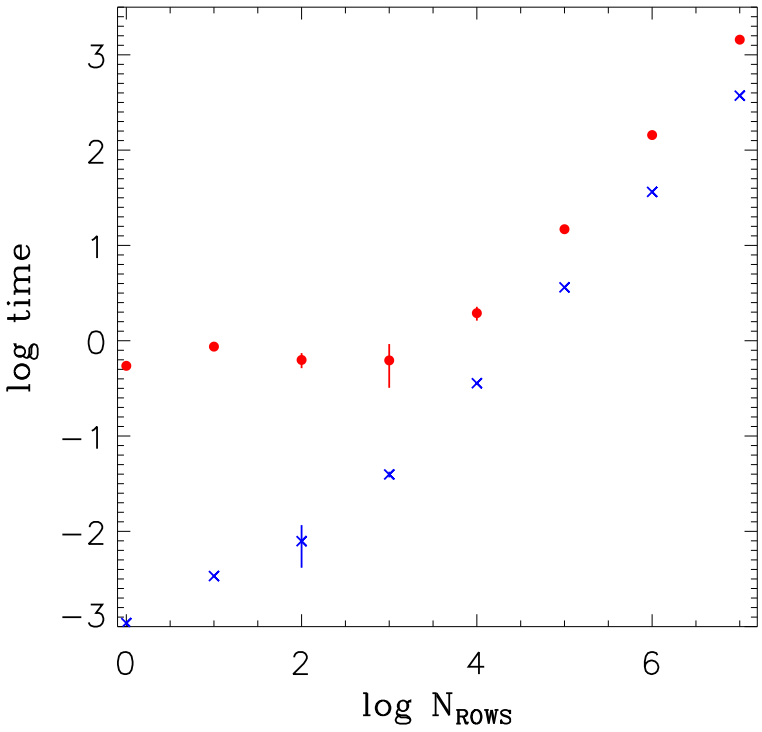

Figure 4.9: Java and C++ comparison. Filled circles are test results for Java, crosses are test results for C. Error bars are shown. The time is in sec.

- a request from the application will retrieve multiple data rows in the single transaction (multiple-row data retrieval method).

The results shows that time-critical applications have to avoid the single-row data retrieval. As we see in Fig. 4.10, the single-row method delays the data retrieval signifi-

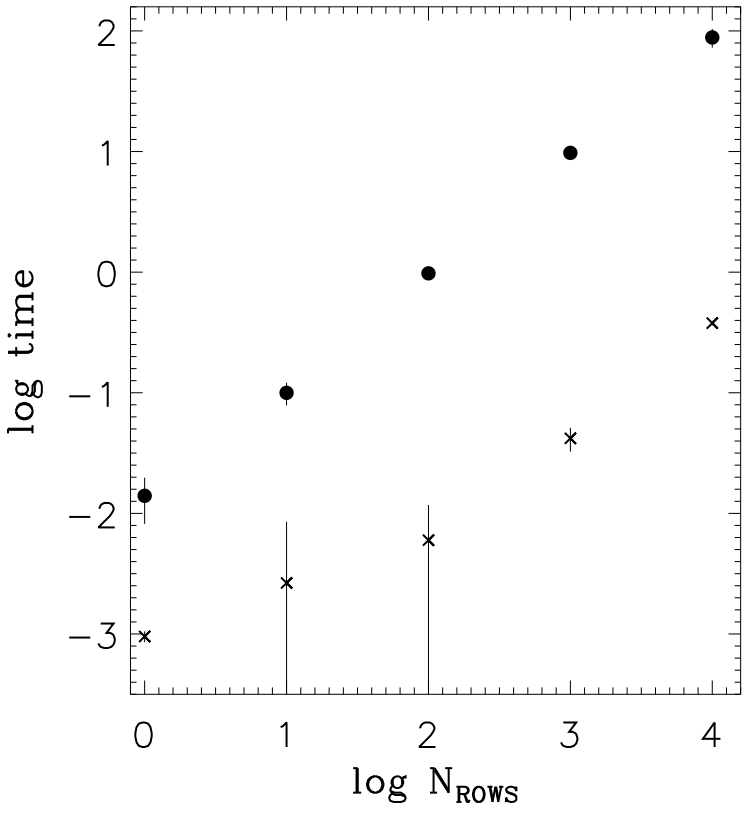

Figure 4.10: The single-row and multiple-row data retrieval. Circles are results for the single-row data retrieval, crosses are results for the multiple-row data retrieval. Time in sec. Error bars are shown.

cantly.

4.5 The Model of the Data

The data flows, data and data processing generated by an astrometric space mission are defined by the principle layout of instruments as well as by the transmission of data from the satellite to the SOC. We define the entities which describe the data by the following groups:

- **Raw data:** data records generated on board of the satellite and transmitted to the SOC. These data include scientific measurements (CCD images of observed objects) and information on the current status of the instruments and the satellite (House-Keeping data).
- **Processed data:** any data generated at the DPC by pipeline except the final catalog.

- **Final catalog:** result of the work of pipeline and other application programs using raw and processed data. This is the principle result of the mission which will be made available to the broad scientific community.

One of the problems which we meet here is the problem of the organization of the incoming data. Fig. 4.5 shows the initial approach of the conceptual model of the raw data. As we can see, the Window entity is a week one and can not be used without an identification with two other entities: Time Count (when this Window was red out on the focal plane of the satellite) and the CCD chip information (where this Window was observed).

It is not practical to identify the Window entity with the use of 5 attributes of 2 different entities. It is necessary to improve the situation at this early stage of the development of the DPC. To make the improvement we have to introduce a new entity: Window Identifier, which will be used as a key for the access to all three entities: the Window, the Time Count and the CCD chip information (see Fig. 4.6).

The data processing will create a number of new entities. We have to introduce these new entities following the most general description of the data processing (Fig. 4.11). Four initial entities (Window, CCD chip information, Time Count and the House-Keeping Data) were already described in the previous chapter.

Window Identifier. The Window Identifier will be used as a key to find the raw data record (see Fig. 4.7). The Window Identifier will be assigned on ground in the SOC or in the DPC. Estimated from the data volume coming from the satellite we need two integer values to store the Window Identifier.

Image Parameter Set. Image Parameter Set (*IPS*) is a collection of values describing the size and the form of the image which was obtained by **Pixel Data Processing** for each Window. The final set of these parameters depends on the algorithm used to process the data. It is important to note that the Pixel Data Processing produces more than one image from a single Window in some cases (10% of all windows). The case (b) in Fig. 2.8 shows an example of a Window with two Images.

Image identifier. The image identifier will be generated by a **Pixel Data Processing** on the basis of the Window Identifier and will serve as a reference for any entities which belong to the processed data. This is the analog of the Window Identifier for the raw data.

Image photometry. Directly from the pixel array of the Window **Pixel Data Processing** will estimate a set of values which will characterize the brightness of the object observed in the Window. As well as the IPS the composition of the Image Photometry attributes will depend on algorithms used by the Pixel Data Processing and other applications.

Preliminary Coordinates. Preliminary coordinates are the result of **Attitude Reconstruction** and consist of four values describing the position and mean errors of the position of the image on the sky. These coordinates are necessary to produce a preliminary identification of the image.

Great Circle Coordinates. This entity will be produced by the **Great Circle Reduction** process, and appears with DIVA/AMEX only. GAIA will skip this process. Nevertheless, to create a general scheme for both missions we have to keep this process

in the scheme. Great Circle Coordinates are one-dimensional coordinates of the center of the image in a coordinate system which is defined such that it is central to the strip which the satellite will scan for a short period of observations. In half a year such Great Circles will cover the whole sky and each object will have a number of Great Circle Coordinates in different coordinate systems. The combination of these one-dimensional coordinates will make it possible to construct the 2-dimensional position of each object.

Object identifier. The **Preliminary Identification** based on the Preliminary Coordinates will assign to each image an unique Object Identifier, which will serve as a reference for the final catalog of the mission. As a number of objects can not be identified in the coarse Preliminary Identification Catalog or can be wrong identified, all Object Identifiers will be newly generated by the **Object Recognition** based on the final Coordinates, Image Photometry and other entities at the very end of the mission .

Coordinates of Reference Stars. Coordinates of Reference Stars are determined by the **Sphere Reconstruction** based on the Great Circle Coordinates and used to find the 2-dimensional position of each object.

Coordinates (or, more accurate, positions on the sky), **Parallaxes** and **Proper Motions** are produced by the **Astrometric Parameter Determination** process and they are the most valuable result of the mission.

Astrophysical parameters. Astrophysical parameters like the brightness of the object and others will be defined at the very end of the mission by the **Astrophysical Parameter Determination** process. The exact list of the astrophysical parameters is defined by the abilities of the instruments on board of the satellite.

Entities described above can be divided on entities based on the raw data, processed data and the final catalog. Some of the entities from raw data will be used to produce the final catalog (Time Count, see Fig. 4.4). We have to note as well that 3 entities play a special role in the data processing (Fig. 4.7). These are Window Identifier, Image Identifier and Object Identifier. They are used as a reference for any other entity in the DPC except House-Keeping Data. The relationships between these entities (Fig. 4.8) are defined by the observational strategy of the scanning satellite.

4.6 Summary

The conceptual model is derived in this chapter determines the construction of the DPC. It flows from the concept of the astrometric scanning satellite and based on the astrometric data processing. The next step in the development of the DPC is the logical design of the DPC which will be made in the next chapter.

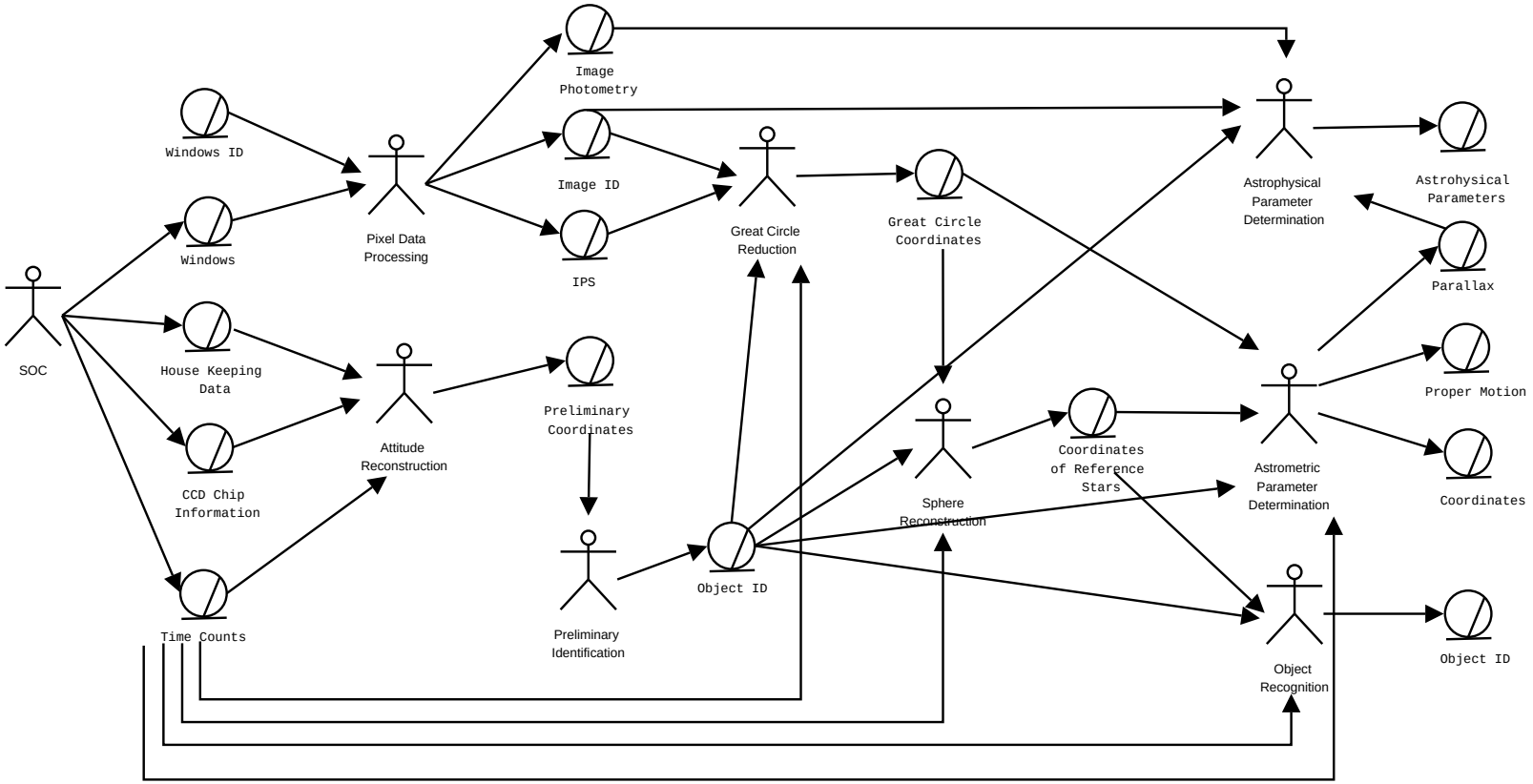

Figure 4.11: The UML model for the data processing in the DPC.

Chapter 5

The Logical Design of the DPC

The logical design of the DPC is dedicated mainly to the logical design of the database of the DPC. To simplify the design of the logical structure of the database we will make a choice of the data storage approach at this chapter and not in the chapter dedicated to the physical design of the DPC. This will make it easier to develop the scheme of the database and to describe the way applications will access the data stored in the database.

5.1 The Choice of the Data Storage Approach

We can organize the data storage as a file system, with the use of a relational DBMS or an object-oriented DBMS. Let us describe the differences between these possibilities and how they answer to the requirements to the data storage formulated in the previous chapter.

File system. At the first glance, a properly organized file system with the storage of data as binary files is the fastest solution. Nevertheless, we need to implement two properties of the data storage which will make this solution hardly usable: the huge data volume and fast changes in the processed data (90% of the data volume will be occupied by processed data in case of GAIA, as we will see later). For the processed data we must implement a way to change an existing data record inside a file or to insert a new data record into an existing file. To do this we have to read and rewrite the whole file or to create an index system for data records. If we take into account all these requirements we will see, that we must create a system which will be similar to a DBMS (the special format for data storage files, the indexes, the direct access to rows in the file). Instead of assembling a new DBMS we can use a commercial DBMS.

ODBMS. An object-oriented DBMS is a natural way to study data with complicated structure. Nevertheless, we have to abandon this possibility, because of a relatively slow response to requests in case of an ODBMS compared with a RDBMS (see, for example, [Thakar et al., 2003] for the test of the performance of ODBMS and RDBMS for the SDSS archive. This work analyses the typical request to astronomical data). Additionally, we need to organize data not by the object but by the attribute of the object (for example, time counts). Finally, we need to respond on a request from pipeline - the main user of the database. Let us suppose that we use ODBMS to store data. In this case each

observation of a star on the sky is stored as an object with some parameters (parameters of object are entities with attributes from the previous chapter). The pipeline needs to take from the database a collection of the same parameters from all objects within some period of observation but not theses objects themselves (not the complete information about an observation). In the case of ODBMS we have to retrieve all objects which will satisfy to our request and, in the next step, to select the required parameters from the collection of objects.

RDBMS. In the case of an RDBMS we are able to access all the data organized in simple tables with the ability to select data by different parameters and an ability to speed up the request with the use of indices. Nevertheless, we have to keep in mind the possibility to lose some information about links between parameters. Raw data, photometry, window centroids and other data types will be organized as detached tables and we must provide additional cross-referenced tables to keep the information about the data processing chain (for example, we have to keep the link between the window and the image parameters set derived from this initial window). As a result, the overall data volume will increase.

Let us compare the advantages and disadvantages of both approaches for an astrometric space mission. We follow the comparison scheme of [Jordan, 1998]:

	ODBMS	RDBMS	in the case of DIVA/AMEX/GAIA
the model of the data	classes, objects, methods	tables, rows, triggers, stored procedures	Most of the work will be done with massive data chunks with the request for the same entity for all objects, all entities for the same object on the sky will be requested only at the end of the mission. The preferred approach is RDBM.
applications	C/C++, Fortran, Java, etc.	4GL, QBE-languages, C/C++, Java, etc.	Some RDBMSs propose 4GL and QBE languages. Generally RDBMS has a better choice of APIs and user tools. The preferred approach is RDBM.
support of objects	full	limited, SQL3	The preferred approach is ODBMS.
support of normalization	can be implemented	native	The preferred approach is RDBMS.
data access	object driven	database driven	As we need an access to an attribute of all entities the database-driven model is preferred. The preferred approach is RDBMS.
data cast	usually not required (the same for data types as for applications)	required	The direct data cast from the data type of the DBMS to the data type of the application is required in the case of RDBMS. The preferred approach is ODBMS.

Figure 5.1: The logical structure of the database.

As we can see, RDBMS is be the better solution in our case.

The choice of the data storage approach is a balance between effectiveness of the solution for the data processing and cost for this solution. The most effective is a self-developed DBMS, but we must reject this solution as we have no resources to create a new DBMS. We will use the second possible solution – a commercial RDBMS which is possible to use for free for noncommercial projects.

5.2 The Logical Structure of the database

The logical structure of the database is based on the conceptual model of the data processing described in Chapter 4. From Fig. 5.1 we see that entities can be grouped in four main tables for the House-Keeping Data, the raw data, processed data and the final catalog.

5.3 The Problems of the Logical Design.

The structure shown in Fig. 5.1 is simple and clear, but the proposed structure can not be implemented for the real database installed in the DPC. Let us review the problems which will arise with such a scheme.

The lack of normalization. The **Processed Data** table is a combination of entities from the processed data, the raw data (**Window Identifier**) and the final catalog (**Object Identifier**). To identify a row in the **Processed Data** table we must use 6 attributes (see Fig. 5.1), but most attributes in the table depend on **Object Identifier** only. As a result, the table will not be in the second normal form of a relational database.

NULL values. For a number of rows in the **Processed Data** table and the **Final Catalog** table we will have NULL values for attributes. For example, at the beginning of the mission we will have no 2D coordinates which will be determined only half a year later.

Non-optimum data processing. Let us suppose that the next portion of data is generated by one of the processes of the pipeline. To insert this portion of data into the database we will have to use an UPDATE operation which is not as fast as the INSERT operation. To select a data portion from the database by the request of the application we need to browse a huge table (Processed Data) whereas the application will need data from only a few columns in the table.

5.4 Summary

The logical design must be upgraded to better satisfy the requirements from the applications. This means that we have to make a detailed description of the requests from applications to the database and to create a design for the database which will satisfy these requests in an optimum way. The direct use of the conceptual model (Fig. 4.7) for the logical scheme of the database is not the final step in the development of the database for the DPC. In the next chapter dedicated to the physical design of the DPC we have to transform the logical scheme of the database to a physical scheme which fulfills the requirements of the data processing to the database.

Chapter 6

The Physical Design of the DPC

The Physical Design of the DPC will consist of two main parts: the design of an optimum database scheme in accordance with the requirements from applications and the design of the DPC which includes hardware/software components.

We start with a more detailed description of entities used in the data processing. The description of processes will help to generate an optimum scheme of the database.

6.1 Attribute Domains the Data. The Required Precision

We have to define domains for each attribute. The domain comes from the requirements on the precision of the data.

Domains for **Time counts**, **House-Keeping data**, **CCD chip information** and **Window** were described previously (see Section 3.2.2). We should add, that the Window for AMEX/DIVA can be stored into VARCHAR variable with a maximum 2184 characters (see Section 3.2.2, each pixel can be stored in 2 characters), whereas the GAIA Window will much smaller and will need no more than 18 characters. The full-chip windows, which cover a whole CCD chip will be stored as BLOB values.

The description of entities for processed data and the final catalog was already done (see Section 4.5). A few notes on some entities are appropriate.

Preliminary Coordinates,**Great Circle Coordinates**, **2D Coordinates** and final **Coordinates** will required with up to 0.1 μas precision (in the case of GAIA) and will range from 0 up to 360 degrees. The DOUBLE for each coordinate will be used and the INTEGER value for error of the coordinate.

Parallaxes. Parallaxes will range from 1 arcsec down to μas. INTEGER is enough to store.

Proper motions. Proper motions will be stored in two INTEGER values with two INTEGER values reserved for errors.

Astrophysical parameters. Astrophysical parameters like the brightness will be defined at the very end of the mission. DOUBLEs will be used to store them.

Table 6.1: Characteristic time intervals for processes in the case of DIVA/AMEX.

Process	Frequency of the process	Critical Time
Initial Data Insert	daily	1-2 hours
WID Generation	daily	1-2 hours
Pixel Data Processing	daily	4-5 hours
Attitude Reconstruction	daily	4-5 hours
Preliminary Identification	daily	4-5 hours
Great Circle Reduction	once per 3 cycles	6-12 hours
Sphere Reconstruction	once per half a year	half a year
Astrometric Parameter Determination	once per half a year	half a year
Object Recognition	once per mission	half a year
Astrophysic Parameter Determination	oncer per mission	half a year

The domains for the entities are summarized in Table 6.3.

6.2 Optimization of the Database Structure

To provide an optimum scheme for the database we start from the description of processes which select data from the database. As the response time for time-critical applications is the most important parameter, the "optimum" database structure must be an optimum from the point of view of applications in our case.

All applications, which are described in the conceptual model (see Fig. 4.11) can be divided into two main groups: daily processes, like Pixel Data Processing, Attitude Reconstruction, Great Circle Reduction and processes, which will be run once per half a year (see Table 6.1). Each process is characterized with it's *frequency* – the typical time interval between two consequent run of this process, and it's *critical time* – the time interval reserved for the work of the process. The process must finish it's work on the data within the *critical time* interval.

We begin with the general description of the time required for the application to retrieve data from the database.

6.2.1 Theoretical Basis for the Design of the Optimum Structure

At the beginning of the design of the optimum physical structure for the database we have following information:

- attributes $\{S_i\}$, where each S_i belongs to some domain (D_j). There are some of attributes which are used by processes in the DPC accompanied by an another

attribute in the most cases. To simplify the development of the database structure we will join such attributes and will consider them further as a single attribute (a composite attribute). For example, instead of two attributes **Window Identifier 1**, **Window Identifier 2** we will use a single attribute **Window Identifier**. This is due to the fact that during the step of the most general description of the processes and their work with the data we will not need to consider each attribute separately. Each process has to retrieve the whole entity as it is described in Fig. 4.11;

- tables $\{T_i\}$, where T_i is defined on the attributes S_i designed on the basis of relationships between attributes and operations on the data. $T_i = [S_1, S_2, \ldots, S_j, \ldots]$;
- operations $\{R_i\}$, where R_i is one of the operations which the process has to perform with the data in the database. The typical R_i in the case of our database is the SELECT or the INSERT statement. Let us suppose, that the operation R_i has to select all rows from the database, the row must consist of attributes $\{S_1, S_2, ..., \ldots, S_j, \ldots\}$, the condition for the operation is that the attribute S_1 belongs to the subdomain D_1 and the attribute S_3 belongs to the subdomain D_3. Hereafter we will use a formal description of such an operation:

  $$R_i = \{< \text{type of the operation} >; < \text{list of attributes} >; < \text{conditions} >\},$$

  in our case

  $$R_i = \{SELECT; \{S_1, S_2, ..., \ldots, S_j, \ldots\}; S_1 \in \{D_1\}, S_3 \in \{D_3\}\}.$$

  If the subdomain is an interval, the condition is described as $S_1 \in [D_1^1, D_1^2]$;
- a set of ppi-keys $\{K_i\}$, where K_i is a primary key or a partitioning key or an indexing key.

A representation with the use of graphs of the structure of data can be used (see, for example, Fig. 4.6 for the raw data entities with attributes). The full description of entities with attributes is a multigraph Γ_0, where each attribute is represented by a vertex, a link between two attributes forms an edge of the graph. The optimization of the database scheme means that we have to find subdivisions of the initial multigraph Γ_0, such, that operations $\{R_i\}$ will take a minimum time on this subdivision. As result we will have subgraphs $\{\Gamma_i\}$ and, finally, we will form table T_i from each subgraph Γ_i.

To estimate the time required for operations we will use the time cost function (TCF) – i.e., the estimation of the time required to perform an operation on the data. The time cost function for operations $\{R_i\}$ depends on attributes selected by request and keys which exists for these attributes:

$$\text{TCF: } f(R_j) = f(\{S_i\}_{R_j}, \{K_l\}_{S_i}).$$

We will use the theoretical description of the TCF for SELECT and INSERT operations (see, for example, [Kulba et al., 1999] for the description of the theoretical modeling of the TCF). As it was mentioned previously (Chapter 3), there will be no UPDATE operations for DPC. The true form of the TCF will depend on the organization of the query optimizer for the true DBMS, nevertheless we can estimate the TCF based on the principles of the relational algebra and a very simple assumption for the form of the TCF, which we are listing below:

1. $R_1 = \{SELECT; S_j; \}$. The operation R_i retrieves all rows for the attribute S_j. In this case the TCF is just the time required to take all records from some physical device (hard disk).

$$f(R_1) = \sum_{i=1,N_j} t_r = N_j\, t_r,$$

where t_r is the time required to take one row from the data storage (each row consists of the attribute S_j only);

2. $R_2 = \{SELECT; S_j; S_j \in [D_j^1, D_j^2]\}$. The operation R_2 retrieves all rows for the attribute S_j, which satisfy to the range circumstance $S_j \in [D_j^1, D_j^2]$. D_j^1 and D_j^2 are the beginning and the end of the interval which is defined on the domain D_j. In the worst case we have to retrieve all records and check the requirement for any record, so that the TCF becomes

$$f(R_2) = \sum_{i=1,N_j} (t_r + t_{check}) = N_j\, (t_r + t_{check}),$$

where t_{check} is the time required to check the row satisfies the requirement. If we have an appropriate index for this attribute we only have to find the begin and the end of the ordered attributes:

$$f(R_2) = 2\, C\, \log_2 N_j + \sum_{i=1,M} t_r = 2\, C\, \log_2 N_j + M\, t_r,$$

where C is the time required to check an index record for one row, M is the number of rows retrieved by the statement (see [Arsenev & Yakovlev, 2001], for example);

3. $R_3 = \{SELECT; S_1, S_2; S_2 \in \{S_1\}\}$. For any row of the attribute S_2 we have to check the existence of the corresponding row of the attribute S_1.

$$f(R_3) = \sum_{i=1,N_2} \sum_{j=1,N_1} (2\, t_r + t_{check}) = N_1\, N_2\, (2\, t_r + t_{check}).$$

It is the worst case for the execution of such a query, if the attribute S_1 has no indices.

4. $R_4 = \{INSERT; S_1; \}$. $f(R_4) = N_1\, t_{insert,S_1}$.

Other queries can be represented as a combination of these simple subqueries with their TCFs. It is important not to confuse the time required to take one row from the data storage (t_r) with the time required to retrieve the record of the same size from the physical device, because t_r includes delays due to the database management system. Let us suppose that we have a subgraph Γ_i which is defined on attributes $\{S_1, \ldots, S_n\}$. The operation R_i works with attributes $\{S_k\}$, where k is a running number of the attribute. There are following possibilities in the case of the SELECT statement:

1) $\{S_k\} \in \Gamma_i$. This is the optimum subgraph for the operation. The time cost function is based on the estimations $1 - 4$.

2) $\exists S_l : S_l \in \{S_k\}, S_l \notin \Gamma_i$. This case requires an additional select of the data for the attribute S_l and the join of tuples for the attribute S_l with tuples for the rest of attributes. We will not consider this case and will reject it.

3) there is no attribute from $\{S_k\}$ in Γ_i. In this case the value of the time cost function is set to 0.

In the case of the INSERT statement attributes are allowed to belong to different graphs ($\{INSERT; \{S_i\};\}$, $\{S_1, \ldots, S_j\} \in \Gamma_k$, $\{S_{j+1}, \ldots, S_n\} \in \Gamma_l$, $\Gamma_l \cup \Gamma_k = \emptyset$). The case of the graph (table) has more attributes than the INSERT operation ($\{INSERT; \{S_i\};\}$, $\exists S_l : S_l \in \{S_i\}$, $S_l \notin\in \Gamma_k$) is restricted. This is due to the absent of the UPDATE operation for the database of the DPC and our intend to exclude any UPDATE operation from the consideration.

We described the attributes previously in Chapter 3, Chapter 4 and this Chapter. The attributes are summarized in Table 6.3, where the domains for each attribute are reviewed, and in Table 6.4, where the origin of each attribute is shown.

On the basis of the description of attributes $\{S_i\}$ and the estimation of the time cost function for operations $\{R_i\}$ we have to find a set of subgraphs $\{\Gamma_i\}$, which will satisfy a requirement of the minimization of the combined TCF.

6.2.2 Queries from Applications to the Database

The description of processes which will work with data is shown in Table 6.2. Most of processes have two operations: INSERT and SELECT.

To construct the time cost function for each possible subgraph Γ_i we have to assign a weight to each process. For example, Initial Data Insert and Pixel Data Processing will run every day and must be completed within a limited time interval, whereas Sphere Reconstruction can be run only every half a year of the mission and has a lower priority compared to Pixel Data Processing. As we can see from Table 6.1, the five daily running processes are the most important processes which define the structure of the database. These processes will have the highest priority. The weight for each process is estimated as

$$w_i = \frac{\nu_i}{\sum_{j=1}^{N} \nu_j},$$

where ν_i is the frequency of the process i (see Table 6.1) and N is the number of processes.

The weighted sum of the time cost functions for each process is minimized over all possible graphs (i.e., over all combinations of attributes).

Table 6.2: Queries of the data processing.

Process	**Query to the database**	**Dependences between attributes**
Initial Data Insert	$\{INSERT; TC, HK, CI, W;\}$	$W \rightarrow TC, CI \rightarrow TC, W \rightarrow CI$
Generate Window Identifier	$\{SELECT; TC, CI, W;\}$ $\{INSERT; WID, TC, CI, W, PRG;\}$	$TC \rightarrow WID, CI \rightarrow WID$ $W \rightarrow WID, WID \rightarrow PRG$
Pixel Data Processing	$\{SELECT; WID, TC, CI, W; WID \in \{WID_i\}\}$ $\{INSERT; IID, TC, CI, IPS, IPH, PRG;\}$	$IID \rightarrow WID, IID \rightarrow PRG$ $IPS \rightarrow IID, IPH \rightarrow IID$
Attitude Reconstruction	$\{SELECT; HK, IID, TC, CI; IID \in \{IID_i\}, HK \in \{HK_i\}\}$ $\{INSERT; PRC, PRG;\}$	$PRC \rightarrow IID, PRC \rightarrow PRG$
Preliminary Identification	$\{SELECT; IID, PRC; IID \in \{IID_i\}\}$ $\{INSERT; OID, PRG;\}$	$OID \rightarrow IID, OID \rightarrow PRG$
Great Circle Reduction	$\{SELECT; IID, TC, CI, IPS; IID \in \{IID_i\}\}$ $\{INSERT; GCC, PRG;\}$	$GCC \rightarrow IID, GCC \rightarrow PRG$
Sphere Reconstruction	$\{SELECT; OID, GCC;\}$ $\{INSERT; COO, PRG;\}$	$COO \rightarrow OID, COO \rightarrow PRG$
Astrometric Parameter Determination	$\{SELECT; OID, COO, TC;\}$ $\{INSERT; PAR, PM, PRG;\}$	$PAR \rightarrow OID, PM \rightarrow OID$ $PAR \rightarrow PRG, PM \rightarrow PRG$
Object Recognition	$\{SELECT; OID, COO, TC;\}$ $\{INSERT; OID_2, PRG;\}$	$OID_2 \rightarrow OID, OID_2 \rightarrow PRG$
Astrophysical Parameter Determination	$\{SELECT; OID, COO, PAR, PM, IPH;\}$ $\{INSERT; AP, PRG;\}$	$AP \rightarrow OID, AP \rightarrow PRG$

Table 6.3: Attributes for the database with attributes' domains.

Entity	Introduced by a process	Domains
Time counts (TC)	SOC	2(INTEGER)
House-Keeping Data (HK)	SOC	11(DOUBLE)
CCD chip information (CI)	SOC	3(SMALLINT)
Window (W)	SOC	VARCHAR
IPS (IPS)	Pixel Data Processing	4(INTEGER)
Window ID (WID)	Raw Data Insert	2(INTEGER)
Image ID (IID)	Pixel Data Processing	2(INTEGER)
Object ID (OID)	Preliminary Identification Object Recognition	2(INTEGER)
Image Photometry (IPH)	Pixel Data Processing	N_{BAND} 3(INTEGER)
Preliminary Coordinates (PRC)	Attitude Reconstruction	2(DOUBLE),2(INTEGER)
Great Circle Coordinates (GCC)	Great Circle Reduction	INTEGER, SMALLINT
Coordinates (COO)	Sphere reconstruction	2(DOUBLE),2(INTEGER)
Parallaxes (PAR)	Astrom. Parameter Determination	2(INTEGER)
Proper Motions (PM)	Astrom. Parameter Determination	4(INTEGER)
Astrophys. Parameters (AP)	Astrophys. Parameter Determination	4(INTEGER)
Program Identifier (PRG)	each process	SMALLINT, CHAR(100)

6.2.3 The Results of the Optimization of the Database Structure

The number of stars observed by the satellite per scan is approximately 700 000 (in the case of DIVA, see [TD0201-05, 2002]). The number of windows transmitted to the ground per scan is approximately 1.6 millions ([TD0284-01, 2002]). The later value is the number of rows which must be inserted into the database and must be selected from the database

Table 6.4: Relationships between attributes.

Entity	Parent Entity	Child Entity
Time counts		CCD chip information, Window
House-Keeping Data		Preliminary Coordinates
CCD chip information	Time counts	
Window	Time counts	
IPS	Program Identifier	
Window ID	Program Identifier	Window, CCD chip information, Time counts
Image ID	Window ID, Program Identifier	IPS, Image Photometry, Preliminary coordinates
Object ID	Image ID, Program Identifier	Coordinates, Parallaxes, Proper Motion, Astrophysical Parameters
Image Photometry	Image ID, Program Identifier	
Preliminary Coordinates	Image ID, Program Identifier	
Coordinates	Object ID, Program Identifier	
Parallaxes	Object ID, Program Identifier	
Proper Motions	Object ID, Program Identifier	
Astrophysical Parameters	Object ID, Program Identifier	

Table 6.5: An optimum database scheme. i is the number of the solution, Γ_i is a subgraph.

i	Γ_i	Comment
1	TC, OID, COO	
2	IID, OID, PRG	
3	OID, AP, PRG	Astrophysical Parameters
4	OID, IPH, COO, PAR, PM	
5	OID, GCC	
6	OID, COO, PRG	Sphere Coordinates
7	OID, PAR, PM, PRG	Astrometrical Parameters
8	IID, GCC, PRG	Great Circle Coordinates
9	IID, TC, CI, IPS	IPS
10	IID, PRC	
11	TC, CI, W	Initial Data Table
12	IID, PRC, PRG	Preliminary Coordinates
13	TC, HK, CI, W	
14	IID, TC, HK, CI	
15	WID, TC, CI, W	Raw Data Table
16	WID, TC, CI, W, PRG	Raw Data Table
17	WID, IID, TC, CI, IPS, IPH, PRG	

by daily processes. This value was used to estimate a time required to process data:

$$t_{est} = \sum_{i=1}^{N} n_i^{rows}\, w_i\, f(R_i, \Gamma_j),$$

where N is the number of processes ($N = 10$, see Table 6.2), Γ_j runs through all possible combinations of attributes S_i (see Table 6.3 for attributes) and n_i^{rows} is the number of rows retrieved by the process i. Finally we have a set of all possible subgraphs defined on attributes with the estimation of time t_{est} corresponding to each subgraph. If we reject all degenerate cases (the estimated time is 0) and cases with maximum value of an estimated time ($\max_i t_{est}^i$, where i runs through all possible graphs), we will have only 17 subgraphs (see Table 6.5).

Based on these subgraphs we have to select a scheme which will an optimum join of subgraphs satisfying all processes. As we can see from the set of subgraphs, there are three types of data groups in three categories of tables: tables for raw (initial data), tables for the processed data and the final table. The final scheme of the database is completed and we start with the description of tables and attributes for each data group.

6.3 The Physical Design of the Database

The Physical Design realize the optimum data scheme and based on the Table 6.5. Three main groups in the schema are the raw data, the processed data and the final catalog.

I.Raw data

RDSW Raw Data Standard Windows. The majority of windows will relative small (7×13 pixels, see Chapter 2) and is stored packed in the VARCHAR column. Windows observed at SM1, SM2, SC1 and SC2 mosaics are stored at the same table. The origin of the window could be found from special column (Window type).

RDFW Raw Data Full Chip Window. This table will store a few procent (2% among all windows) of the large full-chip images. This is the only table in the database which will require a BLOB type.

RDCP Raw Data Child-Parent Chip Relation. The table stores links between the window identifier of the window on the first chip and window identifies of predicted windows (usually the first SM window is the parent window and the second SM window and SC windows – children).

RDHK Raw Data House-Keeping Data. The table stores technical data from the satellite: the relation between TDI counts (T1, T2) from RDSW and physical time, coordinates of satellite axes, satellite rotation rate and so on.

Table 6.6: The raw data tables.

Attribute	Designation	Domain
RDSW: Standard window size		
Window	NW1	INTEGER
Identifier	NW2	INTEGER
Time	T1	INTEGER
Count	T2	INTEGER
CCD chip X-coordinate	CX	SMALLINT
CCD chip Y-coordinate	CY	SMALLINT
Window type	WT	SMALLINT
Window	W	VARCHAR(2184)
RDFW: Full chip window		
Full chip window ID	F1	INTEGER
Time	T1	INTEGER
Count	T2	INTEGER
CCD chip X-coordinate	CX	SMALLINT
CCD chip Y-coordinate	CY	SMALLINT
Window	FW	BLOB
RDCP: Child objects		
Parent	PWN1	INTEGER
Window ID	PWN2	INTEGER
Child	CWN1	INTEGER
Window ID	CWN2	INTEGER
RDHK: House-Keeping Data		

Time	T1	INTEGER
Count	T2	INTEGER
Physical Time	PT1	INTEGER
(Onboard time)	PT2	INTEGER
Attitude angle 1	AA1	DOUBLE
Attitude angle 1 error	EA1	DOUBLE
Attitude angle 2	AA1	DOUBLE
Attitude angle 2 error	EA1	DOUBLE
...	...	...

II. Processed data.

PDPRG Processed Data Program Identifier. This table collects information for all versions of the pipeline and other programs and will be used as a reference of the sources of the processed data.

PDIPS Processed Data Image Parameters Set. The table stores the results of the pipeline treatment of Windows, mainly by the Pixel Data Processing part of the pipeline.

PDWI Processed Data Window - Image Identifiers Relation. This table will store links between the window identifier and identifiers of images extracted from this window.

PDIO Processed Data Image - Object Identifiers Relation. This table will store links between the image identifier and identifier of object associated with the image.

PDPRC Processed Data Preliminary Coordinates. This table will contain an information about preliminary coordinates of the object and an image identifier.

PDGCR Processed Data Great Circle Reduction. The table will store results of the GCR, i.e. coordinates of stars with errors and so on.

PDSR Processed Data Sphere Reconstruction. This table will contain 2 coordinates for each object and errors of coordinates computed as result of Sphere Reconstruction process.

PDOR Processed Data Object Recognition. The table will contain the final Object Identifiers resulted from the Object Recognition process.

Table 6.7: The processed data tables.

Attribute	Designation	Domain
PDPRG: Pipeline or other program ID		
Pipeline or program number	PRG	SMALLINTEGER
Description	DSC	CHAR(1000)
Version	VRS	SMALLINT
Realize date	RDA	DATE
PDIPS: Windows centroids		

Pipeline or program current number	PRG	SMALLINTEGER
Image	NI1	INTEGER
Identifier	NI2	INTEGER
Time	T1	INTEGER
Count	T2	INTEGER
Window centroid X	WX	INTEGER
Window centroid Y	WY	INTEGER
X coordinate error	XE	INTEGER
Y coordinate error	YE	INTEGER
Amplitude of the IPS in ADUs	AMP	INTEGER
Error of the amplitude in ADUs	AMPE	INTEGER
background of the image in ADUs	BA	INTEGER
Error of the background in ADUs	BAE	INTEGER
Shape parameter 1 in pixels	SP1	INTEGER
Error of the shape parameter 1 in pixels	SP1E	INTEGER
...	...	...
PDWI: Window - Image Identifiers Relation		
Pipeline or program current number	PRG	SMALLINTEGER
Image	NI1	INTEGER
Identifier	NI2	INTEGER
Window	NW1	INTEGER
Identifier	NW2	INTEGER
PDIO: Image - Object Identifiers Relation		
Pipeline or program current number	PRG	SMALLINTEGER
Image	NI1	INTEGER
Identifier	NI2	INTEGER
Object	NB1	INTEGER
Identifier	NB2	INTEGER
PDPRC: Preliminary Coordinates		
Pipeline or program current number	PRG	SMALLINTEGER
Image	NI1	INTEGER
Identifier	NI2	INTEGER
FOV identification	FOV	SMALLINT
Preliminary position, α	PALF	DOUBLE
Preliminary position, δ	PDEL	DOUBLE
PDGCR: Great Circle Reduction		
Pipeline or program current number	PRG	SMALLINTEGER
Image	NI1	INTEGER
Identifier	NI2	INTEGER
Position along the Great Circle, arcsec	GCC	DOUBLE
Error of the position, arcsec	GCCE	DOUBLE
Reference Great Circle number	RGC	SMALLINT
PDSR: Sphere Reconstruction		
Pipeline or program current number	PRG	SMALLINTEGER
Preliminary Object	NB1	INTEGER

Identifier	NB2	INTEGER
Position, α arcsec	SALF	DOUBLE
Position, δ arcsec	SDEL	DOUBLE
Error of position, α arcsec	SALFE	DOUBLE
Error of position, δ arcsec	SDELE	DOUBLE
Reference Great Circle	RGC	SMALLINT
PDOR: Object Recognition		
Pipeline or program current number	PRG	SMALLINTEGER
The final Object	NB1	INTEGER
Identifier	NB2	INTEGER
The Preliminary Object	PNB1	INTEGER
Identifier	PNB2	INTEGER
Failure flag	BBF	SMALLINT

III. Final catalog

FDASP Final Data Final Catalog of Astrometric Parameters. This table collects the final result of the astrometric data processing.

FDAPP Final Data Astrophysical Parameters.

FCEXI Final Data Cross-identification table with external catalogs.

FCEXD Final Data List of external catalogs table.

Table 6.8: The final catalog tables.

Attribute	Designation	Domain
FDASP: Final catalog of astrometric parameters		
Pipeline or program current number	PRG	SMALLINTEGER
Object	NB1	INTEGER
Identifier	NB2	INTEGER
Position, α	ALF	DOUBLE
Position, δ	DEL	DOUBLE
Position error, σ_α	EAL	DOUBLE
Position error, σ_δ	EDE	DOUBLE
Proper motion, μ_α	MUA	INTEGER
Proper motion, μ_δ	MUD	INTEGER
Proper motion error, σ_{μ_α}	EMA	INTEGER
Proper motion error, σ_{μ_δ}	EMD	INTEGER
Parallax	PAR	INTEGER
Parallax error	EPA	INTEGER
FDAPP: Final catalog of astrophysical parameters		
Pipeline or program current number	PRG	SMALLINTEGER
Object	NB1	INTEGER
Identifier	NB2	INTEGER

Final photometry, band1	FB1	SMALLINT
Final photometry error, band1	EFB1	SMALLINT
Final photometry, band2	FB2	SMALLINT
Final photometry error, band2	EFB2	SMALLINT
$\log g$	LGG	DOUBLE
...	...	...
FCEXI: Cross-identification with external catalog		
Pipeline or program current number	PRG	SMALLINTEGER
Object	NB1	INTEGER
Identifier	NB2	INTEGER
Catalog's ID number	CTN	SMALLINT
Catalog's name	CNM	VARCHAR(100)
FCEXD: List of external catalogs		
Catalog's ID number	CTN	SMALLINT
Catalog's description	CNM	CHAR(5000)

A number of tables will be included in this section: each external catalog (like Tycho-2 or 2MASS) which will be used for cross-identification will be inserted into the database as a separate table.

The set of the tables described in this chapter is not complete. We can update the number of tables as well as the number of attributes in each table. For example, the table for the **Coordinates for Reference Stars** was not considered (this table has an information about Reference Great Circles) as the size of this table is neglectable compared with other tables of the processed data. The only restriction is that we can not add new primary keys to the four primary keys which come from the four identifying entities described in the conceptual design of the database. In the next Section we will use the database scheme to write down the SQL statements which will be used by the pipeline.

6.4 Requests to the Database from the Data Processing

The required SQL statements can be found from the description of the processes made above, from the most general relation between data (Fig. 6.1) and from the description of the corresponding tables. This is the realization of the general description of the processes (see Table 6.2).

Initial Data Insert. Generate Window Identifier. The initial data insert will load data into four tables: RDSW, RDFW, RDHK and RDCP. It is better to combine the operation of the generation of the window identifier and the initial data insert in the single process.

```
INSERT INTO RDSW VALUES(NW1,NW2,T1,T2,CX,CY,WT,W)
INSERT INTO RDFW VALUES(NW1,NW2,T1,T2,CX,CY,W)
INSERT INTO RDCP VALUES(PWN1,PWN2,CWN1,CWN2)
INSERT INTO RDHK VALUES(T1,T2,PT1,PT2,AA1,EA1,AA2,EA2)
```

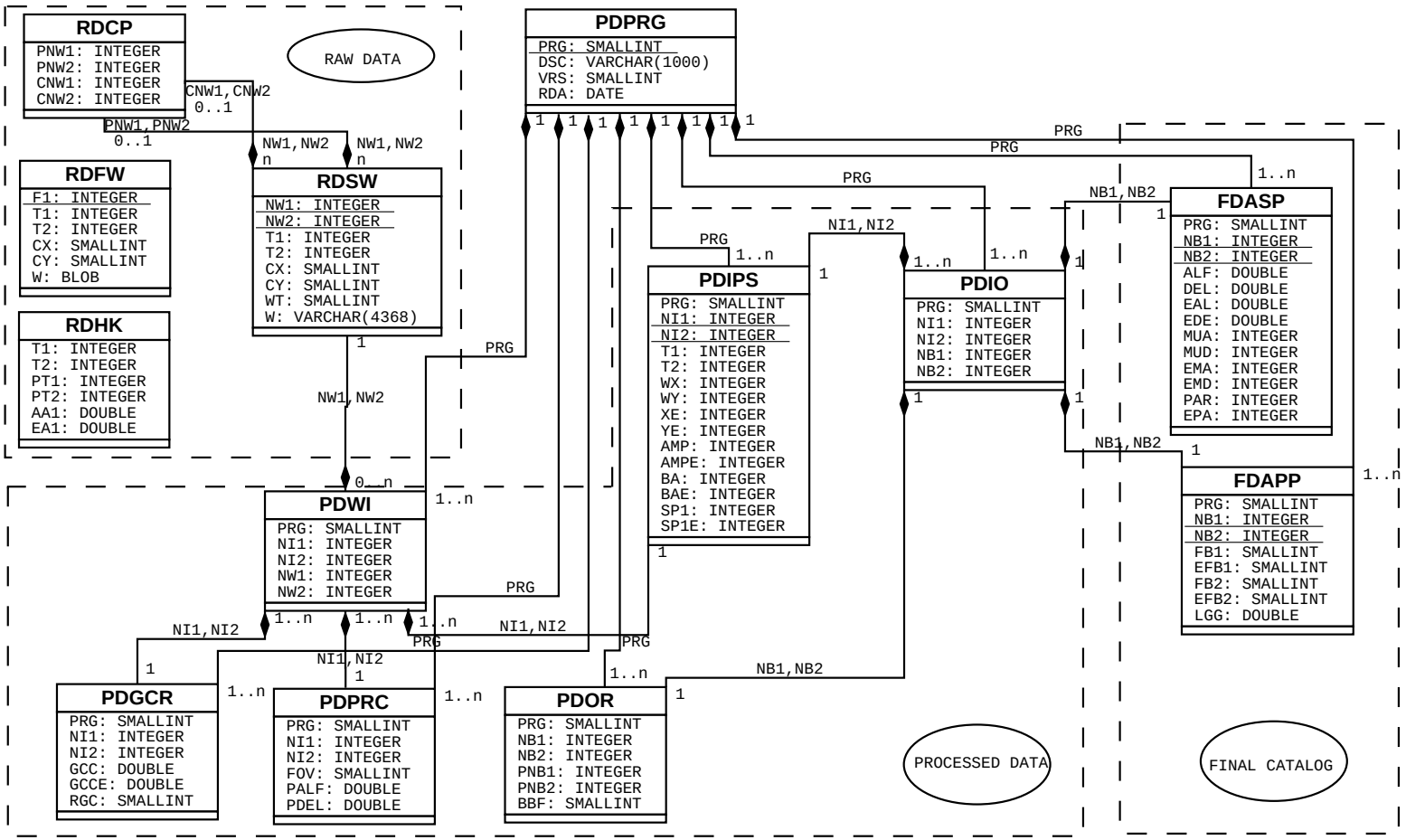

Figure 6.1: The UML scheme of the database. Raw Data, Processed Data and Final Catalog layers are shown. Some tables are not shown (PDSR, FCEXI, FCEXD).

The incoming data file will in fact consist of two files: the first with the House-Keeping data and the second with scientific data (the mixture of Standard Windows and Full Chip Windows). The last one has to be preprocessed before the data insert and divided into the three files (one file for each table of the scientific raw data). At this moment the newly generated window identifiers will be added to data files.

Pixel Data Processing. Pixel Data Processing, Attitude Reconstruction and Preliminary Identification are the most time critical statements among all. Pipeline has to select data from the database and process selected data before the next data portion comes from the satellite. The typical time interval for the whole action is approximately 2 hours for DIVA/AMEX type mission and approximately 24 hours for GAIA. Pixel Data Processing requires to select data for the current cycle of the observation only. To provide a fast data retrieval the window identifier generated at the previous step must have information about the scanning circle. The first part of the window identifier (NW1) is in fact the running number for the scanning circle. As result, the SQL select statement is very simple.

SELECT NW1, NW2, T1, T2, CX, CY, W FROM RDNW
WHERE NW1 >= *i* AND NW1 < *i+3* ORDER BY NW1, NW2;

where i is the running number of the scanning circle. The Pixel Data Processing will as well generate an Image Identifier for each image in the window. As a result, the insert statement is subdivided into the two inserts: IPS for each image and the relation Image Identifier - Window Identifier.

INSERT INTO PDIPS VALUES (PRG, NI1, NI2, T1, T2, WX, WY, XE, YE, AMP, AMPE, BA, BAE, SP1, SP1E)
INSERT INTO PDWI VALUES (PRG, NI1, NI2, NW1, NW2)

Attitude Reconstruction. For the attitude reconstruction two tables will be searched: PDIPS and RDHK.

SELECT PRG, NI1, NI2, T1, T2, WX, WY, XE, YE FROM PDIPS
WHERE NI1>=*i* AND NI1<*i+3* ORDER BY T1,T2
SELECT T1, T2, PT1, PT2, AA1, EA1, AA2, EA2 FROM RDHK
WHERE T1 >= T_1 AND T2 <=T_2 ORDER BY T1,T2

The data retrieved from both tables are measured in the same time interval. The preliminary coordinates will be inserted,

INSERT INTO PDPRC VALUES (PRG, NI1, NI2, FOV, PALF, PDEL)

Preliminary Identification. For the attitude reconstruction two tables will be searched: PDIPS and RDHK.

SELECT PRG, NI1, NI2, T1, T2, WX, WY, XE, YE FROM PDIPS
WHERE NI1>=*i* AND NI1<*i+3* ORDER BY T1,T2
SELECT T1, T2, PT1, PT2, AA1, EA1, AA2, EA2 FROM RDHK
WHERE T1 >= T_1 AND T2 <=T_2 ORDER BY T1,T2

The data retrieved from both tables were measured in the same time interval.

INSERT INTO PDIO VALUES (PRG, NI1, NI2, NB1, NB2)

Great Circle Reduction. The Great Circle Reduction will require IPSs for images within three scanning circles:

SELECT PRG, NI1, NI2, T1, T2, WX, WY, XE, YE FROM PDIPS
WHERE NI1>=i AND NI1<i+3 ORDER BY T1,T2

The results are one-dimensional coordinates of the objects:

INSERT INTO PDGCR VALUES (PRG, NI1, NI2, GCC, GCCE, RGC)

Sphere Reconstruction. The Sphere Reconstruction will access one-dimensional coordinates of objects for half year of observation and will produce 2-dimensional coordinates:

SELECT X.PRG, X.NI1, X.NI2, X.GCC, X.GCCE, X.RGC, Y.NB1, Y.NB2
FROM PDGCR X, PDIO Y
WHERE X.NI1=Y.NI1 AND X.NI2=Y.NI2 AND X.NI1>=$NI1_1$ AND X.NI1 <$NI1_2$
ORDER BY Y.NB1, Y.NB2

INSERT INTO PDSR VALUES (PRG, NB1, NB2, SALF, SDEL, SALFE, SDELE)

Astrometric Parameter Determination. The astrometric Parameter Determination will collect all information about positions of objects determined through the mission to find precise positions, parallaxes and proper motions for each object.

SELECT PRG, NB1, NB2, SALF, SDEL, SALFE, SDELE, RGC FROM PDSR
ORDER BY NB1, NB2

INSERT INTO PDSR VALUES (PRG, NB1, NB2, ALF, DEL, EAL, EDEL, MUA,
MUD, EMA, EMD, PAR, EPA)

Object Recognition. The final object recognition

SELECT PRG, NB1, NB2, SALF, SDEL, SALFE, SDELE, RGC FROM PDSR

INSERT INTO PDOR VALUES (PRG, NB1, NB2, PNB1, PNB2, BBF)

Astrophysical Parameter Determination.

SELECT X.PRG, X.NB1, X.NB2, X.ALF, X.DEL, X.EAL, X.EDE, X.MUA, X.MUD,
X.EMA, X.EMD, Y.NI1, Y.NI2, Z.T1, Z.T2, Z.AMP, Z.AMPE, Z.BA, Z.BAE
FROM FDASP X, PDIO Y, PDIPS Z
WHERE Y.NB1=X.NB1 AND Y.NB2=X.NB2 AND Z.NI1=Y.NI1 AND Z.NI2=Y.NI2
ORDER BY X.NB1, X.NB2

INSERT INTO FDAPP VALUES (PRG, NB1, NB2, FB1, EFB1, FB2, EFB2, LGG)

6.4.1 Keys and Indices

We are setting some general principles for the data placement in the database as follows:

Primary keys. We have 4 types of primary keys : artificial window identifier (NW1,NW2), time counts (T1,T2), image identifier (NI1,NI2) and object identifier (NB1,NB2). Each of them consists of 2 INTEGER values or a (SMALLINT, INTEGER) pair. It is possible to identify each window with a set of 5 fields (time counts - T1,T2; CCD chip information - CX,CY, CN), these values will be unique for each window. Nevertheless the use of a two-column primary key is better, because we will save disk space required to store an index and it will be easier to access the data.

Partitioning key. We will the second part of each identifier (NW2,T2,NI2, NB2) as a partitioning key to distribute the data through the nodes. As NW1, NI1 and NB1 will count the same value - a running number of scanning circles of the satellite, the data could be distributed so, that windows belonging to the same scanning circle will be shared by all nodes. The main requests from pipeline require data from the same scanning circle and, as a result of the data distribution, the request will load all nodes sharing data uniformly.

Coherence of partitioning keys. We set NW2 as the "basic" partitioning key. Any table having another partitioning key, i.e. NI2, NB2 and T2, will be distributed so, that the record due to the image or object will be placed at the same node with the parent window. This way we significantly reduce the number of transmission between nodes in the case of join statement between tables with different primary and partitioning keys.

Order Indexes. Most of the requests needs a huge amount of data ordered by some entity. To speed up the select of the data for requests the indices must be implemented for attributes which were not covered by primary or partitioning keys but will be used in a where-statement of requests.

The SQL statements above are not the final version for the data treatment, nevertheless we can construct a set of indexes for our tables on the basis of supposed requests to the database (most of them are really Primary Keys - PK). These indexes will

IWRDSW ascending and unique index on the columns NW1, NW2 of RDSW table (PK).

ITRDSW ascending and unique index on the columns T1, T2 of RDSW table.

IWRDCP ascending and unique index on the columns PWN1, PWN2 of RDCP table (PK).

IIPDIPS ascending and unique index on the columns NI1,NI2 of PDIPS table (PK).

ITPDIPS ascending and unique index on the columns T1,T2 of PDIPS table.

IIPDWI ascending and unique index on the columns NI1,NI2 of PDWI table (PK).

IIPDPID ascending and unique index on the columns NI1,NI2 of PDPID table (PK).

IBPDPID ascending and unique index on the columns PNB1,PNB2 of PDPID table.

The final scheme of the database is shown in Fig. 6.1.

6.5 The growth of the data with time.

As soon as we have designed the data structure for the mission's database we can estimate the data growth through the mission. For this we have to combine the scanning law, designed data structure and the processing chain. As an every half a year we propose to start an improved version of the pipeline which will produce the next iteration of processed data and as we have to store all processed data during the mission the increase of the data volume will be non-linear.

The transfer between data types can be estimated on the basis of the modeling of the satellite's behavior and the modeling of instruments. The initial record will be the standard chip window in 99% and full chip window in 1 % of the cases. The standard chip window will contain 2 or more images in 10 % of the cases (see [TD0284-01, 2002] for the DIVA/AMEX case). As a result the following formula can be applied to estimate the growth of the size of the database:

$$
\begin{aligned}
\mathbf{DBV(B,\ circle)} &= \int_0^{\mathbf{circle}} (f_{\text{raw}}(\mathbf{circle}) + f_{\text{processed}}(\mathbf{circle}) + f_{\text{final}}(\mathbf{circle}))\mathrm{d}\,\mathbf{circle},\\
f_{\text{raw}}(\mathbf{circle}) &= 0.99\,\mathrm{V(S/N)}\,(\mathrm{size(RDSW)} + \mathrm{size(RDCP)})\,\mathbf{circle}\\
&+ 0.01\,\mathrm{V(S/N)}\,\mathrm{size(RDFW)}\,\mathbf{circle}\\
&+ \nu(\mathrm{telemetry})\,\mathrm{size(RDHK)}\,\mathbf{circle},\\
f_{\text{processed}}(\mathbf{circle}) &= 1.21\,\mathrm{V(S/N)}\,(\mathrm{size(PDIPS)} + \mathrm{size(PDWI)} + \mathrm{size(PDPRC)})\,\mathbf{circle}\\
&+ 3.62\,\mathrm{V(S/N)}\,(\mathrm{size(PDGCR)})\,\Lambda(\mathbf{circle}, 3)\\
&+ 217.4\,V(S/N)\,(\mathrm{size(PDIO)} + \mathrm{size(PDSR)} + \mathrm{size(PDOR)})\,\Lambda(\mathbf{circle}, 183),\\
f_{\text{final}}(\mathbf{circle}) &= (\mathrm{size(FDASP)} + \mathrm{size(FDAPP)})\,\Lambda(\mathbf{circle}, 183),
\end{aligned}
$$

where **DBV** is the data volume in Bytes, **circle** is the running circle of observations, f_{raw}, $f_{\text{processed}}$, f_{final} are the data volume due to the raw data, the processed data and the final catalog correspondingly, V(S/N) is a number of the detected objects on the focal plane during the running circle of observations (this value depends on the Signal-to-Noise ratio), and

$$
\Lambda(a,b) = \begin{cases} 1, & \text{if (a/b - int(a/b))} = 0 \\ 0, & \text{if (a/b - int(a/b))} \neq 0 \end{cases}
$$

Table 6.9: The size of tables.

	DIVA/AMEX, B	GAIA, B	Update rate
Raw Data			
RDSW	2184	18	daily
RDFW	2096928		daily
RDCP	16		daily
RDHK	272		daily
Processed Data			

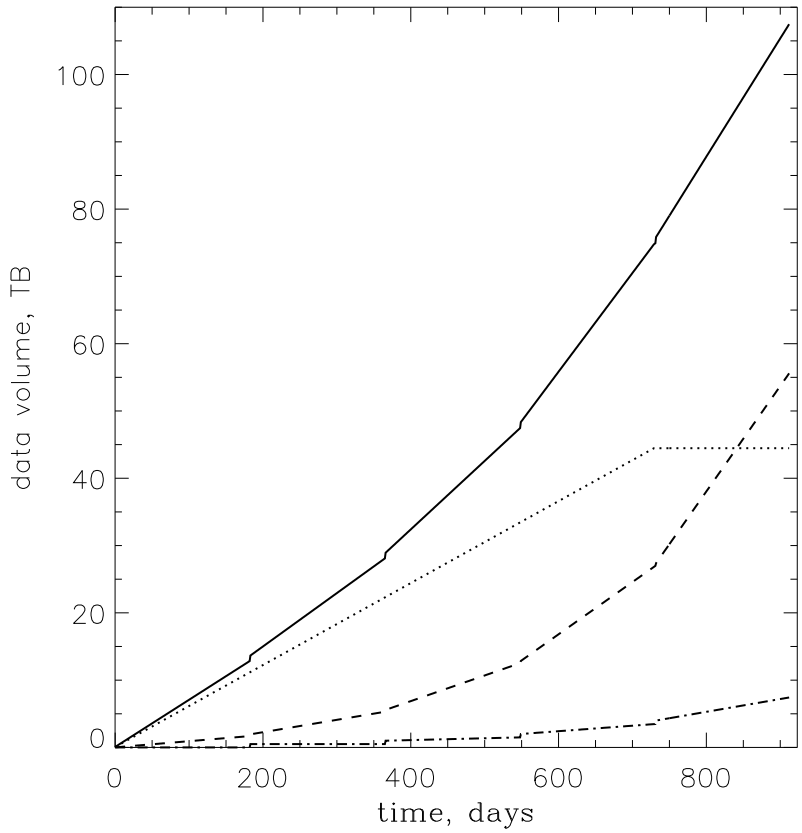

Figure 6.2: The growth of the data during the mission (DIVA/AMEX). The dotted line is the growth of the the raw data, the dashed line is the growth of the processed data, the dashed-dotted is the growth of the final catalog and the solid line is the combined data growth.

PDPRG	1014	daily/once per half a year
PDIPS	46	daily
PDWI	18	daily
PDIO	18	once per half a year
PDPRC	28	daily
PDGCR	28 | -	each 3 days
PDSR	44	once per half a year
PDOR	20	once per half a year
Final Catalog		
FDASP	66	once per half a year
FDAPP	98	once per half a year
FCEXI	112	once per mission
FCEXD	5002	once per mission

The function size(table) returns the size of each table in Bytes (see Table 6.9). As we can see, the data grows with the constant rate every day except in the case of half-year

processes. They increase the data volume every half a year. The non-linear data growth will occur because each new version of the pipeline will need to re-process the raw data obtained previously. In Fig. 6.2 we supposed that the new version of the pipeline runs every half a year and has to re-process previously obtained raw data for half a year.

6.6 Hardware & Software Components and Their Optimum Choice

In the ideal case the choice of the hardware and software components of the DPC is ruled by the needs of scientists and the personal to handle data in reasonable time. In practice one of the major factor in the choice of these components are financial constraints. As result the choice must be "optimum", i.e. the DPC has to consume the minimum resources and and the same time the DPC has to be sufficiently effective to accomplish the task.

6.6.1 A Shared-Nothing Linux Cluster as the Core Component of the Data Processing Center

The most important problem for the DPC during the mission is the scalability. Indeed, the data storage will grow during the mission from a few GB up to 100 TB at the end of the mission in case of DIVA/AMEX (up to 1 PB in case of GAIA). The DPC must supply pipeline and other applications with data providing a fast access to the whole data volume. Solution of many other problems, e.g. safety, backup strategy and etc. depends on the scalability.

To store data we can choose between the "mainframe" strategy and the "cluster" one. In the first case we will store all data on the media (RAID arrays) connected to the same host and operated by this single host. It is clear that the practical realization of this solution will be complicated by the following problems:

- the data transfer bottleneck: the host will transfer all the stored data through the same bus which will slow data transfer,
- the cost: for 100 TB data we will need approximately 1000 harddisks. The price of the harddisk storage media is approximately 1,000 USD for 1 TB in the year 2004 (for IDE disks) with the trend for the price to decrease down to 1,000 USD for 25 TB in the year 2020 (see [Cybersource WP, 2002]).

In Appendix C we tested the dependence of the response time on the number of nodes of the cluster which were used to distribute the data. In Fig. 6.3 we see, that the dependence of the response time is not obvious: the increase of the number of nodes can increase the response time. The reason for this result and the choice of the optimum number of nodes of the cluster will be discussed in the next Chapter.

To use the cluster we have to select between the "usual" network (Ethernet, for example) and the fast switches (or fast network solution like Myrinet, for example).

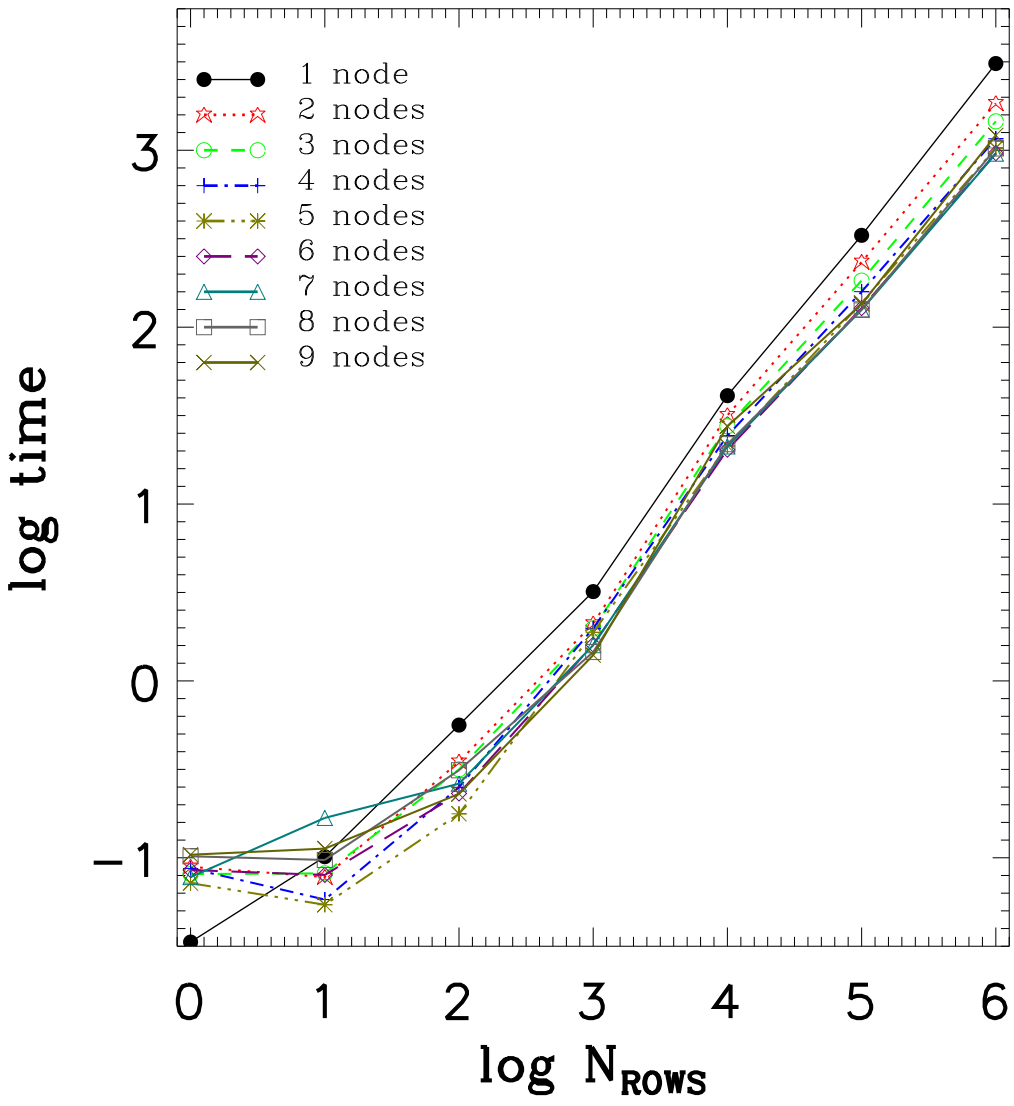

Figure 6.3: The relation between the number of nodes used in the query and the response time. The time is in sec. (See Table C.1).

Myrinet and Ethernet. In our case the main delay is due to the data retrieval from disks and not to the data exchange between nodes (we have to prevent the later case via the design of database structure). The prove of this statement is done in the Appendix E and will be described in the next Chapter. It is unlikely that the database will need Myrinet or fast switches. But as we need a computational facilities up to 32 TFLOPS (in the case of GAIA) we can organize two clusters: a big one with Ethernet and a small one with Myrinet for pipeline and other calculations.

The first solution is more suitable for the archive (offline) data storage. The "cluster" solution is more practical for the online data storage which we need. Indeed, we can

distribute data through independent hosts to reduce the amount of data stored at a single host and to provide an independent access to different chunks of data.

Possible solutions for the cluster are a shared-nothing cluster (SN), shared-memory (SM) and shared-disk (SD). Taking into account the data growth during the mission, we have to reject SN and SM clusters. In fact both have the same negative feature as the "mainframe" solution: we will need to maintain a single host with an enormous storage capacity and very bad scalability.

6.6.2 RDBMS

To store the data we have to select a stable relational DBMS. As a non-commercial project we are allowed to use most of the commercial RDBMSs for free. In the year 2004 we have following commercial DBMS for clusters:

Informix XPS Informix XPS (eXtended Parallel Server) is a very administrator- and user-friendly product. In the fall of 2001 Informix Inc was sold to IBM. Although IBM did not announce officially the close of the Informix product family, this company did not support Informix XPS for Linux any more. It is unlikely that IBM will sell Informix products till the year 2011.

Oracle RAC Oracle RAC (Real Application Cluster) does not suit for our purposes because it works on shared-memory clusters and Oracle does not produce RAC for self-made clusters.

DB2 ESE DB2 Enterprise Server Edition is the RDBMS which can be used under Linux OS and on a self-made cluster. Parameters of this RDBMS (esp. the maximum storage capacity for data pro database server) will define the number of nodes for the mission database cluster.

Finally, we can compare the cost for the Oracle RAC and DB2 ESE. At the end of the year 2004 the price for the Oracle 9i was approximately 40,000 USD per CPU, whereas the price for DB2 ESE v8.1 was approximately 25,000 USD per CPU. The MySQL server was upgraded to a distributed database server in recent years, but MySQL is still not a SQL92 compatible DBMS.

6.6.3 Compilers

In case of the pipeline and applications development from scratch it would be better to chose one programming language for all applications. Unfortunately, this is impossible, because many astronomical and astrometric programs use a number of already completed applications and subroutines written mainly in *Fortran* (some of them were developed in the case of the Hipparcos mission).

As a compromise solution we propose to use *C/C++* as the "core" language and *Fortran 77/90* as the "support" language. Since DB2 does not support Fortran for Linux OS we must design a library for application programmers.

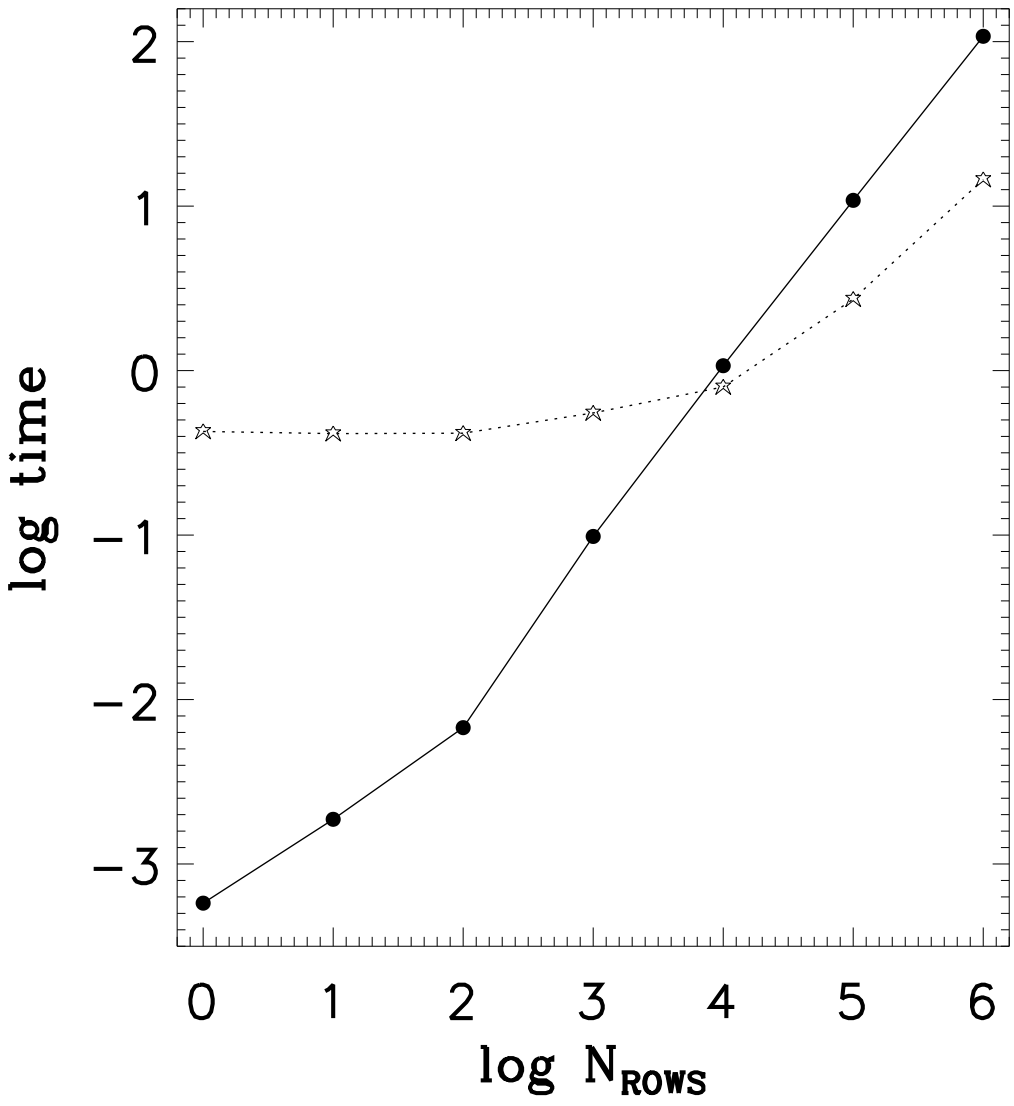

Figure 6.4: The comparison between the direct insert and the preprocessed insert. Filled circles are test results for INSERT statement, stars are test results for the preprocessed insert. The time is in sec. (See Table I.1)

6.6.4 The Data Insert

The DPC will receive a huge raw data volume each day. Even bigger processed data volume will be generated by applications running in the DPC. To reduce the time required for applications to insert the data into the database we will use an external (to applications) utility which will insert results of the work of the applications into the database. In Appendix I we tested the data insert from the application and the preprocessed data insert. In the latter case we used `db2split` utility and `LOAD FROM FILE` command of

Table 6.10: GAIA and DIVA/AMEX clusters.

Parameter	DIVA/AMEX	GAIA
Raw data, TB	4	100
Raw and processed data, TB	50	1024
Database partition servers (DB2 UDB EEE v7.2)	100	2 instances (1000 each)
Nodes, 2xCPU	50	1000
Nodes, 4xCPU	25	500

DB2. Fig. 6.4 shows that the preprocessed data insert is more effective in the case of the big data volumes.

6.6.5 Implementation of the Data Processing Center

The implementation of the data center includes following items:

The list of initial components. This includes a core cluster with a initial number of nodes, writing DVD devices and service PCs. We can fix and install a number of necessary components of the DPC and then, step by step, add new data storage devices and new nodes to the cluster. The components have to be installed and tested half a year before the mission start.

Cluster upgrade requirements. We will select a configuration for the initial cluster which we can easily upgrade during the mission. We must be sure that the hardware components we need for the cluster will be accessible at the end of the mission (e.g., for AMEX in 2011).

Upgrade plan. This plan describes an upgrade of the cluster with the data growth.

The DPC software. We will use DB2 DBMS as the core of the DPC, but we have to design and implement a number of packages and programs to maintain database, load and unload the data, backup the data and trace the performance of the DBMS and the cluster.

Table 6.10 shows the approximate configuration of the cluster in cases of DIVA/AMEX and GAIA.

Initial components

The initial configuration and the upgrade plan for the cluster depends on the time scale in which the new components (nodes) can be added and on the prediction of the data growth in the database. In the case of DIVA/AMEX we propose a half-year period as the time which we have to cover with the initial configuration. This assumes that at the beginning

Table 6.11: The upgrade plan – the growth of data during the mission as a percent of the final whole data volume and the growth of the DIVA/AMEX data cluster.

Day of the mission	Percent of data in the database	Upgrade
0	0.1	Initial cluster
150	9.8	Upgrade 1
270	19.1	Upgrade 2
390	29.2	Upgrade 3
500	39.6	Upgrade 4
580	49.2	Upgrade 5
650	58.8	Upgrade 6
720	68.3	Upgrade 7
780	78.4	Upgrade 8
840	88.2	Upgrade 9
900	98.0	Upgrade 10
910	99.7	Upgrade 11

of the mission we have a data storage for approximately 10 TB of data (10 2-CPU or 4-CPU nodes with 1 TB data space on each, 2 database servers on each node plus one reserve node). Also 32 GFLOPS cluster will be installed(in the case of DIVA/AMEX). A detached PC with DVD burning device and 5 TB RAID array will be used for archiving.

Upgrade plan

The upgrade plan is based on the data growth rate of the mission. The estimation of the growth of the data volume with time made previously (see Fig. 6.2) shows non-linear data growth. In the case of the DIVA/AMEX the data will reach 100 TB at the end of the mission and it is reasonable to add new storage capacities in 10-TB portions (this means that 10 new nodes will be added at each upgrade). The first upgrade will be needed 3 months after the start of the mission, ten upgrades will be necessary approximately every 3 months (see Table 6.11). This upgrade plan was calculated for a 3 year mission duration and can be changed in accordance with the actual duration of the mission.

Required software

OS. Linux clone, possibly the latest version of SuSE.

DBMS. The latest version of DB2 ESE.

Time server. This server is used to synchronize the local time on each node. NTP (`http://www.ntp.org`) can be used.

Compilers. Throughout this work the GNU C/C++ compiler was used (`http://gcc.gnu.org/`). Intel C/C++ and Fortran compiles were tested as well and can be used for the soft-

ware development (`http://www.intel.com/software/products/compilers/clin/` and `http://www.intel.com/software/products/compilers/flin/`).

6.6.6 Classification of Failures

We can classify possible failures by the following way :

Software failure: OS. In the case of an OS failure all work will be restarted from the last commitment point (the time than the last successful results of pipeline work were fixed). The damaged node will be fixed as in a case of node failure (see below).

Software failure: DB2 data load. During each cycle of data load we have to put a big chunk of data into the database. We do not suggest to log a raw data table. If during the raw data load a failure will appear, we will stop the load procedure, delete the newly inserted data portion and repeat the procedure of data load.

Software failure: DB2 server failure. In the case of a server crash, the server will be stopped and the reason of the failure will be analyzed. Usually, this kind of failure indicates problems of hardware or OS.

Hardware failure: Harddisk failure. In the case of the failure on the harddisk the affected disk will be replaced and all data located on the disk will be restored from the backup.

Hardware failure: node failure. In order to substitute a crashed node, we propose to keep an empty node in hot reserve. In the case of a failure all applications and the DBMS server will be stopped, the data will be placed into the reserve node, the corresponding changes will be made in the configuration of the DBMS server, the DBMS server and all applications will be restarted from the last commitment point.

Hardware failure: periphery device. The most critical situation occurs if the external network link with the SOC would crash or the backup device(s) would fail. In the case of a failure produced by the DPC periphery devices we have to keep a reserve of hardware to replace the damaged devices (for example, writing DVD device).

6.6.7 The Backup and Recovery System

There are two groups of processes in the DPC. The first one will retrieve the data for a day of observations and will produce the new processed data daily (Pixel Data Processing, Attitude Reconstruction, Great Circle Reduction). The second group of processes will need the data for at least half a year of observations and will produce the new processed data once per half a year (Sphere Reconstruction, Astrometric Parameters Determination, Object Recognition and Preliminary Identification). Both data chunks produced by these two groups of processes will never be changed after the insert and can easily be archived before an insert to the database. The data insert for each day of the mission will be stored

in the archive and can be easily repartitioned and restored. As a result the optimal choice would be a circular logging with the backup of all input data.

The restore of the data in the case of software failure will be done by the RESTART DATABASE command. The only changes in the database are due to the insert operation and in this case the operation will be terminated and restarted (we have no UPDATE operations for the database).

The case of a hardware failure is more complicated and dangerous. As we deal with the huge amount of data we are unable to remove and add nodes by standard DB2 commands. Indeed, these operations will require a logging during the data redistribution. The maximum storage capacity for the logging at the separate node is (Maximum Number of Log files:128) $\times$ (Maximum size of the Log file: 65535 pages)$\times$ (The page size: 4 kB) = 32 GB. The redistribution of the data among nodes will require a huge time as the size of the data chunk at each node will be close to the maximum capacity of 512 GB.

As a solution we will keep an empty reserve node in the cluster excluded from the database server nodes. In the case of a hardware failure the damaged node will be excluded from the configuration, the archived data volume of the damaged node will be restored at the new node, the name of the damaged node will be assigned to the new node and the database will be restarted.

The IBM Tivoli Storage manager can be used for the archiving from the database cluster.

6.7 Summary

We have created a concept for the DPC which based on a few simple principles: low cost for the realization, low manpower to support and the decrease of the response time for time-critical applications.

As we will see from the next chapter, it is possible to improve the response time significantly for the case of the distributed database and shared-nothing cluster if we abandon the uniform data distribution through nodes of the cluster.

Chapter 7

The Partitioning Problem: Theoretical Background

In the case of distributed databases the problem of the optimum response time is in fact a problem of the data distribution over the nodes of the cluster which is used to store the data. The method for the data distribution can be round-robin, range- or hash-based, but the existing commercial DBMSs use these methods of the data distribution to create a uniform data distribution over the nodes. If we are using the non-homogeneous hardware (as we use for the DPC to reduce the cost for the hardware) the uniform data distribution becomes non-optimal.

In practice we have to create a system which will work with the software we have pre-selected for the Data Processing Center, so we are limited by the relational DBMS we have chosen and the logical database structure designed in the previous chapter. Also, we have to place the data on the shared-nothing cluster.

This chapter describes the theoretical basis which can be used to improve the response time. The task of the improvement of the physical structure of the database can be subdivided into the task of the design of the initial data placement and the task of the data reorganization during the work with the data. The latter task includes the monitoring of the system and the generation of the time stamps for the data reorganization. Also, we have to find an optimum configuration of the cluster itself (the optimum number of nodes of the cluster in our case).

To solve the partitioning problem we have to create a system, which will describe the time, which the DBMS needs to retrieve data as a function of:

- operations $\{R_i\}$,
- nodes of the cluster $\{N_i\}$,
- tables $\{T_i\}$ and indices $\{K_i\}$,
- data storage devices $\{d_i^j\}$.

We will use the volume of the input data M (number of rows) to produce a distribution of the data for the cluster $\{M_i\}$ (number of rows on the node i of the cluster).

The data distribution will be based on the estimation of the response time which the user of the database needs to retrieve the data. The estimation of the response time will be made with the use of the Time Cost Function (TCF).

We have to solve following problems in the framework of the partitioning problem:

- the choice of the optimum number of nodes of the cluster which will share the data,
- the choice of the optimum control node, which will be used by the application to retrieve the data from the cluster,
- the optimum data distribution for the pre-selected optimum control node and the defined number of nodes of the cluster,
- the monitoring of the system.

7.1 The Problem of Physical Data Placement

To solve the problem of the physical data placement in general form we have to develop a physical design of the database with the use of the information about the system which maintains the database.

The solution must provide a scenario for the data placement, grouping and clustering. The solution will include partitioning key definitions and indices for the database scheme. The description of the problem, algorithms for the solution and tests for the distributed databases on shared-nothing systems can be found in [Zilio, 1998]. The case of the more general task is reviewed in [Kulba et al., 1999].

Our case differs from the general case reviewed in [Zilio, 1998]. We have already specified the database scheme on the basis of the "naive" algorithm and the rough estimation of the theoretical time cost function (Chapter 6). Also, we are interested in the non-homogeneous case of the cluster used to store the database.

The algorithms described in the [Zilio & Jhingran, 1994] and [Zilio, 1998] are applicable to the initial design of the database and the reorganization of the database. They are based on the estimation of the workload and are dedicated to the generation of the optimum logical structure of the database, whereas we need an optimum data distribution (which is a part of the physical design of the database) for the already designed logical scheme of the database.

To describe the process of data retrieval from the database we have to describe the way which the DBMS uses to select the data and to transfer the selected volume to the end-user application. The query optimizer creates the execution plan for the query, the query is subdivided into subqueries for each node. After the execution of subqueries (which requires sometimes an intense data exchange between nodes) the result is returned to the control node where the application is running which asked for the data.

We describe in the following the principle procedure, how we will distribute data through nodes and the measure which will be used to generate such a distribution. For the last purpose the Time Cost Function will be used. This TCF must be different from the

theoretical approach used in the previous Chapter and must be based on the use of the real DBMS and the database structure we have developed.

Both the partitioning problem and the problem of the selection of the partitioning key are NP-complete (see, for example, [Garey & Johnson, 2003] and [Zilio, 1998], Appendix F). Nevertheless, as we will see later, it is possible to simplify the problem and to create a quite simple algorithm for the solution by means of some restrictions on the general task.

7.1.1 Data Partitioning Through Nodes. Tablespaces for Data and Indices

The time required for the selection of data from a database depends on the organization of the request and on the physical data placement of the requested data. DB2 can maintain up to 512 GB disk space for one node (in the case of a 32K page size). From the estimation of the required data space (tablespace, indexspace and temporal data space) we can estimate the number of nodes needed for data storage. It is essential to place data in an optimum way to minimize the response time. Unfortunately there is no way to use devices for data storage in current DB2 version for linux (direct use of disk partition). Hence we will limit ourselves to files and will use 2GB files as the maximum file size for data storage. Among System Managed Space (SMS) and Database Managed Space (DMS) we will use the DMS type of tablespace storage due to a better effectiveness of DMS.

Let us suppose our cluster consists of N nodes $\{ N_i\}$. Each node i contains B tablespace files $\{b^i_j\}$ and D index space files $\{d^i_j\}$. We have a limit on the volume of tablespace and indexspace files

$$\sum_j b^i_j + \sum_j d^i_j = S_i$$

$$\sum_{i,j} (b^i_j + d^i_j) = \sum_i S_i = V_{total}$$

The value of the maximum space allowed for the table differs for different DBMSs (for example, for DB2 UDB EEE $S_i = 512GB$).

Data partitioning through nodes for DB2 is performed by the use of a partitioning key – an attribute in the table used to find a node number where the row will be placed (the hash method is used). For our database the natural way to place data is the use of the second parts of the Window Identifier, the Image Identifier and the Object Identifier (NW2, NI2, NB2). The first part of the key stores information about the running number of the scan (NW1) which will be used to retrieve information from the database (see Chapter 6).

The principle idea of the data distribution through nodes is based on the use of the generated identifier to assign a fixed number of rows to each node. The incoming raw data will have no identifier for the Window. The combination of Time Counts and CCD chip information and Window Type can be used as an unique identifier (see Chapter 3).

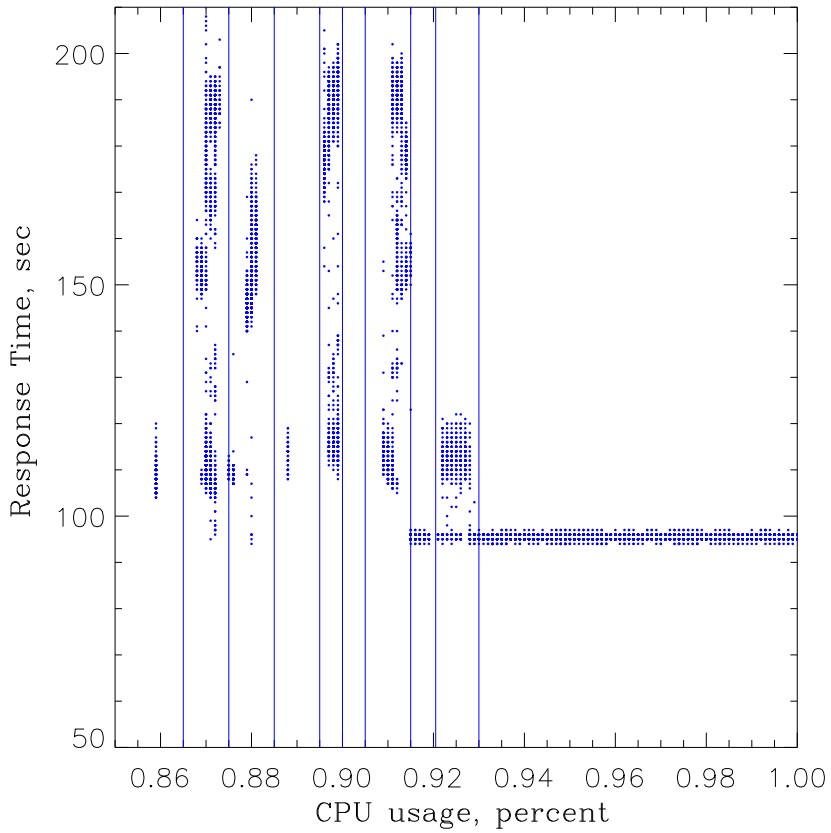

Figure 7.1: The response time vs the CPU usage. To find the dependence between the response time and the CPU usage was subdivided into 10 group (thin vertical lines). The CPU usage interval corresponds to the case of the database cluster shared with other (non-database) applications.

7.1.2 Requests to the Data. Time Cost Function

The Time Cost Function (TCF) is the time which is required to retrieve one row from the database. There are two ways to describe the TCF:

- a parametrical description, which is based on the use of the dependence of the TCF from a number of parameters describing the hardware and software used to store and retrieve data and
- a non-parametrical description which supposes an empirical description of the TCF (the TCF depends on the local time only).

Despite attempts to describe the TCF parametrically (see, for example, [Tomov et al., 2004]) the practical use of the parametrical description of the TCF is doubtful. A simple test can be performed to clarify the problem: on the single node the response time for the same number of rows retrieved was measured simultaneously with the percentage use of the CPU. A single node of the linux cluster installed in Astronomisches-Rechen Institut

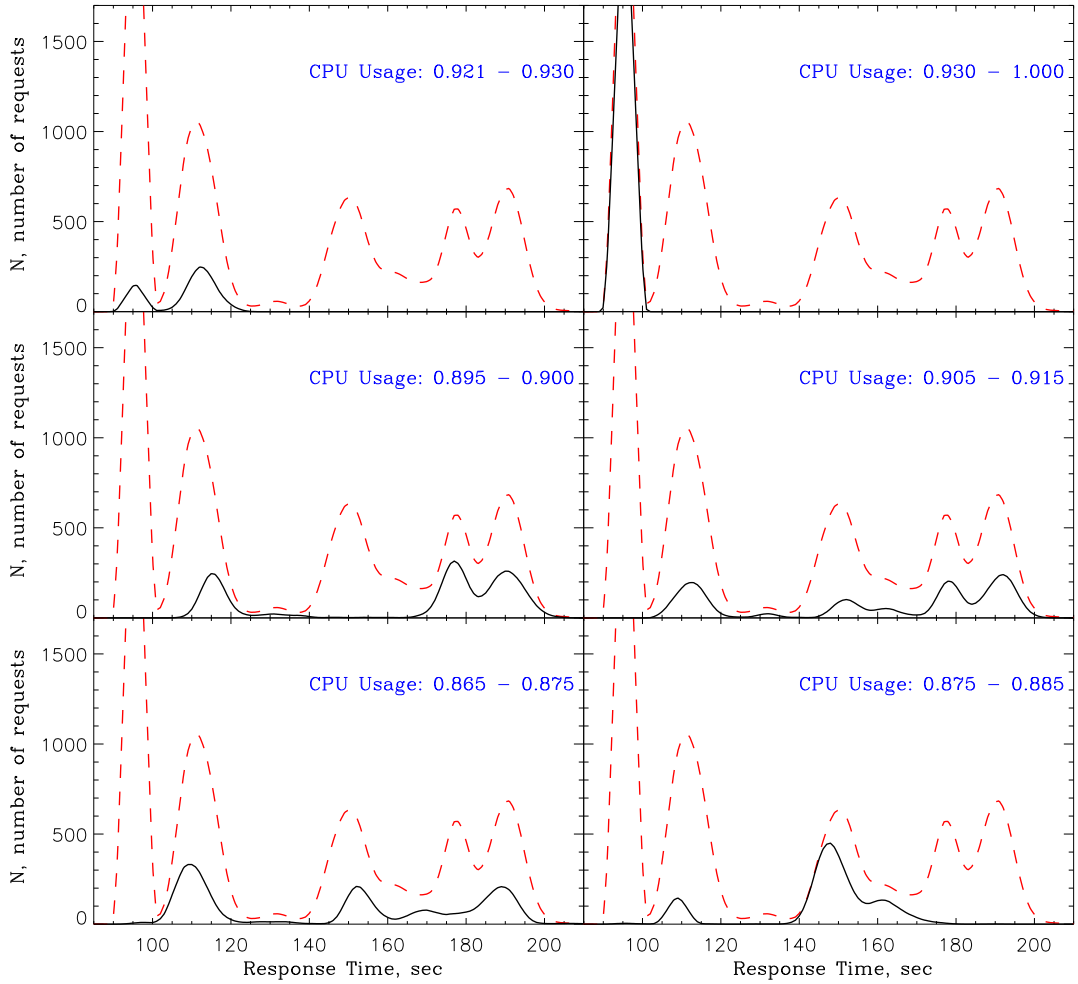

Figure 7.2: The distribution of the response time for different CPU usage. Dashed line is the distribution function for all response time, thick solid line is the distribution function for the response time for the selected range of the CPU usage.

(ARI) was used for the test (dual Pentium 4 2,2 GHz with two 73GB SCSI disks). Operational system was SuSE linux 7.2, DBMS – DB2 UDB EEE v7.2. The database was created at the single node of the cluster with the single table which had the only integer attribute. The number of rows in the table was 10^9 and all rows were retrieved by the request. The response time shown in Fig. 7.1 was measured for the different usage of the CPU, but there is no direct dependence in the response time on the CPU usage at all. In Fig. 7.2 the dashed line shows the distribution of response times for all ranges of the CPU usage and the thick solid line – in each interval of the CPU usage. As we can see,

a minimum response time can occur for the high values of the CPU usage and vice versa. This does not imply, that there is no dependence between the response time and the CPU usage, but the result of the test shows, that the dependence of the TCF on parameters is not additive. We have to reject the parametrical description of the TCF and switch to the non-parametrical one, which will depend on time only and will be found from direct measurements of the response time.

As we can find out from the previous chapter the most important request to the database is a simple select statement. We will suppose that the Time Cost Function $C = \{C_j\} = \{J_j + Q_j\}$, where Q_j - the time required for the select-statement executed on node j and J_j - the time required for the join-statement executed on node j. It is clear, that Q_j will depend on the data placement on the node j only, meanwhile J_j will depend on the data placement on all nodes and the data link rate between the nodes. Finally,

$$Q_i = f^{select}(M_i),$$

$$J_i = \sum_{\substack{j=1 \\ i \neq j}}^{N} f^{join}(M_i, M_j),$$

where f^{select}, f^{join} are functions, describing the time required to select and join sets of rows M_i, M_j. We have to take into account the time required to transfer M_j rows from node j to node i

$$T_{ij} = f_i^{transfer}(M_j).$$

As these transfers and the local query at node i will be done in parallel, the final time required to perform a query at node i will be

$$\begin{aligned} C_i &= \max\{Q_i, T_{ij} | j \in \{n_i\}\} + J_i \\ &= \max\{f^{select}(M_i), f_i^{transfer}(M_j) | j \in \{n_i\}\} + \sum_{\substack{j=1 \\ i \neq j}}^{N} f^{join}(M_i, M_j). \end{aligned}$$

We have to find an appropriate form for the TCF (both f^{select} and $f^{transfer}$).

7.1.3 The form of the TCF

To find a form of the TCF we tested a time required to retrieve a various number of rows from the same node locally (the application runs at the same node) and remotely (the application runs on the another node of the cluster). The test was made at Astronomisches Rechen-Institut with the use of the ARI's cluster (see Appendix H for the full description of the test). The result is shown in Fig. 7.3. Following this Figure we adopted the empirical form for the TCF:

$$\log f = A_0^f + A_1^f \, e^{-A_2^f \log N_{ROWS}},$$

where f is the TCF (response time in seconds per row), M is a number of rows which is distributed over N nodes of the cluster, $N_{ROWS} \equiv M/N$ (we supposed an uniform distribution of the data).

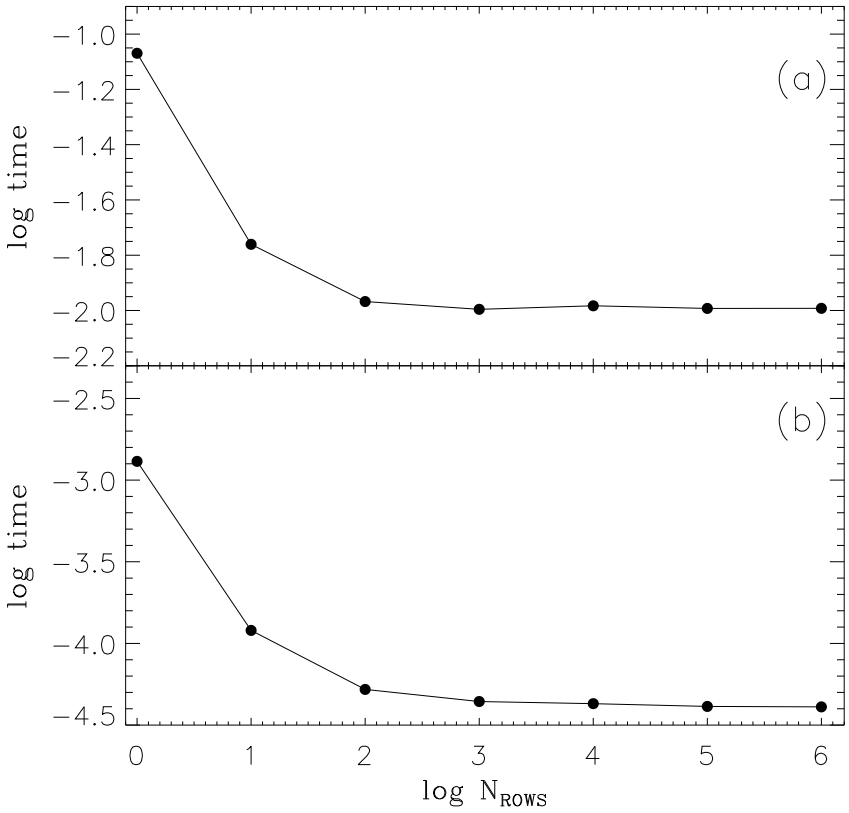

Figure 7.3: The TCF for local and remote data retrieval. The panel (a) shows local data retrieval, the panel (b) – remote data retrieval. Time is in seconds. (See Table H.1).

The fitting of the proposed form of the TCF shows a good agreement for local (Fig. 7.4) and remote (Fig. 7.5) data retrieval.

$$\log f_{local} = (-4.43 \pm 0.29) + (1.52 \pm 0.69) \exp((-1.00 \pm 0.00) \log M/N),$$

$$\log f_{remote} = (-2.04 \pm 0.29) + (0.94 \pm 0.68) \exp((-1.00 \pm 0.00) \log M/N).$$

It is clear, that

$$\lim_{\frac{M}{N} \to \infty} \log f = A_0^f,$$

in practice $\log f = A_0^f$, if the number of retrieved rows $N_{rows} > N_{\text{limit}}$ ($N_{limit} = 10^3$ in our case). As a result, the TCF depends linearly on the number of retrieved rows for a massive data retrieval.

7.1.4 The Stability of the TCF

The increase of the time required to transfer a number of rows between nodes is linear. Nevertheless, the real transfer rate can be influenced by a number of factors (see, for

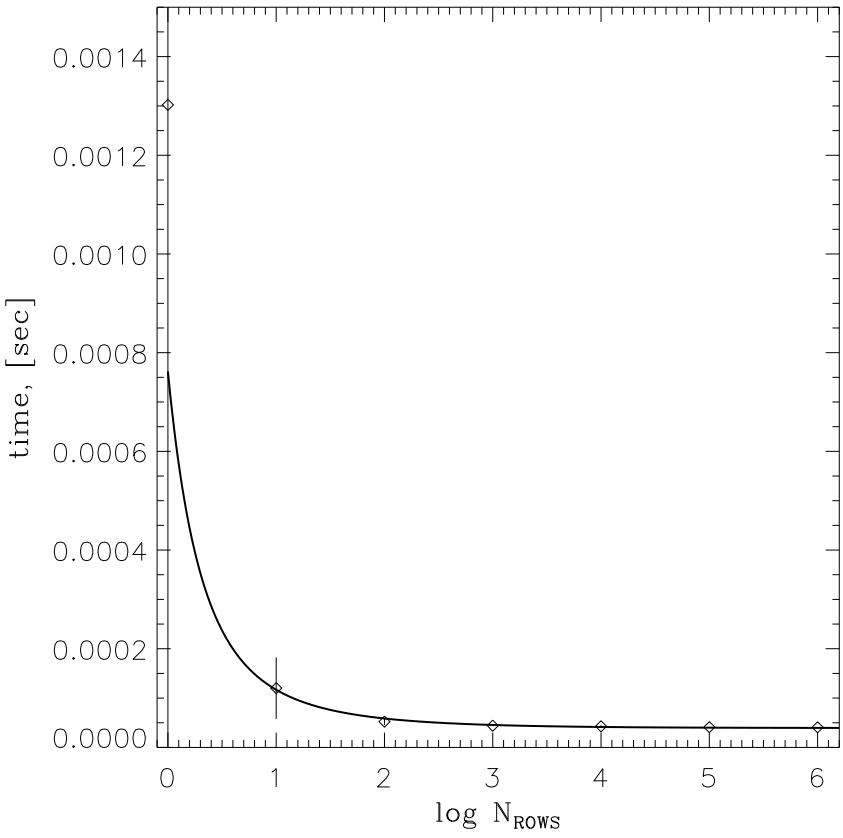

Figure 7.4: The approximation of the TCF for a local data retrieval. The line is a fitted function, squares are measured values for the TCF (see Table H.1).

example, [Arsenev & Yakovlev, 2001]). As result, the transfer rate will be not constant and will depend on a number of parameters (like CPU usage, for example. Also, the TCF depends linearly on the number of rows retrieved if the number of rows exceeds some critical number N_{limit}.

To test the stability of the TCF we retrieved in10^7 rows from the database in the ascending order (see the description of the test in Appendix D). As the node which was used to perform the test was loaded with others, non-database applications and the load was not stable we see a number of quasi-stable response times in Fig. 7.6 instead of the only response time. Each quasi-stable response time is dispersed around some value. The estimation of this dispersion will be important to characterize the performance of the cluster at the selected moment of time t. High dispersion is an indicator of fast changes in the number of running programs, the use of CPU and memory. Based on the results of the test the distribution of the response time for the same request can be described as a sum of Gauss function:

$$N(t) = \sum_{i=1}^{n} e^{-(t-t_0)^2/(2\,\sigma^2)},$$

where $N(t)$ is a distribution function for the response time, t is the response time and

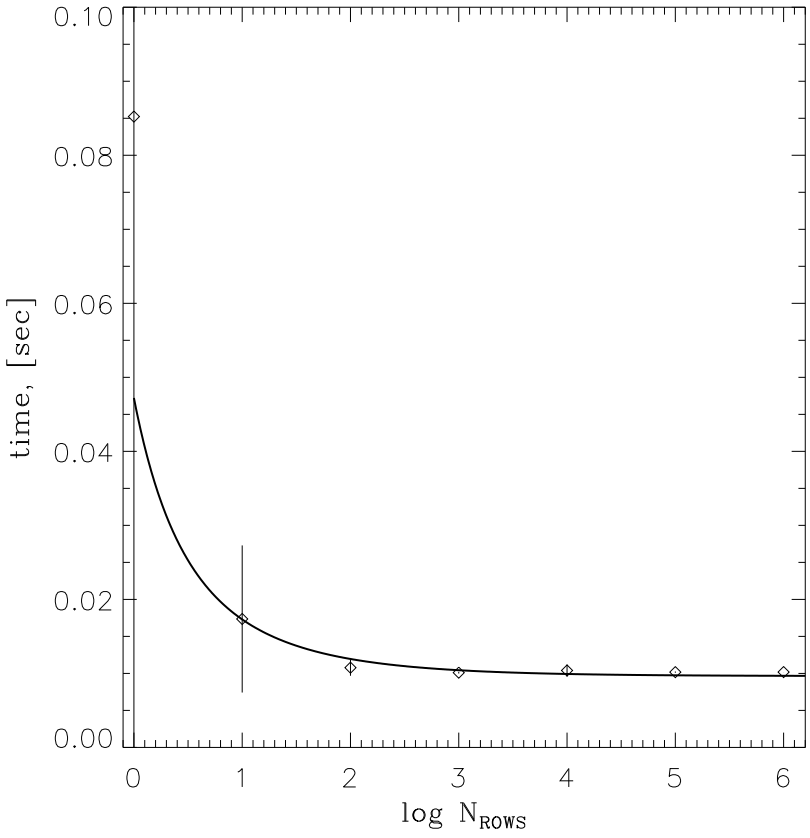

Figure 7.5: The approximation of the TCF for a remote data retrieval. The line is a fitted function, squares are measured values for the TCF (see Table H.1).

σ is the dispersion of the response time around some quasi-stable response time t_0 (see Fig. 7.7 and Appendix D for an example of the use of the distribution function for the determination of the probability of the quasi-stable solution).

In Appendix E we described the test which was made to measure the response time for the requests with the increase of the number of rows retrieved. We increased the number of retrieved rows from 100 rows in the transaction up to 7 754 700 rows. The result is shown in Fig. 7.8. As we can see, the linear dependence is confirmed in the experiment but shows as well a very big dispersion of the time required to retrieve the data around some average linear dependence. In practice we can not rely on the TCF once measured at some node of the cluster but we have to trace the performance of the cluster and the change of the TCF. Indeed, the difference between two linear dependences constructed on the same node with the same request but for different time intervals (which means a different node overload) reached 300 %. Appendix E shows that despite the dependence of the response time on the number of rows retrieved is generally linear, the coefficients of the linear relation are unstable in the case of a real cluster loaded with a number of applications. Theoretically the remote node should always show the longer response time compared with the response time for the same number of rows retrieved than the local node. In practice the remote note can be much faster than the local node due to a number

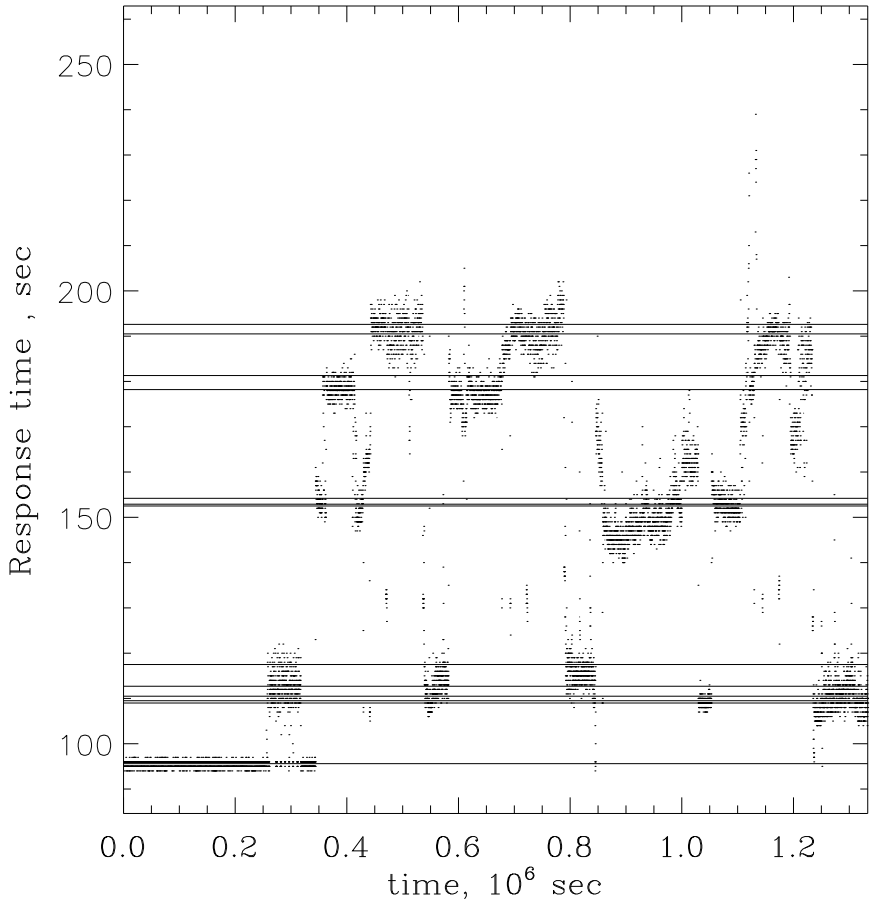

Figure 7.6: The response time and the quasi-stable solutions (thin lines).

of factors (the main factor in the test was the overload of the local node with an external to the DBMS work). As we can see from the experiment as well, the null-point for the linear relation can be assigned to zero.

7.1.5 The Time Cost Function in the Form of Linear Matrix Equation

We made the following assumptions:

- there is no join-statement in the query. Indeed, we are interested mostly in the select-statement, which requires to select rows from each node and transfers these rows to the control node. As result, $f^{join} \equiv 0$;

- the form of select function f^{select} is a simple linear dependence on the number of selected rows M_i;

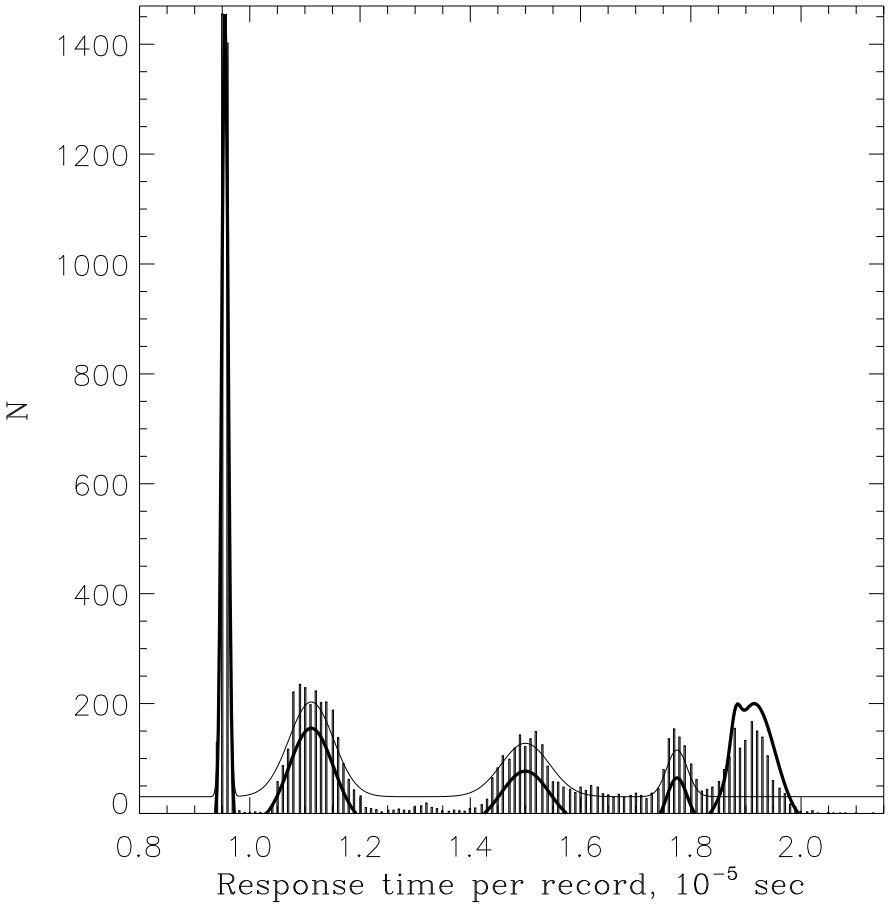

Figure 7.7: The distribution function of the response time. The approximations of the distribution function are shown in case of solutions [1,6] from Table D.2 (thick line) and solutions [1,4] (thin line).

- the form of the transfer function $f^{transfer}$ is linearly dependent on the number of transfered rows as well;
- the response time does not depend directly on the hardware and the communication architecture. We do not use some analytical assumptions about this dependence but we have to estimate this dependence during tests.

The linear form of the select and transfer functions is

$$f^{select}(M_i) = f_i \, M_i$$

and

$$f_j^{transfer}(M_i) = t_{ij} \, M_i.$$

To find C_i for each node i we have to fill the following matrix and to take the maximum value from each row. We assume that the operator $\widehat{\Omega}$ returns the maximum value from each row of the matrix and converts $[m \times n]$ matrix to $[m \times 1]$:

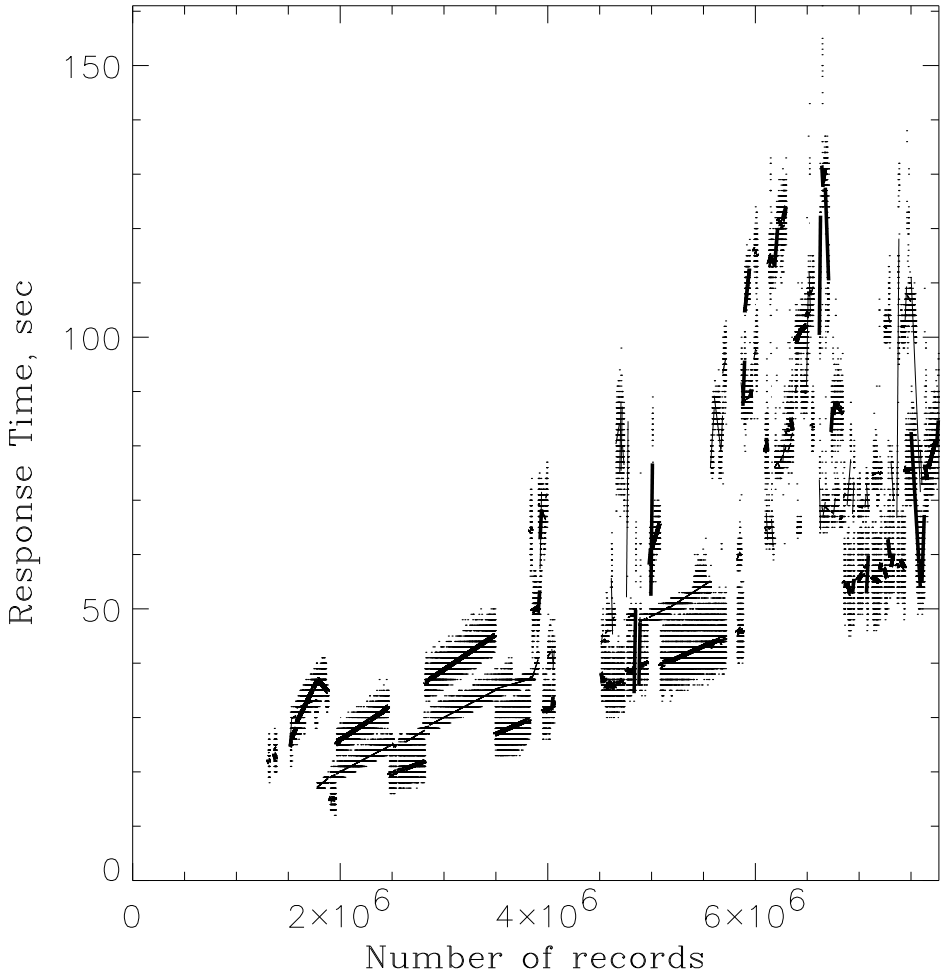

Figure 7.8: The form of the TCF and the dependence of the TCF on the number of retrieved rows. The approximation of the TCF by linear function is shown - thin line for the local node ($TCF(N_{rec}) = A_0^0 + A_1^0\, N_{rec}$), thick line for the remote node ($TCF(N_{rec}) = A_0^1 + A_1^1\, N_{rec}$), see Table E.1.

$$\mathbf{C} = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_N \end{pmatrix} = \widehat{\Omega} \begin{pmatrix} f_1\, M_1 & (f_2 + t_{2N})\, M_2 & \cdots & (f_N + t_{1N})\, M_N \\ (f_1 + t_{21})\, M_1 & f_2\, M_2 & \cdots & (f_N + t_{2N})\, M_N \\ \vdots & \vdots & \vdots & \vdots \\ (f_N + t_{N1})\, M_1 & (f_2 + t_{N2})\, M_2 & \cdots & f_N\, M_N \end{pmatrix},$$

$$\mathbf{C} = \widehat{\Omega}(\mathbf{F} + \mathbf{T})\, \mathbf{M},$$

where $\mathbf{F}$ matrix is a diagonal matrix with $f_{ij} \equiv f_i$, if $i = j$, and $f_{ij} \equiv 0$, if $i \neq j$. The matrix $\mathbf{F} + \mathbf{T}$ is a tool to describe the state of the cluster and to predict the time required to retrieve $\mathbf{M}$ rows from any control node.

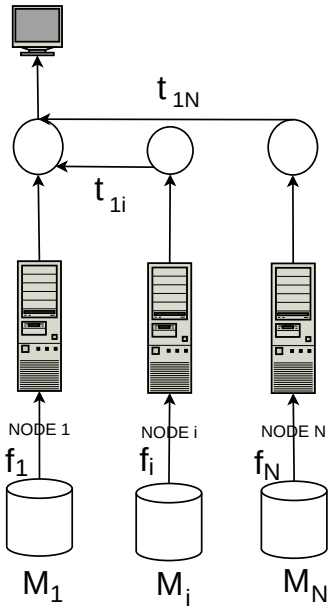

Figure 7.9: The matrix approach to the description of the cluster.

In the case of the local data retrieval we have f_i, in the case of the data placed on the node j and the request was made at the node j we have t_{ij}. From the previous Section and Appendix D we assumed a Gaussian form for the dispersion law for both coefficients f_i and t_{ij}. The natural way is to estimate not only the matrix $\mathbf{F}+\mathbf{T}$ but the dispersion matrices $\delta\mathbf{F}$ and $\delta\mathbf{T}$. If the matrix $\mathbf{F}+\mathbf{T}$ describes the behavior of the cluster for the selected DBMS, the dispersion matrices describe the instability of the cluster performance due to the overload of the cluster by other tasks.

7.2 The Minimax Problem in the Case of the System of Linear Equations

To find an optimum data distribution $\mathbf{M}_{opt}$ we have to minimize the $\mathbf{C}$ ($\mathbf{C} = \mathbf{\Omega}(\mathbf{F}+\mathbf{T})\,\mathbf{M}$). The "minimization" does not mean, that all members of the $\mathbf{C}$ must be minimal, but that the $\mathbf{C}$ must include C_i so that there is no C_j: $C_j < C_i$. We are looking for the minimum of $\mathbf{C}$, whereas each member of the matrix $\mathbf{C}$ is a maximum of the corresponding row of the matrix $\mathbf{(F+T)\,M}$: $C_i = \max_j(f_i + t_{ij})\,M_i$. This is the minimax problem in the case of the system of equations (see [Sukharev, 1989]).

Two features of the TCF are able to reduce the complexity of the problem: these are the constant number of rows we have to distribute and the linear form of equations.

Let us suppose, that we have to distribute M rows of equal size among N **equivalent** nodes. We simplify the problem taking $(f_i + t_{ij}) \equiv 1$ for any i and j. This mean that there is no difference between the local and remote data retrieval.

Theorem 1. The distribution of M rows among N equivalent nodes without specification of the control node (*the local and remote data retrieval have the same response time*) is

given by the solution of the minimax problem $M_{min} = \min_i \max_j M_j$ in the case of the uniform distribution $M_j = M/N$.

Proof. Let us suppose that the Theorem 1 is wrong, i.e., $\exists M_i : M_i < M/N$. But, as the number of rows to distribute is fixed: $M_i + \sum_{j=1, i \neq j}^{N} M_j = M$, $\exists M_k : M_k > M/N$, that contradicts with the minimax requirement $M_i = \max_j M_j$. $\square$

For a transition to the case of non-homogeneous nodes we use a simple geometrical analogy. Let us suppose that we have to set a point in N-dimensional space so, that the maximum component of its coordinate of the point will reach a minimum, all coordinates are positive and the sum of all coordinates must be equal to M. It is proved in Theorem 1, that the problem is solved if each coordinate is M/N. The case of the non-homogeneous nodes in our approach of linear dependence of the response time on the number of rows corresponds to the linear transformation of one coordinate system to another one. The coordinate M_i will be transfered to the coordinate $\tilde{M}_i = (f_i + t_{ij})\, M_i$ (j is fixed).

Theorem 2. The distribution of M rows among N nodes with the control node j and matrix $\mathbf{F} + \mathbf{T}$ satisfies the minimax solution in the case $M_i = \frac{1}{\sum_{k=1}^{N} (f_{ij} + t_i)/(f_{kj} + t_k)}\, M$.

Proof. In the case of system of linear equations we have the transformation to the new coordinate system with coefficients of the $\mathbf{F} + \mathbf{T}$ matrix as coefficients of transformation. Indeed, the new values of the $\tilde{M}_i$ for each node are $(f_{ij} + t_i)\, M_i$ where j is a preselected control node. The sum of the number of rows in new coordinate system will be $\sum_{i=1}^{N}((f_{ij} + t_i)\, M_i)$ and the new optimum data placement in case of the minimax solution is $\sum_{i=1}^{N}((f_{ij} + t_i)\, M_i)/N$ for each node. The uniform distribution of the rows in the new coordinate system means $(f_{kj} + t_k)\, M_k = (f_{lj} + t_l)\, M_l)$ for any k, l. Combined with the minimax solution and Theorem 1, this will lead us to $M_k = \frac{1}{\sum_i (f_{kj} + t_k)/(f_{ij} + t_i)}\, M$. $\square$

7.3 Estimation of the gain

Accepting the method of data placement proposed above we can easily estimate the theoretical gain of the new method. In case of a uniform data placement of M rows the time required to retrieve the data is governed by the slowest node.

$$t_{\text{uniform}} = \frac{M}{N} \max_i (f_{ij} + t_j),$$

where j is the preselected control node. In the case of the optimum data distribution the retrieval time for the control node is

$$t_{opt} = M_j\, t_j = \frac{M}{t_j \sum_i 1/(f_{ij} + t_i)}\, t_j = \frac{M}{\sum_i 1/(f_{ij} + t_i)},$$

and the gain is

$$\Delta t = M\, (\frac{\max_i (f_{ij} + t_j)}{N} - \frac{1}{\sum_i 1/(f_{ij} + t_i)})$$

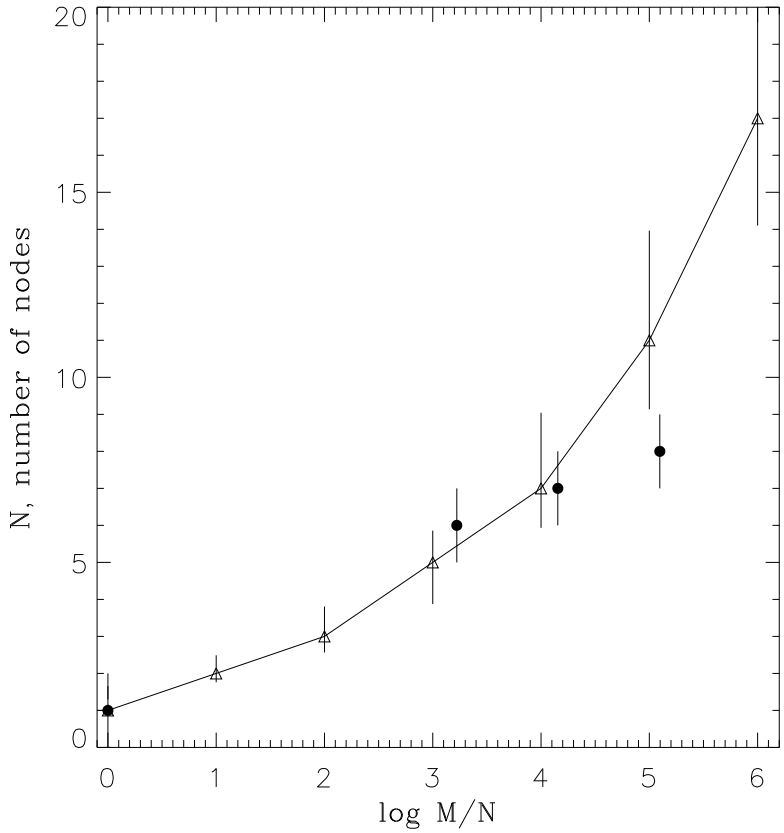

Figure 7.10: The optimum number of nodes. The line shows the optimum number of nodes for M/N rows per node. Triangles are the predicted values with errors. Errors are estimated from the errors of f_{local} and f_{remote} functions (see Section 7.1.3). Points are experimental results (see Test C, Table C.1).

7.4 The Use of the TCF

There are a number of tasks which can be solved with the use of TCF and the matrix $\mathbf{F} + \mathbf{T}$, they will be reviewed in the following.

7.4.1 Optimum Number of Nodes

To find an optimum number of nodes to be used in the cluster we can use simple and reasonable requirements: we install a new node if this improves the response time of the request. Let us suppose that we use an uniform data distribution through the cluster. In the case of **N** nodes and **M** rows which will be retrieved by the request the time required to retrieve rows stored at the local node is $f_{\text{local}}(M, N) = f(M/N)$, and the time required to retrieve rows stored at the remote node is $f_{\text{remote}}(M, N) = f(M, N) + t(M, N) = f(M/N) + t(M/N)$, where $t(M, N)$ is a time required to transfer M rows from the remote node in the case of N nodes. We will suppose that the cluster is completely homogeneous.

The number of rows to distribute is fixed (**M**). If the new node is added, the number of rows stored at each node decreases and the time required to retrieve rows from the local node and from the remote node decreases as well. If we retrieve all rows **M** (distributed over the cluster) the response time is defined by the slowest operation – the remote data retrieval, and the response time is $f(M/N)+t(M/N)$. The requirement for the optimum number of nodes is that the data retrieval after the add of the new node is faster than the data retrieval from the local node before the add of the new node. This mean, that the new node must be added if

$$f(M/N) + t(M/N) < f(M/(N-1)),$$

this is the requirement of the optimum number of nodes.

$f(M,N)$ and $t(M,N)$ are the coefficients of the matrix **F**+**T**, but in the case we are studying they are no more constant and depend on the number of rows retrieved (we adopt the relation which was described in Section 7.1.3).

The requirement of the optimum number of nodes becomes

$$f(M/N) + t(M/N) < f(M/(N-1)),$$

and we neglect $f(M/N)$ ($f(M/N) \ll t(M/N)$, see Section 7.1.3). Rewriting the requirement of the optimum number of nodes with the use of the proposed form of the coefficients and new variables $y = N$ and $x = M/N$ we will have this requirement in a new form:

$$y^{-B_2^f} > \frac{B_0^t - B_0^f}{B_1^f}\, x^{B_2^f} + \frac{B_1^t}{B_1^f}\, x^{B_2^f - B_2^t},$$

where $B_k^f = A_k^f / \ln 10$, $B_k^t = A_k^t / \ln 10$. The result for the test cluster with the use of values for $f(M/N)$ and $t(M/N)$ from Section 7.1.3 is shown in Fig. 7.10. The optimum number of nodes is the line in Figure, if we will use more nodes (the space over the line) the response time will increase, as we see from the test for the dependence of the response time on the number of nodes (see Appendix C, Table C.1).

7.4.2 Optimum Control Node

Among all nodes $\{\mathbf{N_i}\}$ we have to find a node $N^{\rm opt}$ which the application will use to retrieve the data. The optimum control node is the node which corresponds to the minimum response time:

$$i^{\rm opt} : C_{i^{\rm opt}} = \min_i C_i.$$

7.4.3 The Status of the Cluster

It is important for the data retrieval to keep a stable response time on the cluster. To analyse the time required to select the data and the dispersion of this time we use an average TCF at some moment of time and the dispersion of the TCF for the time interval around this moment of time:

$$< (F+T) >_{\{R_i\},[t_0,t_1]},\ \delta(F+T)_{\{R_i\},[t_0,t_1]}.$$

The average value of the TCF and the dispersion are calculated for the time interval $[t_0, t_1]$ and a group of operations $\{R_i\}$ in the assumption that the response time distributed around the average value by the Gauss law. The dispersion and the average values of the TCF can serve as an indicator of the cluster overload by other application, and is used for the planning of the daily work.

7.4.4 Optimum Data Distribution

Let us assume an input of the data at the moment t_0. We have to design an optimum data distribution so, that at the moment t_s of the data selection by the operation R_i the TCF will reach its minimum. The task is to find $\{M_{\text{opt}}\}$:

$$< (F + T) > M_{\text{opt}} = C_{\text{opt}},$$

$$C_{\text{opt}} : \min_i < (F + T) > \mathbf{M}_i,$$

where i runs through all possible distribution of rows $\mathbf{M}$ over nodes of the cluster. The solution of the problem was found in Section 7.2.

7.4.5 Data Redistribution

In the case of a data redistribution we have to know two TCFs: for the time interval t_1 before redistribution ($(F+T)_{t_1}$) and another one for the desired time interval t_2 ($(F+T)_{t_2}$).

With the use of the TCF we are able to answer two principle questions: the first one is the prediction of the time required to retrieve a number of rows from the database and, the second one is a scenario for the optimum placement of the data with the final goal to minimize the response time $\mathbf{C}$. These two tasks are named *direct* and *reverse* tasks in the following. In the first case we already have a number of rows placed at the cluster ($\mathbf{M}$ rows) and we compute the matrix $\mathbf{F} + \mathbf{T}$ to find the select and transfer coefficients (f_i and t_{ij}). We use the matrix $\mathbf{F} + \mathbf{T}$ which we have found in the first task (the matrix depends on the characteristics of the cluster and the DBMS but not on the data placement) to find an optimum $\mathbf{M}$.

The same method can be extended to the case of data expansion. With the $\mathbf{F}+\mathbf{T}$ matrix we are able predict an optimum scenario in the case than we will need to change the number of nodes N.

The matrix $\mathbf{F}+\mathbf{T}$ is not stable. The coefficients of the matrix depend on the load of the cluster. The number of applications running on the cluster will change during the mission. As a result we have to compute the matrix $\mathbf{F} + \mathbf{T}$ periodically to monitor cluster performance. This also means, that we have to keep on the cluster the test database (specially designed and distributed over the nodes of the cluster) to compute $\mathbf{F} + \mathbf{T}$. This test database has a pre-defined distribution of rows $\mathbf{M}$ over nodes and uses the disk space on the cluster. In the next Section we are trying to minimize the disk space which the test database occupies.

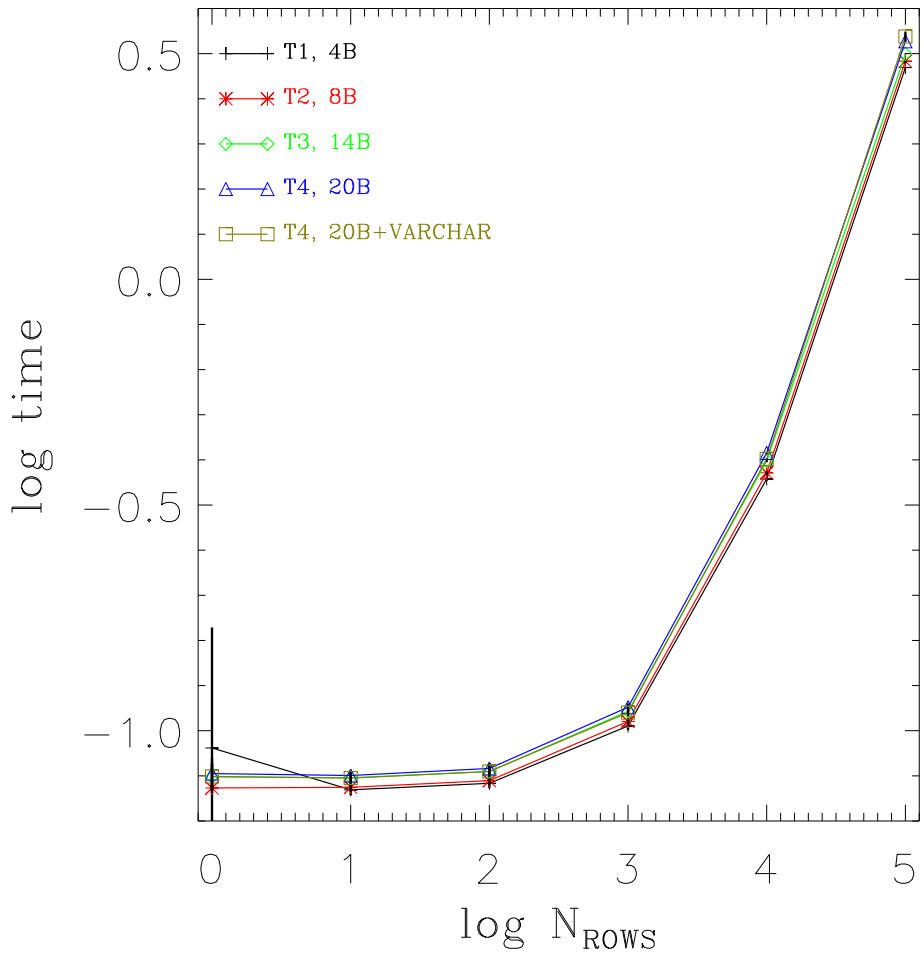

Figure 7.11: The dependence of the response time from the number of rows retrieved and the size of the table in the case of the local data retrieval. Time is in sec. (See Table F.1).

7.5 Minimization of the Size of the Test Database

To estimate the TCF for the desired database (DIVA/AMEX or GAIA mission database) we simulate the tables of the database and produce benchmark tests for the database. To save the disk space we have to reduce the size of the test database. This means that we have to decrease a number of attributes for each table and the number of rows in each table. The best solution would be the only table with the only attribute and the minimum number of rows. Still the benchmark produced with the use of the test database must be scaled to the case of the mission database.

Two relations have to be found for the minimization of the size of the test database: the dependence of the TCF on the size of the rows retrieved from the database (this will allow us to decrease the number of attributes in the table) and the dependence of the TCF on the number of rows retrieved (this will allow us to decrease the number of rows in the

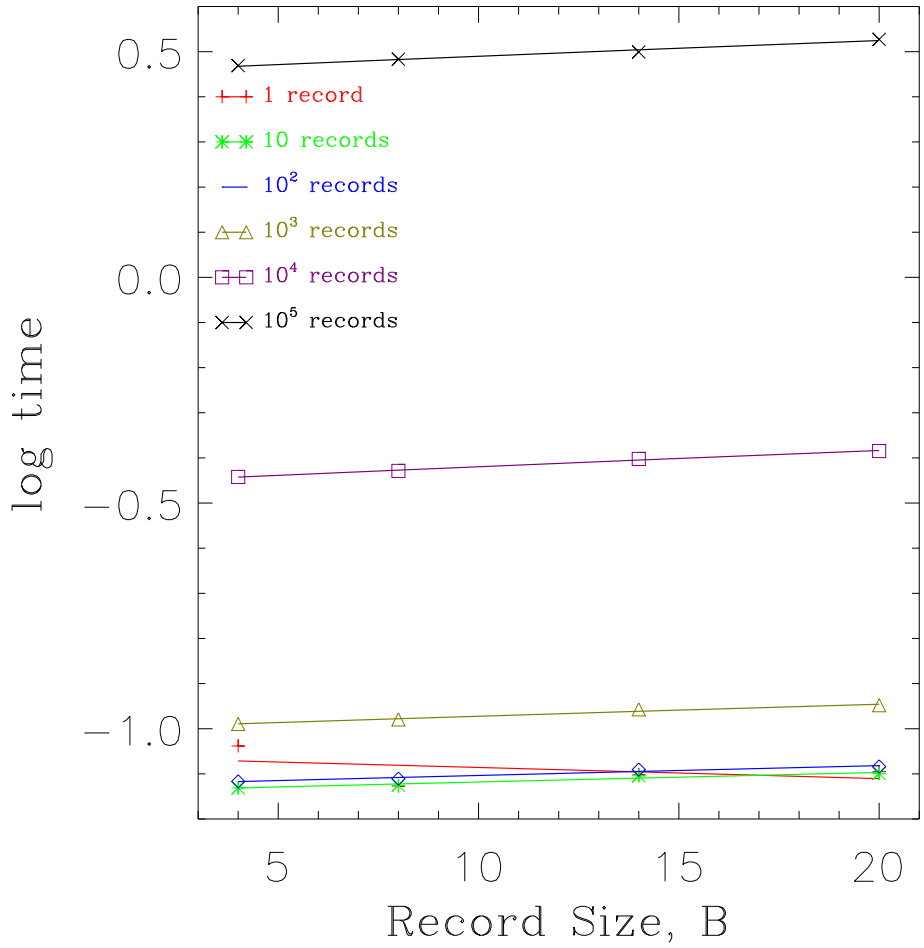

Figure 7.12: The approximation of the dependence of the response time from the size of the table in the case of the local data retrieval. (See Table F.1).

test database). The second question was already answered (see Fig. H). The minimum number of rows for the table of the test database is 10^4 per node (the TCF becomes constant).

To answer the first question let us assume that we have two tables: T_1 with attributes $\{S_1, S_2, \ldots S_j \ldots\}$ and T_0 with the only attribute $\{S_1\}$, and S_1 is a 4B INTEGER attribute. It is necessary to find a dependence which transforms the time required to retrieve N_0 rows from table T_0 to the time required to retrieve N_1 rows from table T_1. We also assume that the only indexed column in the table is S_1 and we have an ordered ascending index. We look for this relation in the form

$$t_1 = f(t_0, T_0, T_1, N_0, N_1)$$

and neglect the dependence from parameters which describe the state of the node (CPU overload, disk characteristics, RAM and others).

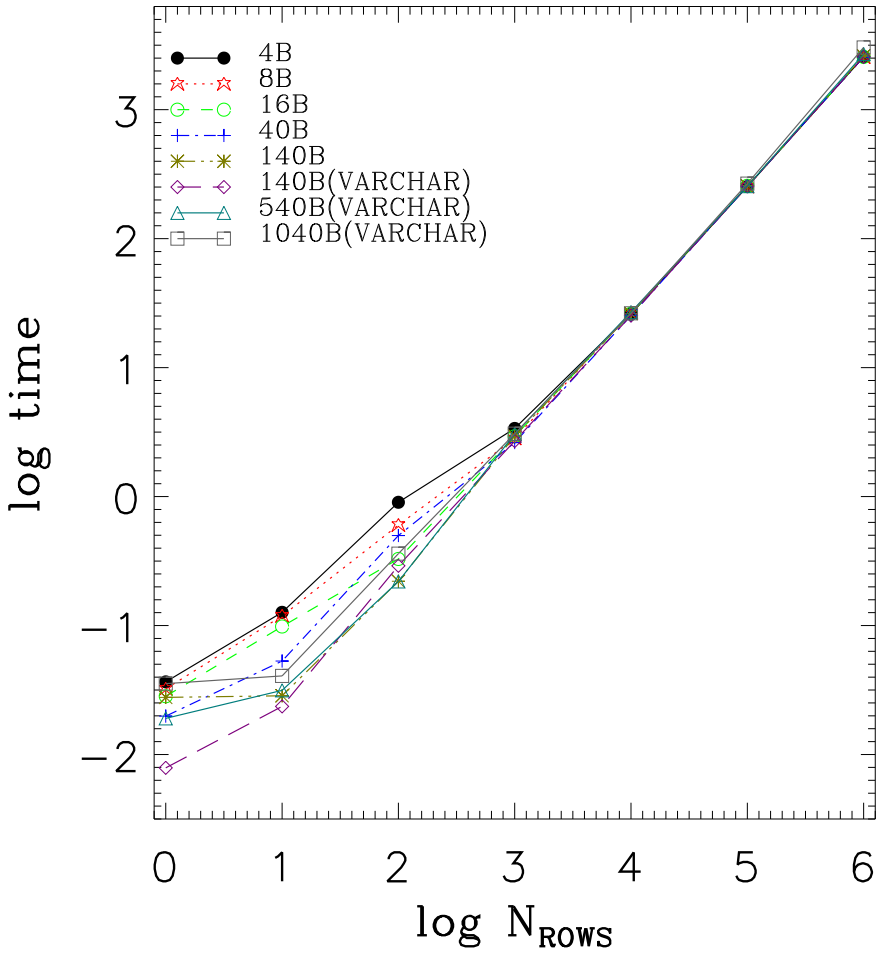

Figure 7.13: The dependence of the response time from the number of rows retrieved and the size of the table in the case of the remote data retrieval. Time is in sec. (See Table G.2).

The description of the response time we used previously is based on the linear dependence on the number of rows $t = A_0\, N_{rows}$ (see Section 7.1.3). To estimate the dependence of the response time from the size of the table we will suppose that $A_0 = A_0^0 + A_0^1 \,\mathrm{size(table)}$.

Appendices F and G describe tests which were made to find a dependence of the response time from the size of the table. The results for the local data retrieval are shown in Fig. 7.11 and Fig, 7.12, for the remote data retrieval in Fig. 7.13 and Fig, 7.14. The results shows that the response time depends on the number of rows retrieved and the size of the table, but at the same time the difference in response times for a 4B table and a 100B table does not exceed 2% and can be neglected. A small 4B table can be used for tests and the results can be used to optimize the data distribution for all tables. Nevertheless it is highly recommended to construct a dependence of the response time from the size of the table for each node in the cluster. If the distributed table is big enough (1kB, for example) and the number of rows placed at each node exceeds 10^6, the

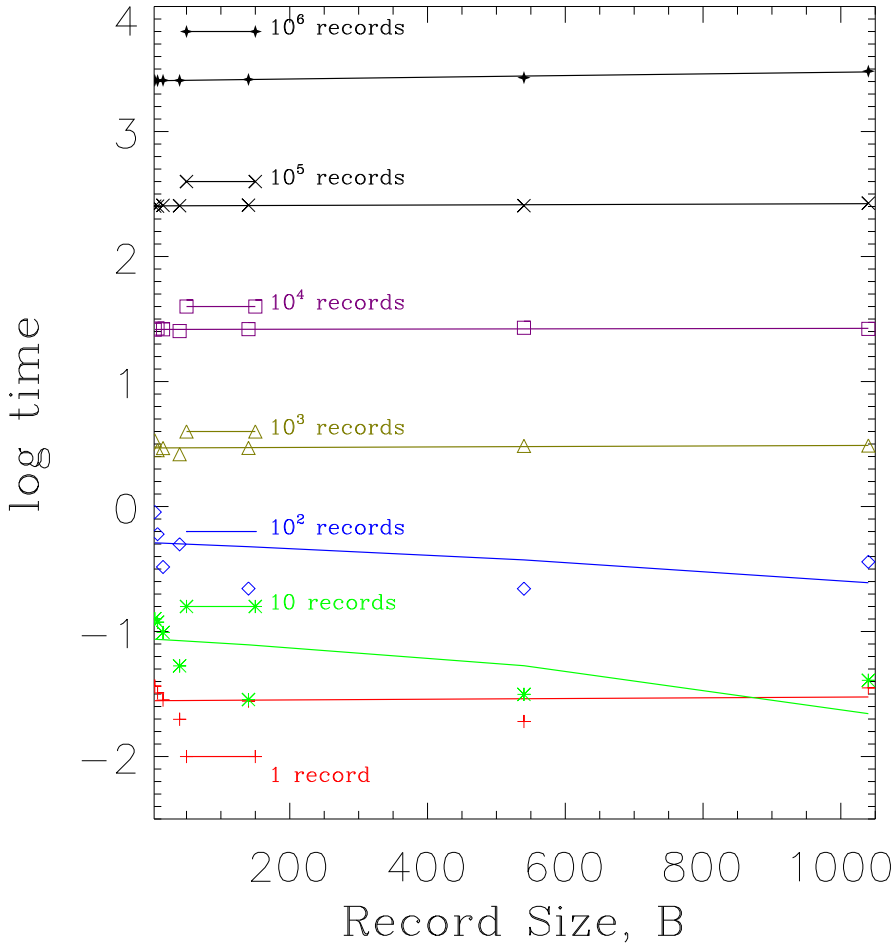

Figure 7.14: The approximation of the dependence of the response time from the size of the table in the case of the remote data retrieval. (See Table G.1).

difference in the response time can reach 10% and more. The use of the small test table in this case becomes more complicated, because the results of the test for each node must be scaled with the use of the constructed function TCF(N_{rec}, size(Table)) (see Table F.2 and Table G.1 in Appendices F and G for examples of the scale function).

7.6 Summary

The solution of the partitioning problem described in this chapter was used as a basis for the construction of the Database Statistic System (DBSS), which monitors the cluster and proposes an optimum data distribution. The DBSS is described in the next chapter.

Chapter 8

The Partitioning Problem: Practical Implementation of the Solution and Results

To realize the optimum physical data placement described in previous chapter we wrote a software system based on the use of the DB2 ESE DBMS. The system which we have to create (DB Statistic System, *DBSS*) must perform an optimal data placement, it must provide information about the present performance of the database and it must propose an optimal distribution of the data over the database nodes with the use of generated partitioning keys.

8.1 Requirements to the DBSS

At the beginning all functions of the DBSS are analyzed:

- local query on the node,
- remote query,
- insert results of the tests into the DBSS database,
- select the results of tests from the database,
- construction of the $\mathbf{F}+\mathbf{T}$ matrix,
- optimum solution for the constructed $\mathbf{F}+\mathbf{T}$ matrix,
- optimum data distribution,
- data load.

Fig. 8.1 shows the Use Case diagram for the DBSS.

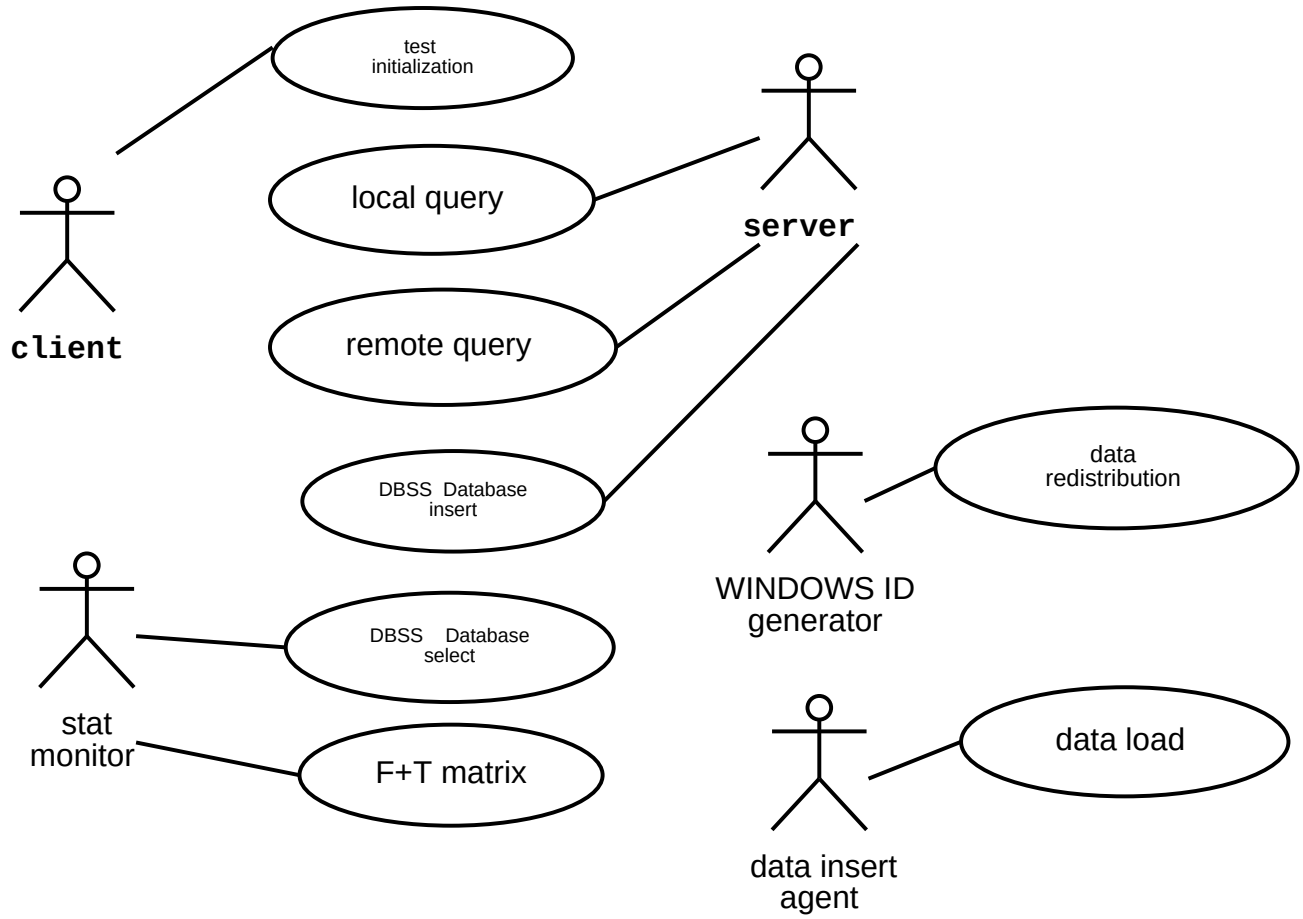

Figure 8.1: Use case diagram for the DBSS.

8.2 An Implementation of the DBSS for Astrometrical Databases. The Practical Solution of the Partitioning Problem.

DBSS can be subdivided into the three sections:

- the DBSS database which collects an information about the system,
- the optimum solution for the data placement,
- the data placement.

Fig. 8.4 shows the general scheme for the DBSS. To find the TCF (the $\mathbf{F}+\mathbf{T}$ matrix) for the tested system we run tests with the use of the DBSS database and store the results of the tests there. The duration of the test depends on the number of nodes used in the DBMS configuration. As we can see from the previous Chapter, it is possible to reduce the size of requested data (and the size of the table with the simulated data in the DBSS database) to 10^3 4B records. On our test cluster of 10 nodes (each node has

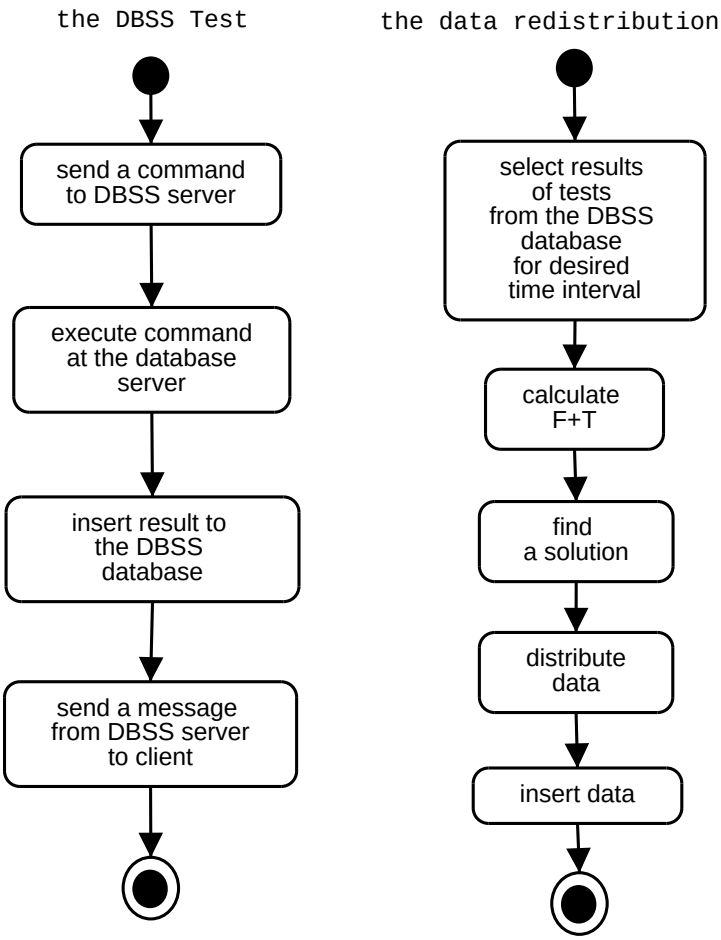

Figure 8.2: Activity diagram for the DBSS.

a dual Pentium IV 2.0 GHz CPU and two 73 GB SCSI hard disk) the duration of the test takes approximately 2 minutes. The frequency of the test depends on the schedule of the DPC and on the number of applications running on the cluster. The decision about the frequency of the test must be done by DBA of the system on the basis of the actual situation with the cluster and the frequency of the data upload to the database.

The activity diagram for the DBSS is shown in Fig. 8.2 and the sequence diagram in Fig. 8.3.

8.2.1 Tests and the DBSS database

The DBSS database consists of two tables (see Table 8.1) which collect information about tests done at the system. The structure of the database is very simple and reflects an idea of the whole system to manage the behavior of the DBMS with only one parameter – the time of query execution. Also we have to create a table with simulated data on each node of the cluster. This table consists of the only 4B INTEGER attribute and must have at least 10^3 rows.

The part of the software responsible for the test done by the DBSS consists of the client

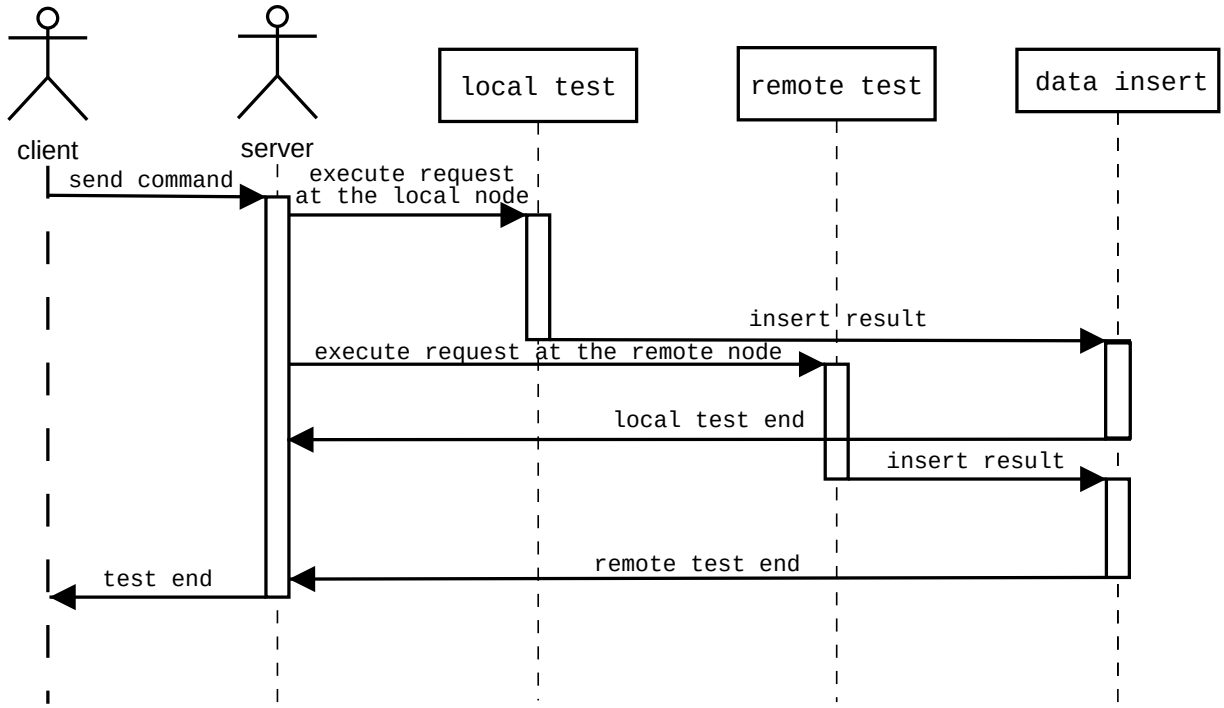

Figure 8.3: Sequence diagram for the DBSS.

and server sides. Both client and server are based on the use of Unix sockets and a TCP protocol. Both use a Unix socket system library (C++ class responsible for the use of sockets). The description of the library and an example of the client/server program can be found in [Chan, 1997].

The server side uses a C++ class as well, which encapsulates all SQL requests to the database.

The SQLStatement class

The SQLStatement class is used to create and execute a test SQL statement and insert the response time to the DBSS database.

```
class SQLStatement {
   private:
     char * database_name;
     char * table_name;
     char * from_statement;
     char * where_statement;
     char * inp_table_def;
     char * error_string;

     EXEC SQL BEGIN DECLARE SECTION;
     char into_class[1000];
```

```
    char header[1000];
    char where_class[1000];
    long longint;
    short shortint;
    struct {
     long testid;
     char descr[1000];
     char db_name[10];
     char table_name[10];
     char into_class[100];
     char sql_statement[1000];
    } BDB_DESC;
    struct {
     long testid;
     short node_at;
     short node_from;
     char time_begin[27];
     char time_end[27];
     double nrecords;
    } BDB_RES;
   char sql_string[500];
  EXEC SQL END DECLARE SECTION;

  public:
    SQLStatement(int test_id, int node_at, int node_from, char * descr,
     char * db_name, char * tb_name, char * fr_statement,
     char * wh_statement,char * i_table_def);
    int open_database ();
    int close_database ();
    int time_begin ();
    int time_end ();
    int make_query();
    int insert_DESC();
    int insert_RES();
    void WriteStatement();
    void TraceBDBRecords();
    char * error() {return error_string;};
};
```

The database and the table. The names of the database and table for the request are stored in the char arrays `database_name` and `table_name`.

The definition of the statement. The statement is stored in three char arrays: `from_statement` for the FROM-part of the request, `where_statement` for the circumstances of the request and `inp_table_def` for the description of the format of the requested table.

Description of the SQL failure. In case of the failure of the request SQLCA array is stored in the `error_string` char array.

The DBSS database structures. The structure of the DBSS database is encapsulated in two structures: `BDB_DESC` and `BDB_RES` corresponding to two tables of the DBSS database.

`open_database` **function.** This function realizes `CONNECT TO :database_name` SQL statement. In the case of a failure the SQLCA array stores in `error_string`.

`close_database` **function.** This function realize `CONNECT RESET` SQL statement. In case of failure the SQLCA array stores in `error_string`.

`time_begin` **function.** The `time_begin` inserts the current timestamp into the `BDB_RES.time_begin`.

`time_end` **function.** The `time_end` inserts the current timestamp into the `BDB_RES.time_end`.

`make_query` **function.** The `make_query` function executes the prepared SQL statement and stores the number of records retrieved into `BDB_RES.nrecords`.

`insert_DESC` **function.** This function inserts the description of the executed query into the DBSS database (`BDB_DESC` table).

`insert_RES` **function.** This function inserts the results of the query i into the DBSS database (`DBD_RES` table), i.e. the number of records retrieved and two timestamps: the first one for the start of the query and the last one for the end.

`WriteStatement` **function.** The function writes SQL statement which is executed to the standard error output.

`TraceBDBRecords` **function.** This function writes the `BDB_DESC` and `BDB_RES` records to the standard output.

`error` **function.** This function returns the error string `error_string`.

`SQLStatement` **initializer.** The SQLStatement is initialized with: the type of test (`int test_id`), the number of the node at which the test will be executed (`int node_at`), the number of the node from which the data will be retrieved (`int node_from`), the description of the test (`char * descr`), the database name (`char * db_name`), the table name (`char * tb_name`), the from-class of the SQL statement (`char * fr_statement`), the where-class of the SQL statement (`char * wh_statement`) and the description of the format of the table which will be used in the test (`char * i_table_def`).

Command File

The command file defines the tests which will be executed. The description includes the type of the statement (`[Statement_Type]`=1 – a command to execute the SQL statement, `[Statement_Type]`=-1 – a command to the server to finish the work), the targeted host for the test (the field `[Host]`), the node of the DBMS where the query will be executed (the `[Node]` field), the node where the data for the query stored (the field `[Node_From]`), the database to connect (`[Database]`), the table which will address the request (`[Table]`), the SQL statement divided into the where-class and from-class

Table 8.1: The DBSS Database.

Attribute	Designation	Domain
BDB_DESCR: Description of tests		
Test ID	TESTID	INTEGER
Description	DESCR	VARCHAR(1000)
Database	DB_NAME	CHAR(10)
Table	TABLE_NAME	CHAR(10)
INTO variables	INTO_CLASS	CHAR(100)
SQL statement	SQL_STATEMENT	VARCHAR(1000)
BDB_RES: Results of test		
Test ID	TESTID	INTEGER
Test node number	NODE_AT	SMALLINT
Control node number	NODE_FROM	SMALLINT
Timestamp at the beginning of the test	TIME_BEGIN	TIMESTAMP
Timestamp at the end of the test	TIME_END	TIMESTAMP
Number of records proceeded	NRECORDS	DOUBLE

([From_Class] and [Where_Class]) and how many times the request will be executed ([Ntimes]).

```
[Statement_Type]
1
[Host]
seth
[Node]
0
[Node_From]
0
[Database]
BDB
[Table]
TEST_MAIN
[From_Class]
select * from TEST_MAIN
[Where_Class]
where nw2>=0 and nw2< 10000
[Ntimes]
20
```

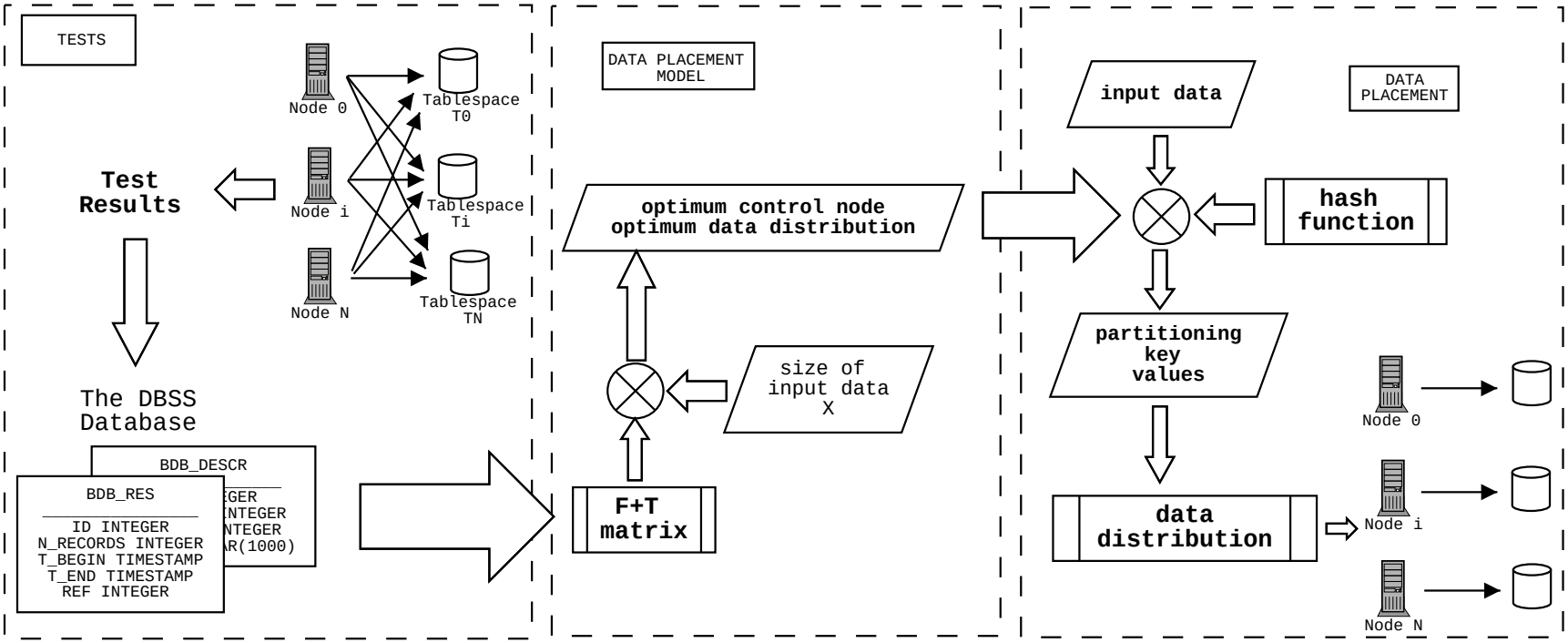

Figure 8.4: The principle scheme of the DBSS.

8.2.2 The Optimum Solution for the Data Placement

The calculation of the **F+T** matrix is done straightforward on the basis of the test results selected from the DBSS database for the desired time interval. The optimum control node and data distribution are proposed on the basis of the **F+T** matrix.

8.3 The Data Placement

The data placement is done in two steps: the first one is the generation of the partitioning key and the second one is the data load.

Unfortunately there is no `HASH` function for the DB2 linux version. Instead a spatial table `TABLE_HASH` must be distributed over the nodes of the cluster. The `TABLE_HASH` includes only one INTEGER column, which is used as a partitioning key for the distributed table.

The table with the partitioning key column distributed optimally is split using the `db2split` utility of DB2 and loaded with the `LOAD FROM FILE` command at each node.

8.4 The Implementation of the DBSS

8.4.1 The Tested Cluster

To test the developed DBSS we have selected three nodes of the cluster installed at the Astronomisches Rechen-Institut. We limited ourselves to the three nodes with the purpose to simplify the presentation of the results of the test.

Each node is a Pentium 4 2.2 GHz with 512 MB RAM and two 73GB SCSI disks. The OS was SuSE linux 7.2. DB2 UDB EEE v7.2 was installed at these nodes, the Gnu C++ compiler was used to compile the DBSS (gcc version 2.95.3).

8.4.2 The DBSS Database

The DBSS database was created on three node: two tables to store the results of the test (`BDB_RES` and `BDB_DESCR`) and three tables with the simulated data (one table at each node, each table has the only 4B indexed INTEGER attribute and was filled with 10^5 rows). The requests from each node of the cluster were issued by the DBSS client each half an hour to select 10^5 rows locally and remotely at each node.

Fig. 8.5 shows the results. The statistic for 10 days was used to make a redistribution of the data. The **F+T** matrix was constructed on the basis of these 10 days statistic (see Table 8.2).

We used the method described in the Section 7.2 to find the optimum control node (node 0 in our case) and the optimum data distribution. From the matrix **F+T** the optimum data placement is 98 % of the rows for the node 0, 1% of the rows for the node 1 and 1% of the rows for the node 2.

Table 8.2: The **F**+**T** matrix with dispersion.

Node of the data retrieval	Node of the data placement		
	0	1	2
0	$(5.47 \pm 0.91)\,10^{-5}$	$(8.18 \pm 0.69)\,10^{-3}$	$(8.67 \pm 0.83)\,10^{-3}$
1	$(8.16 \pm 0.90)\,10^{-3}$	$(8.12 \pm 0.92)\,10^{-5}$	$(9.65 \pm 0.95)\,10^{-3}$
2	$(8.57 \pm 0.70)\,10^{-3}$	$(9.81 \pm 0.94)\,10^{-3}$	$(5.49 \pm 0.77)\,10^{-5}$

8.4.3 The Proof of the Results

The comparison was made between proposed data distribution, standard one (uniform) and the single-node configuration (100 % of the data on one node).

The test tables were created at three nodes. The table `TEST_MAIN` was used to distribute data corresponding to the proposed data distribution, the table `TEST_BAD` for the uniform data distribution and the table `TEST_LONE` for the single-node data distribution.

```
create NODEGROUP NG012 on nodes (0,1,2)@

create tablespace TS_BDB IN NODEGROUP NG012
managed by database  using (FILE '/home/data/bdb_sp1' 500000)@

create tablespace IS_BDB IN NODEGROUP NG012
managed by database using (FILE '/home/data/bdb_isp1' 500000)@


create table TEST_MAIN
(
    NW1          INTEGER               not null ,
    NW2          INTEGER               not null,
    T1           INTEGER                       ,
    T2           INTEGER                        ,
    CX           SMALLINT                       ,
    CY           SMALLINT                       ,
    WT           SMALLINT                       ,
    W            VARCHAR(20),
    primary key (NW2)
) IN TS_BDB INDEX IN IS_BDB
@

create table TEST_BAD
(
    NW1          INTEGER               not null ,
    NW2          INTEGER               not null,
    T1           INTEGER                       ,
```

```
    T2          INTEGER                         ,
    CX          SMALLINT                        ,
    CY          SMALLINT                        ,
    WT          SMALLINT                        ,
    W           VARCHAR(20),
    primary key (NW2)
) IN TS_BDB INDEX IN IS_BDB
@

create table TEST_LONE
(
    NW1         INTEGER              not null ,
    NW2         INTEGER              not null,
    T1          INTEGER                      ,
    T2          INTEGER                       ,
    CX          SMALLINT                      ,
    CY          SMALLINT                      ,
    WT          SMALLINT                      ,
    W           VARCHAR(20),
    primary key (NW2)
) IN TS_BDB INDEX IN IS_BDB
@

create index IN_MAIN on TEST_MAIN (NW1 ASC)@
create index IN_BAD  on TEST_BAD (NW1 ASC)@
create index IN_LONE  on TEST_LONE (NW1 ASC)@
```

The number of records in the tables is 10^6 NW1 value runs from 0 to $10^6 - 1$.

Request

Three SQL statements were executed consequently:

```
SELECT * FROM TEST_MAIN where I1>=0 and I1< I_MAX
```

```
SELECT * FROM TEST_BAD where I1>=0 and I1< I_MAX
```

```
SELECT * FROM TEST_LONE where I1>=0 and I1< I_MAX
```

where $I\_MAX \in [1, 10, 10^2, 10^3, 10^4, 10^5, 10^6]$. Measurements for each point were made 5 times.

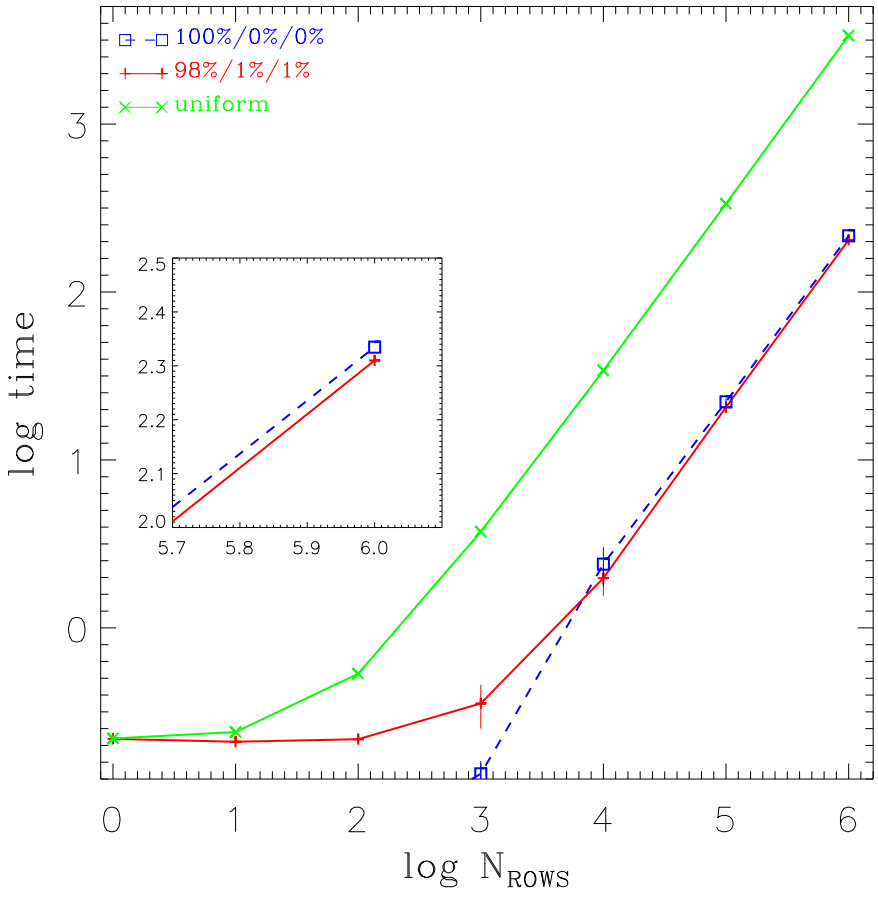

Figure 8.5: The response time for the uniform data distribution, the optimum data distribution and the single-node distribution. Time is in sec. (See Table 8.3).

Results

The results of the test shows that even in the case of the homogeneous cluster we can improve the performance of the data retrieval significantly. Indeed, if we use the cluster which is overloaded with the work like the ARI cluster (the CPU usage at each node was close to 100 % during the test as the cluster was loaded with the N-body simulation) and does not care to distribute data properly we will have 20 times slower response time compared to the use of the optimum data distribution (see Fig. 8.5 and Table 8.3). We also can not reject the use of the distributed database as the use of the single node of the cluster for the storage of the data (100 % at the single node) shows an increase of the response time compared with the optimum data distribution for the distributed database.

8.5 Summary

The Database Statistic System (DBSS) was created and tested. The test was performed on the cluster which was the prototype of the DIVA/AMEX cluster – 10 nodes shared-nothing cluster with DB2 UDB EEE installed. During the test the cluster was loaded

Table 8.3: The response time for various data distributions.

The number of records retrieved	The response time with dispersion, sec, for data distribution		
	uniform	98%/1%/1%	100%/0%/0%
1	0.220 ± 0.012	0.218 ± 0.008	0.053 ± 0.018
10	0.240 ± 0.006	0.210 ± 0.004	0.049 ± 0.011
10^2	0.533 ± 0.006	0.217 ± 0.007	0.051 ± 0.012
10^3	3.738 ± 0.019	0.355 ± 0.104	0.135 ± 0.026
10^4	34.106 ± 0.122	1.979 ± 0.425	2.396 ± 0.624
10^5	335.654 ± 0.869	20.612 ± 1.563	22.200 ± 1.101
10^6	3362.795 ± 8.089	204.174 ± 3.346	216.087 ± 3.907

with the work which suites perfectly to the situation of the DIVA/AMEX cluster shared with the non-database applications.

We proved, that the use of the proposed data distribution with the switching control node (which is determined on the basis of the $\mathbf{F}+\mathbf{T}$ matrix) improves the response time significantly and makes it possible to use database to store and retrieve data for time-critical applications.

Chapter 9

A Benchmark for an Astrometrical Database

The construction of a special benchmark for the case of an astronomical database targets a number of goals. First of all, we have to estimate the efficiency of the data placement in case of the time-critical requests. Also, we have to estimate the overall performance of the designed software and hardware systems. Finally, we need to compare the cost of different hardware solutions.

The data retrieval in the DPC can be done by two ways: the retrieval of the data record-by-record (each record by a separate transaction) and the retrieval of the whole data chunk in a single transaction. The *transaction* in case of the data retrieval in the DPC can vary from process to process and can contain the all retrieved data (in the case of Pixel Data Processing, for example) or a single record (in case of Object Recognition). In our case to measure the efficiency the tps (transactions per second) is not suitable, but we will use the rps (records per second) measure. This is more suitable from the point of view of the application (pipeline) and allows to estimate the time required by the application to access data.

The construction of the benchmark based on [The Benchmark Handbook, 1993].

9.1 The Application Environment

The benchmark supposes the use of the chained processes with access to the data. The description of processes can be found in the Chapter 2. The most important feature is the absence of update operations over database. Only Pixel Data Processing and Preliminary Identification can be run in parallel, all other processes require an input from the preceeding process in the processing chain.

The DBMS is fixed on the RDBMS. The example of the benchmark was realized for the DB2 UDB EEE v7.2.

The processes can access data remotely or locally from one of the nodes of the cluster, so both “remote” and “local” terminals must be implemented.

9.2 The Logical Design of the Database

The design of the database is based on the attributes and relations described in Chapter 5 and Chapter 6. We simplified the structure of the database to reduce the its size, whereas the structure of the database remains.

9.2.1 Attributes, Relations and the General Structure of the Database

The structure of the database is based on 5 tables only (instead of 13 tables in Chapter 6): the raw data table, the processed data table and the final catalog table and on the two tables for relations: raw data – processed data table and processed data – final catalog table.

The relations between attributes are of the type one-to-many types: a single raw data row can correspond to a number of processed data rows and a number of processed data rows correspond to a single final catalog row.

9.2.2 The Transaction Profile

It is clear from the processing chain of the mission that we need to optimize the select statements mainly. The select statements will take records from tables loaded by the previous process in the processing chain or from the incoming raw data.

The definitions of select and insert statements were made in Chapter 3. The combination of the requests with the corresponding weights can be used as a basis for the estimation of the overall performance of the database during the mission.

To simplify the description of the operations on the database (the full description for each process can be found in Chapter 6) we propose a general scheme of data processing:

RI raw data insert,

RS1 raw data select by the current number of the scanning circle,

PI processed data insert,

PS1 processed data select by the number of the scanning circle,

PS2 processed data select by the time interval,

PS3 processed data select for the corresponding windows (using the raw data record - processed data record relation),

FI final catalog data insert,

FS1 final catalog data select by the object identifier,

FS2 final catalog data select for the corresponding images,

FS3 final catalog data select for the corresponding windows.

Table 9.1: The benchmark database.

Attribute	Designation	Domain
RD: Raw Data		
Window	NW1	INTEGER
Identifier	NW2	INTEGER
Time	T1	INTEGER
Count	T2	INTEGER
CCD chip X-coordinate	CX	SMALLINT
CCD chip Y-coordinate	CY	SMALLINT
Window type	WT	SMALLINT
Window	W	VARCHAR(2184)
PD: Processed Data		
Image	NI1	INTEGER
Identifier	NI2	INTEGER
Time	T1	INTEGER
Count	T2	INTEGER
Coordinate Parameter 1	CP1	INTEGER
Coordinate Parameter 2	CP2	INTEGER
Coordinate 1	CO1	DOUBLE
Error of coordinate 1	ECO1	DOUBLE
Coordinate 2	CO2	DOUBLE
Error of coordinate 2	ECO2	DOUBLE
FD: Final catalog		
Object	NB1	INTEGER
Identifier	NB2	INTEGER
Position, α	ALF	DOUBLE
Position, δ	DEL	DOUBLE
Position error, σ_α	EAL	DOUBLE
Position error, σ_δ	EDE	DOUBLE
Proper motion, μ_α	MUA	INTEGER
Proper motion, μ_δ	MUD	INTEGER
Proper motion error, σ_{μ_α}	EMA	INTEGER
Proper motion error, σ_{μ_δ}	EMD	INTEGER
Parallax	PAR	INTEGER
Parallax error	EPA	INTEGER
RDPD: Raw Data - Processed Data Links		
Window	NW1	INTEGER
Identifier	NW2	INTEGER
Image	NI1	INTEGER
Identifier	NI2	INTEGER
PDFD: Processed Data - Final Catalog Links		
Image	NI1	INTEGER
Identifier	NI2	INTEGER
Object	NB1	INTEGER
Identifier	NB2	INTEGER

Finally we will have following statements to execute for the benchmark:

RI: Raw data insert.

INSERT INTO RD VALUES(NW1, NW2, T1, T2, CX, CY, WT, W)

RS1: Raw data select. Sometimes users need data ordered by a relation between child and parent windows. In the example below the program receives information about the parent window for each object window as well.

SELECT NW1, NW2, T1, T2, CX, CY, WT, W FROM RD
WHERE NW1 >= i AND NW1 < i+3 ORDER BY NW1, NW2

PI: Processed data insert. We will have a number of statements which will insert processed data from the pipeline and other applications into the database.

INSERT INTO PD VALUES(NI1,NI2,T1,T2,CP1,CP2,CO1,ECO1,CO2,ECO2)

Select statement to the tables of the processed data can be divided by type of requested attribute: image identifier, time count or object identifier.

PS1: Processed data select by the number of the scanning circle.

SELECT NI1,NI2, T1, T2, CP1,CP2,CO1,ECO1,CO2,ECO2 FROM PD
WHERE NI1 >= i AND NI1 < i+3 ORDER BY NI1, NI2

PS2: Processed data select by the time interval.

SELECT NI1,NI2, T1, T2, CP1,CP2,CO1,ECO1,CO2,ECO2 FROM PD
WHERE T1 >= $T1_{begin}$ AND T1 < $T1_{end}$ ORDER BY T1, T2;

PS3: Processed data select for the corresponding windows.

SELECT Y.NW1, Y.NW2, X.NI1, X.NI2, X.T1, X.T2, X.CP1, X.CP2, X.CO1,
X.ECO1, X.CO2, X.ECO2
FROM PD X, RDPD Y
WHERE Y.NW1>= $NW1_{begin}$ AND Y.NW1 < $NW1_{end}$ AND X.NI1=Y.NI1 AND
X.NI2=Y.NI2
ORDER BY Y.NW1, Y.NW2;

FI: Final catalog data insert. The insert statement for the final data.

INSERT INTO FDFC VALUES(PRG, NB1, NB2, ALF, DEL, EAL, EDE, MUA, MUD,
EMA, EMD, PAR, EPA, FB1, EFB1, FB2, EFB2)

FS1: Final catalog data select by the object identifier.

SELECT NB1, NB2, ALF, DEL, EAL, EDE, MUA, MUD, EMA, EMD, PAR, EPA
FROM FC
WHERE NB1>=$NB1_1$ AND NB1<$NB1_2$ AND NB2>=$NB2_1$ AND NB2<$NB2_2$
ORDER BY NB1, NB2

FS2: Final catalog data select by the image identifier.

SELECT X.NB1, X.NB2, X.ALF, X.DEL, X.EAL, X.EDE, X.MUA, X.MUD, X.EMA,
X.EMD, X.PAR, X.EPA, Y.NI1, Y.NI2
FROM FC X, PDFC Y
WHERE X.NB1=Y.NB1 AND X.NB2=Y.NB2 AND Y.NI1>=$NI1_1$ AND Y.NI1<$NI1_2$
AND Y.NI2>=$NI2_1$ AND Y.NI2<$NI2_2$
ORDER BY X.NI1, X.NI2

FS3: Final catalog data select by the window identifier.

SELECT X.NB1, X.NB2, X.ALF, X.DEL, X.EAL, X.EDE, X.MUA, X.MUD, X.EMA,
X.EMD, X.PAR, X.EPA, Y.NI1, Y.NI2, Z.NW1, Z.NW2
FROM FC X, PDFC Y, RDPD Z
WHERE X.NB1=Y.NB1 AND X.NB2=Y.NB2 AND Y.NI1=Z.NI1 AND Y.NI2=Z.NI2
AND AND Z.NW1>=$NW1_1$ AND Z.NW1<$NW1_2$ AND Z.NW2>=$NW2_1$ AND
Z.NW2<$NW2_2$
ORDER BY Z.NW1, Z.NW2

The statements described above have different importance in the calculation of the benchmark. RS1 and PS1 are time-critical and the most important measures in the benchmark.

Each select statement must be realized as a static cursor statement running over all selected rows. The begin of the execution of the statement starts with the declaration of the cursor end ends with the close of the cursor.

9.3 Scaling Rules

The number of rows in each table are governed by the daily input into the raw data table and the multiplicity factor for processed and final data tables. Usually we will have only 10% of cases with two or more images in a normal window, as a result, the number of rows in the processed data table must be scaled as $N_{PD1} = 1.1\, N_{RD}$, and this value has to be used for the Processed Data Insert. The Processed Data Select must deal with the whole volume of the processed data, so the number of rows in the processed data table will be $N_{PD2} = 100\, T_{mission}\, N_{FC}$, where $T_{mission}$ is the mission duration in years, N_{FC} is the number of objects in the final catalog and we supposed that each star will be observed at least 50 times for half a year (as it would be in the case of DIVA/AMEX, see [TD0284-01, 2002]). The number of objects in the final catalog is calculated on the basis of the characteristics of satellite's instruments (see [TD0201-05, 2002] for the case of DIVA/AMEX and [de Boer et al., 2000] for the case of GAIA). Table 9.2 reviews the scaling rules for the benchmark database.

Table 9.2: The size of raw, processed and final data tables, in number of records (based on [TD0201-05, 2002]for DIVA/AMEX, [de Boer et al., 2000] for GAIA)

	DIVA/AMEX	GAIA
Raw Data Table	$1.5\,10^6$ per scan	$3\,10^8$ per scan
Processed Data Table	10^7 per half year	$1.5\,10^{10}$ per half year
Final Catalog Table	$4\,10^7$	10^9

9.4 The Atomicity, Consistency, Isolation and Durability Properties

The Atomicity requirements for the transaction can be satisfied, because there are no update operations. Insert operations must be done before any other operations over the database will be initiated and the only operation with the database is a select.

The only possible failure in the atomicity can occur in the case of the insert operation. In this case the situation is classified as a software failure, the insert must be canceled, the database must be reset to the pre-insert state and insert operation must be repeated. This situation is not typical for the database and will not be considered in the benchmark.

The Consistency of the database can be broken only during the insert of the data or in the case of a hardware/software failure (the latter will not be considered). The insert of the data must add a new portion to the database but not modify the data already loaded. This condition can be checked by the existence of unique keys for each table of the data : (NW1,NW2) for raw data, (NI1,NI2) for processed data, (NB1,NB2) for the final catalog. The consistency of the tables of relations can be estimated based on the relationships between entities themselves: in case of RDPD the unique (NW1,NW2) will correspond to one or many (NI1,NI2) and the unique (NI1,NI2) will correspond to the single (NW1,NW2); in the case of PDFD the unique (NI1,NI2) will correspond to the only (NB1,NB2) and the unique (NB1,NB2) will correspond to one or many (NI1,NI2).

The Isolation of the tables will be set in the case of the data load. Any other operations over the database will not influence the isolation level of any table in the database. The data processing chain does not require simultaneous data access for the select and insert operations.

The Durability of the database is based on the stable content of tables. The time required for the restoration of the database depends on the type of the failure that occurs in the system. Basically the restoration is based on the archived copy of the inserted data portion. Possible failures and the way to restore a database was described in Chapter 6.

9.5 Partitioning

The only type of the partitioning which is allowed during the test is a horizontal one. The records can be distributed through nodes uniformly or in accordance with the method

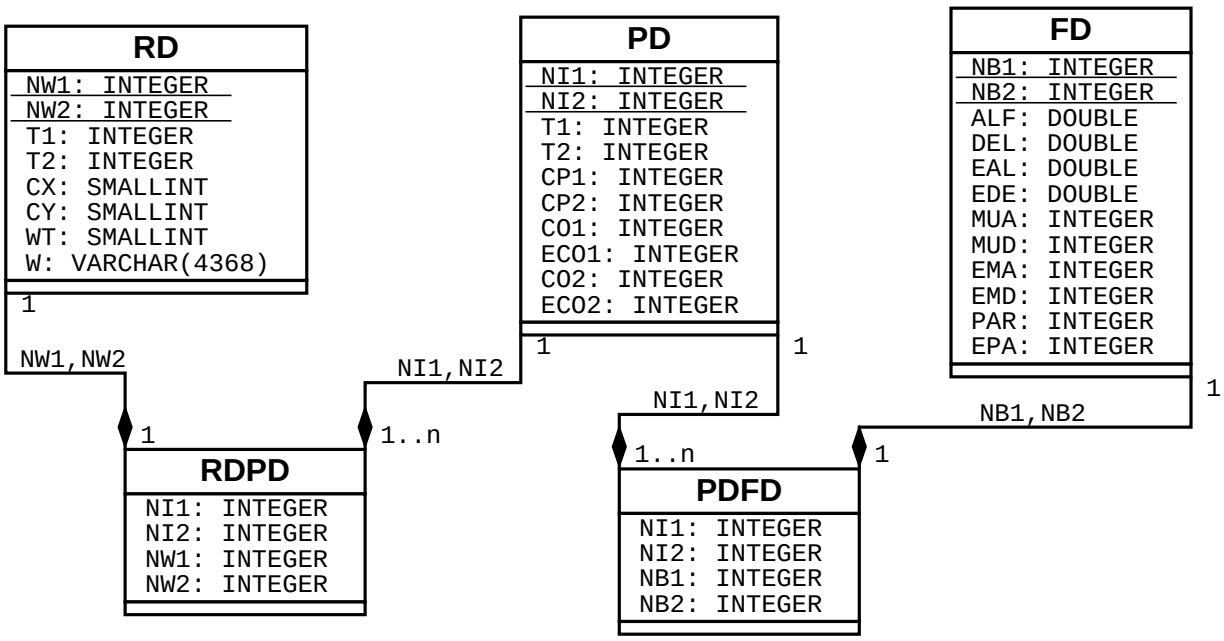

Figure 9.1: The UML scheme of the Benchmark Database.

described in Chapter 7. This method can be applied to the partitioning keys of the raw data table (NW1,NW2), whereas the other two keys are generated on the basis of coherency of the partitioning keys of three tables. As a result we can have a slight disbalance (20% of raw data records can correspond to 2 records of processed data) in the partitioning of the processed data.

9.6 The Generation of the Input Data

The generation of the input data must be done in to steps:

1. The generation of the RD and FD tables with unique identifiers for (NW1,NW2) and (NB1,NB2). The generation of the RD.W attribute must be done with all ASCII character set (each character in the VARCHAR string has an integer value in the range from 0 to 255 and must be randomly selected).

2. The generation of the PD table with unique identifiers (NI1,NI2) and parallel generation of two tables RDPD and PDFD. For the table RDPD in 20 % of records (NW1,NW2) must correspond to two different (NI1,NI2). For the table PDFD each (NB1,NB2) must correspond to 100 (NI1,NI2).

The range and the way of the generation of values is described in the Table 9.3.

Table 9.3: Benchmark input data generation.

Attribute	Designation	Distribution
RD: Raw Data		
Window	NW1	0
Identifier	NW2	[0,$N_{RECORDS}-1$]
Time	T1	random, [0,2000000]
Count	T2	random, [0,2000000]
CCD chip X-coordinate	CX	random, [0,2047]
CCD chip Y-coordinate	CY	random, [0,2047]
Window type	WT	random, [0,13]
Window	W	see text
PD: Processed Data		
Image	NI1	0
Identifier	NI2	[0,1.2 $N_{RECORDS}-1$]
Time	T1	random, [0,2000000]
Count	T2	random, [0,2000000]
Coordinate Parameter 1	CP1	random, [0,10]
Coordinate Parameter 2	CP2	random, [0,10]
Coordinate 1	CO1	random, [0.0,360.0)
Error of coordinate 1	ECO1	random, (0,0.001]
Coordinate 2	CO2	random, [0.0,180.0)
Error of coordinate 2	ECO2	random, (0,0.001]
FD: Final catalog		
Object	NB1	0
Identifier	NB2	[0,$N_{OBJECTS}-1$]
Position, α	ALF	random, [0.0,360.0)
Position, δ	DEL	random, [-180.0,180.0]
Position error, σ_α	EAL	random, (0,0.001]
Position error, σ_δ	EDE	random, (0,0.001]
Proper motion, μ_α	MUA	random, [-1000.0,1000.0]
Proper motion, μ_δ	MUD	random, [-1000.0,1000.0]
Proper motion error, σ_{μ_α}	EMA	random, [0.0, 10.0]
Proper motion error, σ_{μ_δ}	EMD	random, [0.0, 10.0]
Parallax	PAR	random, [-1.0,1.0]
Parallax error	EPA	random, [0.0, 0.001]

9.7 Response Time

Measurements of control times must be provided for two cases: a client mode (measurement of the control time on the host which executed an application) and a server mode (the time required for the database server to retrieve a data chunk at the request of an application locally).

The measured control times for each select statement can be treated individually or as

a sum with some weight coefficients as an estimation of the overall performance of the database, the software and the hardware.

9.7.1 Specification of Control Times

The response time is specified as $RT = T_2 - T_1$, where T_1 is the begin of the execution of the SQL statement inside the application (declaration of the cursor) and T_2 is the end of the execution (close cursor). The data retrieved by the application in case of local response time were not displayed or written to file or device but stored in variables defined by the application.

9.7.2 Computation of the Response Time

The computation of the response time must be done for each statement (like PS1) on the basis of 100 consequent executions of the statement. The dispersion of the measured response time must be computed:

$$t_{PS1}[sec] =< t > \pm\sigma_t.$$

9.7.3 Computation of rps

The rps value must be calculated on the basis of the response time and the number of records retrieved. Additionally for the RD statement RS1 the dependence of the rps on the number of records must be tested. RS1 must be executed with 1, 10, 10^2, 10^3, 10^4, 10^5, 10^6 records, at least 5 measurements must be provided for every point. The result has to be plotted with axes $\log N_{RECORDS} - \log rps$ with errors for rps.

9.8 System Under Test Definition

9.8.1 Models of the Target System

The Target System can be a single server or a cluster. In the case of a single PC the local and remote terminals are the same. In the case of a cluster the remote terminal must be executed at a server outside of the cluster and the local terminal at one of the nodes of the cluster.

9.8.2 Hardware Definition

The hardware used in the test must be described as following:

1. Server/cluster nodes: CPU and memory,
2. Data storage media (usually hard disks) attached to nodes of the server: IDE/SCSI, capacity, model.

3. Network hardware (Ethernet cards, Myrinet cards) : transfer rate.

9.8.3 Software Definition

The description of the software components used in the test:

1. DBMS: type of the DBMS, version.
2. Programming language for the realization of the benchmark: language, manufacture, version.
3. Communication interface for the terminal-DBMS server and between database servers of the DBMS in the case of a distributed database.

9.8.4 DBMS configuration

The configuration of the DBMS must be described in the following way:

1. The server configuration: number of servers on each node.
2. The tablespaces configuration.

9.9 Pricing

The cost for the tested system must be detached on the cost of the hardware component and software components of the system. In addition the cost of the storage media can be supplied if available.

The cost/rps measure must be provided for the PS1 statement.

9.10 Implementation and Results

The implementation of the benchmark is described below.

9.10.1 System Under Test

The targeted system is the single node.

Hardware Definition: Pentium IV 2.0 GHz with 512 MB RAM, SCSI 73GB hard disk.

Software Definition: DB2 UDB EEE v7.2, C++ (g++ 3.2.2), OS: SuSE linux v8.0.

DBMS Configuration: single-node DB2 UDB EEE v7.2.

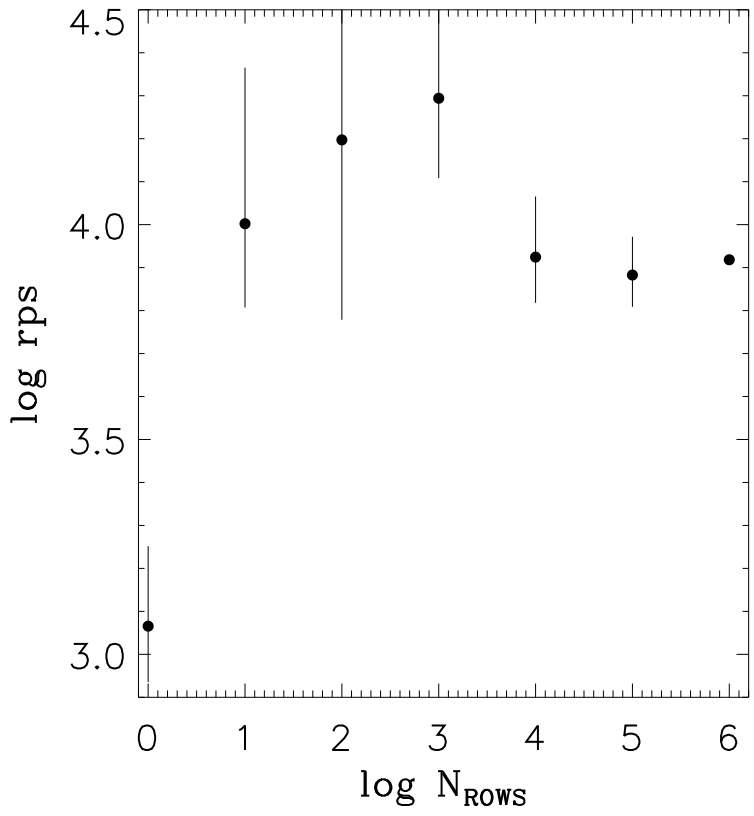

Figure 9.2: Benchmark. The records per seconds measure for RS1 statement. Errors are shown. Time is in sec. (See Table 9.5).

9.10.2 Benchmark Database

The benchmark database was created with the use of the following script:

```
create nodegroup NG00 on nodes (0) @

create tablespace T00_BDB IN NODEGROUP NG00
managed by database using (FILE '/work/data2/t00.t1' 512000)@

create tablespace I00_BDB IN NODEGROUP NG00
managed by database using (FILE '/work/data2/t00.i1' 512000)@

create table RD
(
    NW1          INTEGER,
    NW2          INTEGER,
    T1           INTEGER,
    T2           INTEGER,
```

```
    CX          SMALLINT,
    CY          SMALLINT,
    WT          SMALLINT,
    W           VARCHAR(2000)
) IN T00_BDB INDEX IN I00_BDB
@
create index IDX_NW1 on RD (NW1 ASC)@
create index IDX_NW2 on RD (NW2 ASC)@

create table PD
(
    NI1         INTEGER,
    NI2         INTEGER,
    T1          INTEGER,
    T2          INTEGER,
    CP1         DOUBLE,
    CP2         DOUBLE,
    CO1         DOUBLE,
    ECO1        DOUBLE,
    CO2         DOUBLE,
    ECO2        DOUBLE
) IN T00_BDB INDEX IN I00_BDB
@
create index IDX_NI1 on PD (NI1 ASC)@
create index IDX_NI2 on PD (NI2 ASC)@

create table FD
(
    NB1         INTEGER,
    NB2         INTEGER,
    ALF         DOUBLE,
    DEL         DOUBLE,
    EAL         DOUBLE,
    EDE         DOUBLE,
    MUA         DOUBLE,
    MUD         DOUBLE,
    EMA         DOUBLE,
    EMD         DOUBLE,
    PAR         DOUBLE,
    EPA         DOUBLE
) IN T00_BDB INDEX IN I00_BDB
@
create index IDX_NB1 on FD (NB1 ASC)@
create index IDX_NB2 on FD (NB2 ASC)@

create table RDPD
```

```
(
    NW1         INTEGER,
    NW2         INTEGER,
    NI1         INTEGER,
    NI2         INTEGER
) IN T00_BDB INDEX IN I00_BDB
@
create index IDX_RDPD_NW1 on RDPD (NW1 ASC)@
create index IDX_RDPD_NW2 on RDPD (NW2 ASC)@
create index IDX_RDPD_NI1 on RDPD (NI1 ASC)@
create index IDX_RDPD_NI2 on RDPD (NI2 ASC)@

create table PDFD
(
    NI1         INTEGER,
    NI2         INTEGER,
    NB1         INTEGER,
    NB2         INTEGER
) IN T00_BDB INDEX IN I00_BDB
@
create index IDX_PDFD_NI1 on PDFD (NI1 ASC)@
create index IDX_PDFD_NI2 on PDFD (NI2 ASC)@
create index IDX_PDFD_NB1 on PDFD (NB1 ASC)@
create index IDX_PDFD_NB2 on PDFD (NB2 ASC)@
```

The number of records for tables: RD 10^6, PD 10^4, FD 10^4, RDPD $1.2\,10^4$, PDFD $5\,10^5$. The rules for the data generation are described in the Chapter 9.

9.10.3 Control Times

The control times for each benchmark statement are listed in Table 9.4. The dependence of the response time on the number of rows is shown in Table 9.5 and Fig. 9.2.

9.10.4 Pricing

The pricing for the tested single-node system is 2000 USD/ 8000 records per second.

9.11 Summary

The benchmark for an astrometric database was developed to provide a basis for the comparison between different hardware and software solutions for the realization of the astrometrical data processing.

Table 9.4: The response time and rps for benchmark statements

The statement	The response time per record with dispersion, sec	rps, rows per second
RS1	$(1.20 \pm 0.32)\,10^{-4}$	8983 ± 2048
PS1	$(2.93 \pm 0.65)\,10^{-5}$	35594 ± 6690
PS2	$(2.54 \pm 0.30)\,10^{-5}$	39850 ± 4270
PS3	$(1.62 \pm 0.26)\,10^{-4}$	6303 ± 908
FS1	$(4.39 \pm 0.60)\,10^{-5}$	23143 ± 2587
FS2	$(1.26 \pm 0.13)\,10^{-3}$	804 ± 78
FS3	$(1.85 \pm 0.12)\,10^{-3}$	544 ± 35

Table 9.5: The response time for RS1.

The number of records	The response time with dispersion, sec
1	$(8.6 \pm 3.0)\,10^{-4}$
10	$(9.95 \pm 5.64)\,10^{-4}$
10^2	$(6.35 \pm 10.30)\,10^{-3}$
10^3	$(5.08 \pm 2.71)\,10^{-2}$
10^4	1.19 ± 0.33
10^5	13.10 ± 2.43
10^6	120.70 ± 10.43

Chapter 10

Conclusions.

The development of the Data Processing Center for the astrometric space mission is a critical task. The success or failure of the whole mission depends on the work of the time-critical applications in the DPC. Also, the DPC must handle a huge data volume (up to 1 PB in the case of GAIA).

The DPC has to receive raw data from the SOC, to store raw and processed data and to provide an access to these data for users of the DPC (most important for the pipeline). The last requirement supposes that we have to supply a very fast access of the pipeline and applications to the database. The presented study was dedicated to the development of the concept of the DPC and, at the same time, to the decrease of the response time for applications running in the DPC.

As a result, the target of the development of the DPC was the choice of components and solutions which are able to reduce the response time.

10.1 Summary of Results and Contributions

The conceptual design of the DPC sets a model for the work with the data and a model for the development of applications. To reduce the response time we selected a simple client/server architecture (with the direct connection of the application to the database) and a multi-row data retrieval.

The logical design of the DPC includes mainly the logical design of the database. An improvement of the logical design and a migration from the logical scheme to the physical scheme of the database is made on the basis of the estimation of the time cost function. The physical design of the DPC reviews possibilities for the data storage, data processing and data mining in the DPC. The solution is based on the shared-nothing linux cluster with the relational RDBMS.

The estimation of the TCF and the use of the TCF allow to solve a number of problems like following:

- a choice of the optimum number of nodes,
- an initial data distribution for the data,

- a redistribution of the data for the optimization of the response time.

The problem of the minimization of the response time in the case of the shared-nothing cluster is a problem of the data placement. We developed a practical scheme for the data distribution. This scheme was realized in the Database Statistic System (DBSS), which monitors the state of the cluster and proposes an optimum data distribution. The test of the DBSS was made in Astronomisches Rechen-Institut with the use of the 10-node linux cluster. The test shows that even in the case of the homogeneous cluster the data must be properly distributed over the nodes of the cluster.

Finally the benchmark for the astronomical databases was proposed. The benchmark uses the most general processing chain for the astronomical data and estimates the response time from the typical requests in the case of the astronomical data processing.

10.2 Future Work

The future work can target following:

- a practical realization of the DPC for one of the astronomical/astrometrical mission,
- an adaptation of the solution based on the linear form of the TCF on the case of non-linear TCF (a number of retrieved rows $N_{rows} < 10^3$).

Indeed, in this work we used an exponential form for the description of the TCF:

$$\log t(M) = (A_0 + A_1 \exp(-A_2 \log M)) + \log M,$$

where $t(N)$ is a response time, M is a number of retrieved rows. We supposed in this study that $A_0 \gg A_1 \exp(-A_2 \log M)$. In this case $t(M) = e^{A_0} M$, where $A_0 \equiv$ const. Nevertheless it is possible to solve the matrix equation, which was used for the description of the cluster, even in the case of non-linear TCF. The solution will become iterative and we will require a careful analysis of the fitting of the problem.

10.3 Possible Use

The concept of the DPC developed in the presented study can be applied not only to the astrometric space mission. The data processing described in the study supposes the three levels of data (the raw data – the processed data – the final catalog). This scheme suits for any scanning mission which uses the CCD detector at the focal plane of the telescope. Some of such missions are already lunched (SDSS, for example) and a number of them will start in the nearest future (GAIA, the Large Synoptic Survey Telescope – `http://www.lsst.org` and others). The data processing in the astronomy becomes the very important task.

Bibliography

[Arregoces, Portolani, 2003] Arregoces M., Portolani M., Data Center Fundamentals, Cisco Press, 2003

[Arsenev & Yakovlev, 2001] Arsen'ev B.P., Yakovlev S.A., Integration of the Distributed Databases, Lan', S.-Petersburg, 2001 (in russian)

[Barndorff-Nielsen & Cox, 1997] Barndorff-Nielsen O.E., Cox D.R., Asymptotic Techniques for the Use in Statistics, Chapmann and Hall, London, 1989

[TD0284-01, 2002] Bastian U., Biermann H., Hirte S., IDAP: Interface Document for DIVA Astrometry and Pipeline, Version 1.0, TD0284-01

[TD0201-05, 2002] Bastian U., Hirte S., DIVA: Conversations and Notations for the Basic Model of the Instrument and of the Elementary Measurements, Version 1.1, TD0201-05, 2002,

[TD0251-03, 2001] Bastian U., Schilbach E., Biermann H., A Model Observation Strategy for DIVA, Version 1.2, TD0251-03

[The Benchmark Handbook, 1993] The Benchmark Handbook for Databases and Transaction Processing Systems, Second Edition, ed. Jim Gray, Morgan Kaufmann, San Francisco, 1993

[de Boer et al., 2000] de Boer K.S., Gilmore G., Høg E., Lattanzi M.G., Lindegren L, Luri X., Mignard F., de Zeeuw P.T., Perryman M.A.C., Pace O., GAIA: Composition, Formation and Evolution of the Galaxy. Report on the Concept and Technology Study, ESA-SCI(2000)4, 2000

[Booch G. et al., 1999] Booch G., Rumbaugh J., Jacobson I., The Unified Modeling Language User Guide, Addison Wesley Longman, 1999

[de Bruijne J., 2003] de Bruijne J., GAIA: Scanning Law, 2003 `http://www.rssd.esa.int/SA/GAIA/docs/info_sheets/IN_scanning_law.pdf`

[Chan, 1997] Chan Terrence, 1997, Unix System Programming Using C++, Prentice Hall PTR

[Chavez, 2000] Chavez J., Multi-tier Internet Architecture with Java, UML and OOA & D, Astronomical Data Analysis Software and Systems IX, eds. Manset N., Veillet C., Crabtree D., ASP Conf. Ser., Vol. 216, 75

[Cook et al., 1999] Cook J., Janacek C., Snow D., The DB2 Cluster Certification Guide, Prentice Hall PTR, New Jersey, 1999

[Cybersource WP, 2002] Linux vs. Windows: Total Cost of Ownership Comparison, Cybersource, 2002
(`http://www.cyber.com.au/cyber/about/linux_vs_windows_pricing_comparison.pdf`)

[DI-AED-RS-0001] DIVA Mission & System Requirements Specification, Ref.Num. DI-AED-RS-0001, Astrium, 2001

[Dluznevskaya O., 2002] Dluzhnevskaya O., World Centres for Astronomical Data, Automated Data Analysis in Astronomy, ed.: Gupta R., Singh H.P., Bailer-Jones C.A.L., New Delhi, Narosa Pub. House ,31, 2002

[Dobrovidov & Koshkin, 1997] Dobrovidov A.V., Koshkin G.M., Nonparametric Signal estimation, Physical and Mathematical Publishing Company of Russian Academy of Science, Moscow, 1997

[Egret et al., 2000] Egret D., Hanisch R.J., Murtagh F., Search and discovery tools for astronomical on-line resources and services, Astron. Astrophys. Suppl. Ser., 143, 137, 2000

[Egret D., 2001] Egret D., Astronomical Data Centers, Information Systems, and Electronic Libraries, Virtual Observatories of the Future, ASP Conference Proceedings, Vol. 225, ed.: Brunner R.J., Djorgovski G.S., Szalay A.S., San Francisco, 108, 2001

[EF5/FR/PC/038.02] GAIA System Level Technical Reassessment Study. Final Report, Ref.Num. EF5/FR/PC/038.02, Astrium, 2002

[Garofalakis & Ioannidis, 1996] Garofalakis Minos N., Ioannidis Yannis E., Multi-dimensional resource sheduling for parallel queries, In Proc. of the 1996 ACM SIGMOD Int. Conf. on Management of Data, Montreal, Canada, 365, 1996

[Garofalakis & Ioannidis, 1997] Garofalakis Minos N., Ioannidis Yannis E., Parallel Query Scheduling and Optimization with Time- and Space-Shared Resources The VLDB Journal, 296, 1997

[Garey & Johnson, 2003] Garey Michael R., Johnson David S., Computers and Intractability. A Guide to the Theory of NP-Completeness, F.H. Freeman and Company, New York, 2003

[Galeev, 2002] Galeev E.M., Optimization: Theory, Examples, Tasks, URSS, Moscow, 2002 (in russian)

[Genova et al., 2000] Genova F., Egret D., Bienaymé O., Bonnarel F., Dubois P., Fernique P., Jasniewicz G., Lesteven S., Monier R., Ochsenbein F., Wenger M., The CDS hub. Pn-line services and links at the Centre de Données astronomiques de Strasbourg, Astron. Astrophys. Suppl. Ser., 143, 1, 2000

[Grant et al., 2000] Grant Carolyn S., Accomazzi Alberto, Eichhorn Guenther, Kurtz Michael J., Murray Stephen S., The NASA Astrophysics Data System: Data holdings, Astron. Astrophys. Suppl. Ser., 143, 111, 2000

[Grauer R. et al., 2003] Graue R., Kampf D., Röumlßer S., Bastian U., Seifert W., DIVA optical telescope, Future EUV/UV and Visible Space Astrophysics Missions and Instrumentation, ed.: Blades J.S., Siegmund O. H. W., Proceedings of the SPIE, Vol. 4854, 9, 2003.

[MySQL BWP, 2003] A Guide to Lower Database TCO, MySQL Business White Paper, December 23, 2003

[Jordan, 1998] Jordan D., C++ Object Databases. Programming with ODMG Standard, Addison-Wesley, Reading, 1998

[Jordi C., 2003] Jordi C., GAIA: Astro Focal Plane, 2003 `http://www.rssd.esa.int/SA/GAIA/docs/info_sheets/IN_astro_focal_plane.pdf`

[Kharchenko et al., 2004] Kharchenko N.V., Piskunov, A.E., Scholz R.-D., Huge hight presicion stellar catalogues : Present Day and the Future, presented at the Russian Astronomical Conference - 2004, `http://www.inasan.ru/rus/rvo/vak2004/nvk_rus.ppt`

[Kharchenko et al., 1997] Kharchenko N., Rybka S., Yatsenko A., Schilbach E., Predicted star counts and mean parallaxes down to the 23td magnitude, Astron.Nachr 318, 163, 1997

[Kovalevsky, 1991] Kovalevsky J., 1991, Ap&SS, 177, 457, 1991

[Kovalevsky, 1995] Kovalevsky J., 1995, Modern Astrometry, Springer

[Kulba et al., 1999] Kul'ba V.V., Kovalevsky S.S., Kosjashenko S.A., Sirotjuk V.O., Theoretical Basis for the Design of the Optimum Structures of the Distributed Databases, SINTEG, Moscow, 1999 (in russian)

[Lasker et al., 1990] Lasker B.M., Sturch C.R., McLean B.J., Russell J.L., Jenkner H., Shara M.M., The Guide Star Catalog. I - Astronomical foundations and image processing, Astron. Journ., 99, 2019L, 1990

[van Leeuwen, 1997] van Leeuwen Floor, The HIPPARCOS Mission, Space Sci. Review, 81, 201, 1997

[Mignard & Kovalevsky, 2002] Mignard F., Kovalevsky J., Space astrometry missions: principles and objectives, Astrometry from ground and from space, ed.: Capitaine M., Stavinschi M., Bucharest, 169, 2003

[Naiburg & Maksimchuk, 2002] Naiburg E.J., Maksimchuk R.A., UML for Database Design, Addison Wesley Longman, 2002

[O'Flaherty et al., 1997] O'Flaherty K.S., Perryman M.A.C., Heger D., McDonald A.J.C., Bouffard M., Strim B. and the Hipparcos Sceince Team, The Hipparcos and Tycho Catalogues, Vol.2: The Hipparcos Satellite Operations, SP-1200, ESA Publications Division, c/o ESTEC, Noordwijk, Netherlands, 1997

[Papadimitriou & Steiglitz, 1998] Papadimitriou Christos H., Steiglitz Kenneth, Combinatorial Optimization: Algorithms and Complexity, Dover, New York, 1998

[Perry, 2002] Perry R., Maximazing the Business Value of Enterprise Database Applications on Linux Platform, IDC White Paper, IDC, 2002 (`http://www.idc.com`)

[Scheithauer S. et al., 2001] Scheithauer S., Theil S., Wiegand M., Bastian U., DIVA post-mission attitude reconstruction: the Great-Circle Reduction, Astronomische Nachrichten, vol. 322, 197, 2001

[Snevely, 2002] Snevely R., Enterprise Data Center Design and Methodology, Prentice Hall PTR, 2002

[Sukharev, 1989] Sukharev Aleksei G., Minimax Algorithms in Problems of Numerical Analysis, Nauka, Moscow, 1989 (in russian)

[Thakar et al., 2003] Thakar A.R., Szalay A.S., Kunszt P.Z., Gray J., Migrating A Multiterabyte Archive from Object to Relational Databases, Computing in Science and Engineering (CISE) 2003 , vol.5, no.5, Sep/Oct 2003, p.16

[Tomov et al., 2004] Tomov N., Dempster E., Williams M. H., Burger A., Taylor H., King P., Broughton P., Analytical Response Time Estimation in Parallel Relational Database Systems, Parallel Computing 30, ed. by G.R. Joubert, Elsevier Science BV (North-Holland), 249, 2004

[Willemsen P. G. et al., 2003] Willemsen P. G.,Bailer-Jones C. A. L., Kaempf T. A., de Boer K. S., Automated determination of stellar parameters from simulated dispersed images for DIVA, Astron. Astrophys. 401, 1203, 2003

[Zilio, 1998] Zilio Daniel C., Physical Database Design Decision Algorithms and Concurrent Reorganization for Parallel Database Systems, PhD, University of Toronto, 1998

[Zilio & Jhingran, 1994] Zilio Daniel C., Jhingran Anant, Partitioning Key Selection for a Shared-Nothing Parallel Database System, IBM Research Report RC 1980(87739), 1994

Appendix A

The choice of the programming environment. C/C++ and Java

It is important to optimally select the programming environment for the project. In fact we have to choice between C/C++ and Java languages to code the core applications for the Data Center. The main criteria for the selection will be the time required for the application coded in one of language to retrieve a number of records from the database.

A.1 Test PC

Hardware: Pentium 4 2,4 GHz PC with 512 MB RAM and 40 GB IDE disk.

Software: OS : Mandrake linux 9.1 with 2.4.21-0.13mdk core. DBMS : DB2 UDB EEE v7.2 in one-node mode. C/C++ compiler : gcc version 3.2.2 (Mandrake Linux 9.1 3.2.2-3mdk). Java compiler : gij 3.2.2.

A.2 Test database

The test database simulates the raw data table and contains the only table placed in a separate SMS tablespace with indexes in a separate SMS tablespace as well.

```
create tablespace TS_BDB managed by database
using (FILE '/home/data/bdb_sp1' 100000)@

create tablespace IS_BDB managed by database
using (FILE '/home/data/bdb_isp1' 100000)@


create table TEST_MAIN
(
    NW1         INTEGER              not null ,
    NW2         INTEGER              not null,
```

```
    T1          INTEGER                     ,
    T2          INTEGER                      ,
    CX          SMALLINT                     ,
    CY          SMALLINT                     ,
    WT          SMALLINT                     ,
    W           VARCHAR(2000),
    primary key (NW2)
) IN TS_BDB INDEX IN IS_BDB
@
```

The number of records in the table is 10^7 simulated such that the value of NW2 runs from 0 to $10^7 - 1$.

A.3 Request

The SQL statement

```
SELECT * FROM TEST_MAIN where nw2>=0 and nw2 < i
```

was used to select data into the cursor and fetch them into application's variables (i is the number of records retrieved). This statement was realized for both Java and ESQL/C. The statement was executed 20 times to measure the dispersion of the response time.

A.4 Results

Table A.1: ESQL/C and Java.

Number of records	$t^{ESQL/C}$, sec	$\sigma_t^{ESQL/C}$, sec	t^{Java}, sec	σ_t^{Java}, sec
1	$1.0\,10^{-3}$	$2.4\,10^{-6}$	0.55	$6.9\,10^{-4}$
10	$3.4\,10^{-3}$	$1.9\,10^{-5}$	0.87	$1.1\,10^{-3}$
10^2	$7.9\,10^{-3}$	$1.4\,10^{-5}$	0.63	$1.2\,10^{-2}$
10^3	$4.0\,10^{-2}$	$7.2\,10^{-6}$	0.62	$9.1\,10^{-2}$
10^4	0.36	$1.5\,10^{-5}$	1.95	0.10
10^5	3.6	$1.6\,10^{-3}$	14.8	0.07
10^6	36.4	0.47	143.9	0.36
10^7	372.6	72.3	1441.8	100.0

Results are summarized in the Table A.1 and Fig. 4.9. The retrieval of the single record with the use of Java takes 500 times more time compared with the use of C/C++. The difference decreases with the increase of the number of records retrieved and stabilizes with 10,000 records. Nevertheless the use of Java takes 4 times more time than the use of C/C++.

Appendix B

The single-row and multiple-row data retrieval

In this test the difference between massive data retrieval with the use of dynamical cursor and row by row retrieval will be shown.

B.1 Test PC

The test was made on the same PC and with the same configuration as in Test 1.

B.2 Test database

The test tables were created at the single node with data and indexes in separate DMS tablespaces.

```
create tablespace TS_BDB managed by database
using (FILE '/home/data/bdb_sp1' 200000)@

create tablespace IS_BDB managed by database
using (FILE '/home/data/bdb_isp1' 200000)@


create table TEST_MAIN
(
    NW1         INTEGER              not null ,
    NW2         INTEGER              not null,
    T1          INTEGER                      ,
    T2          INTEGER                       ,
    CX          SMALLINT                      ,
    CY          SMALLINT                      ,
    WT          SMALLINT                      ,
```

```
    W           VARCHAR(20),
    primary key (NW2)
) IN TS_BDB INDEX IN IS_BDB
@
```

The number of records in the table is 10^4 simulated so that I1 runs values from 0 to $10^4 - 1$.

B.3 Request

The SQL statement

```
SELECT * FROM TEST_MAIN where I1>=0 and I1< I_MAX
```

was used to select data into the cursor and fetch them into application's variables, where $I\_MAX \in [1, 10, 10^2, 10^3, 10^4]$. The statement was realized with the ESQL/C, for the cursor:

```
 EXEC SQL CONNECT TO :database;
EXEC SQL PREPARE c_prep FROM :sql_string;
EXEC SQL DECLARE c_base CURSOR FOR c_prep;
EXEC SQL OPEN c_base;
 i=0.0;
 for (;;) {
  EXEC SQL FETCH c_base INTO :nw1, :nw2, :t1, :t2, :cx, :cy, :wt,  :w;
  if (SQLCODE!=0) break;
  i=i+1;
 }
EXEC SQL CLOSE c_base;
EXEC SQL CONNECT RESET;
```

and for the row by row:

```
for (i=0; i< ii; i++) {
 EXEC SQL CONNECT TO :database;

 EXEC SQL SELECT * INTO :nw1, :nw2, :t1, :t2, :cx, :cy, :wt, :w
  FROM TEST_MAIN WHERE nw2= :i;

 EXEC SQL CONNECT RESET;
 }
```

Measurements for each point were made 20 times.

B.4 Results

Results are show in Table B.1 and Fig. 4.10.

Table B.1: The difference between row by row data retrieval and cursor.

Number of records	Response time, sec	
	Row by row	Cursor
1	$(1.40 \pm 0.58)\,10^{-2}$	$(0.10 \pm 0.01)\,10^{-2}$
10	$(9.99 \pm 2.12)\,10^{-2}$	$(0.26 \pm 0.59)\,10^{-2}$
10^2	$(97.75 \pm 11.31)\,10^{-2}$	$(0.60 \pm 0.58)\,10^{-2}$
10^3	9.75 ± 0.94	$(4.19 \pm 0.94)\,10^{-2}$
10^4	88.26 ± 15.69	$(37.80 \pm 1.69)\,10^{-2}$

Appendix C

The relation between the response time and the number of nodes

The goal of the test is to find a relation between the number of nodes used for the data storage and the time required to retrieve records.

C.1 Test PC

Hardware: linux cluster installed in Astronomisches-Rechen Institut, Heidelberg. The cluster consists of 10 nodes (dual Pentium 4 2,2 GHz, each with 2 73GB SCSI disks) and the front-end server (Pentium 4 2,4 GHz).

Software: OS: SuSE linux 7.2, DBMS: DB2 UDB EEE v7.2 in ten nodes configuration.

C.2 Test database

The test database simulates the raw data table and contains the only table placed in a separate DMS tablespace with indexes in a separate DMS tablespace as well. 9 nodes were used to distribute 9 tables, each uses from 1 to 9 nodes.

```
create nodegroup NG0 on nodes(0)@
create nodegroup NG01 on nodes (0,1)@
create nodegroup NG02 on nodes (0,1,2) @
create nodegroup NG03 on nodes (0,1,2,3) @
create nodegroup NG04 on nodes (0,1,2,3,4) @
create nodegroup NG05 on nodes (0,1,2,3,4,5) @
create nodegroup NG06 on nodes (0,1,2,3,4,5,6) @
create nodegroup NG07 on nodes (0,1,2,3,4,5,6,7) @
create nodegroup NG08 on nodes (0,1,2,3,4,5,6,7,8) @
create nodegroup NG09 on nodes (0,1,2,3,4,5,6,7,8,9) @
```

```
create tablespace T0_BDB IN NODEGROUP NG0
managed by database using (FILE '/work/data2/t2_t0' 512000)@

create tablespace I0_BDB IN NODEGROUP NG0
managed by database using (FILE '/work/data2/t2_i0' 512000)@

create tablespace T01_BDB IN NODEGROUP NG01
managed by database using (FILE '/work/data2/t2_t01' 512000)@

create tablespace I01_BDB IN NODEGROUP NG01
managed by database using (FILE '/work/data2/t2_i01' 512000)@

create tablespace T02_BDB IN NODEGROUP NG02
managed by database using (FILE '/work/data2/t2_t02' 512000)@

create tablespace I02_BDB IN NODEGROUP NG02
managed by database using (FILE '/work/data2/t2_i02' 512000)@

create tablespace T03_BDB IN NODEGROUP NG03
managed by database using (FILE '/work/data2/t2_t03' 512000)@

create tablespace I03_BDB IN NODEGROUP NG03
managed by database using (FILE '/work/data2/t2_i03' 512000)@

create tablespace T04_BDB IN NODEGROUP NG04
managed by database using (FILE '/work/data2/t2_t04' 512000)@

create tablespace I04_BDB IN NODEGROUP NG04
managed by database using (FILE '/work/data2/t2_i04' 512000)@

create tablespace T05_BDB IN NODEGROUP NG05
managed by database using (FILE '/work/data2/t2_t05' 512000)@

create tablespace I05_BDB IN NODEGROUP NG05
managed by database using (FILE '/work/data2/t2_i05' 512000)@

create tablespace T06_BDB IN NODEGROUP NG06
managed by database using (FILE '/work/data2/t2_t06' 512000)@

create tablespace I06_BDB IN NODEGROUP NG06
managed by database using (FILE '/work/data2/t2_i06' 512000)@

create tablespace T07_BDB IN NODEGROUP NG07
managed by database using (FILE '/work/data2/t2_t07' 512000)@

create tablespace I07_BDB IN NODEGROUP NG07
```

```
managed by database using (FILE '/work/data2/t2_i07' 512000)@

create tablespace T08_BDB IN NODEGROUP NG08
managed by database using (FILE '/work/data2/t2_t08' 512000)@

create tablespace I08_BDB IN NODEGROUP NG08
managed by database using (FILE '/work/data2/t2_i08' 512000)@

create tablespace T09_BDB IN NODEGROUP NG09
managed by database using (FILE '/work/data2/t2_t09' 512000)@

create tablespace I09_BDB IN NODEGROUP NG09
managed by database using (FILE '/work/data2/t2_i09' 512000)@

create table T0_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
    T2          INTEGER                        ,
    CX          SMALLINT                       ,
    CY          SMALLINT                       ,
    WT          SMALLINT                       ,
    W           VARCHAR(2000)
) IN T0_BDB INDEX IN I0_BDB
  partitioning key (NW2) using hashing
@
create table T01_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
    T2          INTEGER                        ,
    CX          SMALLINT                       ,
    CY          SMALLINT                       ,
    WT          SMALLINT                       ,
    W           VARCHAR(2000)
) IN T01_BDB INDEX IN I01_BDB
  partitioning key (NW2) using hashing
@

create table T02_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
```

```
    T2          INTEGER                         ,
    CX          SMALLINT                        ,
    CY          SMALLINT                        ,
    WT          SMALLINT                        ,
    W           VARCHAR(2000)
) IN T02_BDB INDEX IN I02_BDB
  partitioning key (NW2) using hashing
@
create table T03_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
    T2          INTEGER                        ,
    CX          SMALLINT                       ,
    CY          SMALLINT                       ,
    WT          SMALLINT                       ,
    W           VARCHAR(2000)
) IN T03_BDB INDEX IN I03_BDB
  partitioning key (NW2) using hashing
@
create table T04_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
    T2          INTEGER                        ,
    CX          SMALLINT                       ,
    CY          SMALLINT                       ,
    WT          SMALLINT                       ,
    W           VARCHAR(2000)
) IN T04_BDB INDEX IN I04_BDB
  partitioning key (NW2) using hashing
@
create table T05_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
    T2          INTEGER                        ,
    CX          SMALLINT                       ,
    CY          SMALLINT                       ,
    WT          SMALLINT                       ,
    W           VARCHAR(2000)
) IN T05_BDB INDEX IN I05_BDB
  partitioning key (NW2) using hashing
```

```
@
create table T06_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
    T2          INTEGER                        ,
    CX          SMALLINT                       ,
    CY          SMALLINT                       ,
    WT          SMALLINT                       ,
    W           VARCHAR(2000)
) IN T06_BDB INDEX IN I06_BDB
  partitioning key (NW2) using hashing
@
create table T07_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
    T2          INTEGER                        ,
    CX          SMALLINT                       ,
    CY          SMALLINT                       ,
    WT          SMALLINT                       ,
    W           VARCHAR(2000)
) IN T07_BDB INDEX IN I07_BDB
  partitioning key (NW2) using hashing
@
create table T08_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
    T2          INTEGER                        ,
    CX          SMALLINT                       ,
    CY          SMALLINT                       ,
    WT          SMALLINT                       ,
    W           VARCHAR(2000)
) IN T08_BDB INDEX IN I08_BDB
  partitioning key (NW2) using hashing
@
create table T09_MAIN
(
    NW1         INTEGER               not null ,
    NW2         INTEGER               not null,
    T1          INTEGER                       ,
    T2          INTEGER                        ,
```

```
    CX          SMALLINT                    ,
    CY          SMALLINT                    ,
    WT          SMALLINT                    ,
    W           VARCHAR(2000)
) IN T09_BDB INDEX IN I09_BDB
  partitioning key (NW2) using hashing
@
```

The number of records in the table is 10^7 simulated such that the value of NW2 runs from 0 to $10^7 - 1$.

C.3 Request

The SQL statement

```
SELECT * FROM TEST_MAIN where nw2>=0 and nw2 < i
```

was used to select data into the cursor and fetch them into application's variables (i is number of records).

C.4 Results

Results are summarized in the Table C.1 and Fig. 6.3. The response time decreases with the increase of the number of nodes if the number of records stored at each node exceeds some limit (100 records for this test).

Table C.1: The request time for N nodes.

Number of records	$t \pm \sigma_t$, sec for N nodes								
	1	2	3	4	5	6	7	8	9
1	$(3.3 \pm 1.7)\,10^{-2}$	$(8.9 \pm 26.6)\,10^{-2}$	$(8.1 \pm 25.1)\,10^{-2}$	$(8.7 \pm 22.8)\,10^{-2}$	$(7.2 \pm 20.7)\,10^{-2}$	$(8.5 \pm 16.5)\,10^{-2}$	$(7.8 \pm 15.7)\,10^{-2}$	$(10.2 \pm 14.4)\,10^{-2}$	$(10.4 \pm 13.0)\,10^{-2}$
10	$(10.1 \pm 0.8)\,10^{-2}$	$(7.8 \pm 3.7)\,10^{-2}$	$(8.2 \pm 4.1)\,10^{-2}$	$(5.8 \pm 3.5)\,10^{-2}$	$(5.4 \pm 4.1)\,10^{-2}$	$(8.0 \pm 5.4)\,10^{-2}$	$(16.8 \pm 6.2)\,10^{-2}$	$(9.7 \pm 6.5)\,10^{-2}$	$(11.3 \pm 5.6)\,10^{-2}$
10^2	$(56.3 \pm 0.8)\,10^{-2}$	$(34.9 \pm 6.0)\,10^{-2}$	$(31.6 \pm 3.8)\,10^{-2}$	$(25.1 \pm 3.6)\,10^{-2}$	$(17.7 \pm 4.0)\,10^{-2}$	$(23.1 \pm 7.6)\,10^{-2}$	$(26.3 \pm 8.5)\,10^{-2}$	$(31.3 \pm 8.0)\,10^{-2}$	$(23.0 \pm 5.8)\,10^{-2}$
10^3	3.2 ± 1.2	2.1 ± 0.6	2.0 ± 0.4	2.0 ± 0.3	1.9 ± 0.3	1.6 ± 0.2	1.6 ± 0.2	1.5 ± 0.2	1.4 ± 0.1
10^4	41.0 ± 9.7	31.7 ± 6.1	27.5 ± 4.3	24.3 ± 4.4	21.3 ± 3.9	20.3 ± 3.6	21.2 ± 3.7	21.8 ± 3.5	27.5 ± 3.4
10^5	330.4 ± 18.6	233.6 ± 18.3	183.3 ± 12.3	158.7 ± 10.6	137.0 ± 13.4	128.2 ± 15.8	125.3 ± 13.4	126.1 ± 12.7	135.4 ± 8.8
10^6	3093.0 ± 1409.0	1843.0 ± 617.0	1457.0 ± 406.3	1162.0 ± 161.5	1030.0 ± 131.2	962.6 ± 121.5	957.3 ± 117.6	1023.0 ± 101.2	1214.0 ± 92.6

Appendix D

The stability of the request

With this test we would like to check the stability of the response time for the request on the node which will be load with other non-database work.

D.1 Test PC

The test was produced on the one node of the linux cluster installed in Astronomisches-Rechen Institut. The description of the node configuration was done in the previous test. The cluster was shared with a number of applications using the cluster for numerical calculations (CPUs and the Myrinet network was used by external applications but not hard disks and not the Ethernet network).

D.2 Test database

The test table was created at a single node, the table simulates the raw data table and placed in a separate DMS tablespace with indexes in a separate DMS tablespace as well.

```
create nodegroup NGS on node(3)@

create tablespace TS1  managed by database
using (FILE '/home/data/bdb_st1' 100000)@

create tablespace IS_ST managed by database
using (FILE '/home/data/bdb_ist1' 100000)@


create table TEST_STAB
(
    I1          INTEGER                not null ,
) IN TS_BDB INDEX IN IS_BDB
@
```

The number of records in the table is 10^7 simulated such that the value of NW2 runs from 0 to $10^7 - 1$.

D.3 Request

The SQL statement

```
SELECT * FROM TEST_STAB where I1>=0 and i1< 1000000
```

was used to select data into the cursor and fetch them into application's variables (i is the number of records).

D.4 Results

Results for the test were collected during 15 days with a number of unknown applications running on the node. The resulting response time for the selection of 10^7 records is 133.3 ± 36.4 seconds. As non-database applications run for days (which is the normal situation for the DIVA/AMEX in case of shared cluster) we have at least 10 quasi-stable solutions for the response times.

The distribution function of the response time can be calculated as

$$f(\tau) = \sum_{i=1}^{N} A_0 \, e^{-(\tau - A_1)^2 / A_2} + A_3,$$

where τ is a response time per record. The probability to arrive at one of the quasi-stable solution is

$$P(i) = \int_{\tau_i^0 - 3\,\sigma_i}^{\tau_i^0 + 3\,\sigma_i} f(\tau) d\tau \Big/ \int_0^{\infty} f(\tau) d\tau$$

Results are shown in Tables D.1 and D.2 and Figures 7.6 and 7.7.

Table D.1: The quasi-stable solutions for the response time.

Number of solution	Period, in request cycles	request time with dispersion, sec
1	0 – 2650	95.6 ± 0.6
2	2750 – 2825	112.7 ± 3.0
3	3520 – 3580	154.2 ± 2.4
4	3635 – 3850	178.2 ± 1.9
5	4110 – 4240	192.6 ± 2.3
6	4630 – 4695	110.5 ± 2.5
7	5000 – 6000	181.3 ± 11.7
8	6200 – 6250	117.5 ± 9.5
9	7000 – 7800	152.5 ± 6.4
10	7850 – 7960	109.0 ± 1.1
11	8050 – 8350	152.9 ± 2.8
12	8610 – 8700	190.5 ± 2.3
13	9200 – 9400	109.5 ± 4.0

Table D.2: The distribution function of the response time.

Number of solution	A_0	A_1	A_2	A_3	Probability of the solution
1	1979.6	$0.96\,10^{-5}$	$0.59\,10^{-7}$	2.43	0.31
2	239.5	$1.11\,10^{-5}$	$4.04\,10^{-7}$	-0.14	0.24
3	135.0	$1.50\,10^{-5}$	$4.40\,10^{-7}$	4.52	0.16
4	118.2	$1.78\,10^{-5}$	$1.88\,10^{-7}$	35.60	0.09
5	89.5	$1.88\,10^{-5}$	$0.87\,10^{-7}$	64.38	0.05
6	299.5	$1.92\,10^{-5}$	$3.79\,10^{-7}$	-138.68	0.15

Appendix E

The form of the TCF

To check the form of the TCF two table was created at two node of the cluster. Two applications retrieved data from the table : one locally, at the same node where the table was placed and one remotely, from an other node of the cluster.

E.1 Test PC

The test was produced on two nodes of the cluster of Astronomisches-Rechen Institut. See Test C for the description.

E.2 Test database

```
create nodegroup NG0 on node(0)@

create tablespace T0 IN NODEGROUP NG0
managed by database  using (FILE '/work/data1/bdb0.t1' 512000,
FILE '/work/data1/bdb0.t2' 512000)@

create tablespace I0 IN NODEGROUP NG0
managed by database  using (FILE '/work/data1/bdb0.i1' 512000,
FILE '/work/data1/bdb0.i2' 512000)@


create table T0_MAIN
(
    NW1         INTEGER              not null ,
    NW2         INTEGER              not null,
    T1          INTEGER                      ,
    T2          INTEGER                       ,
    CX          SMALLINT                      ,
    CY          SMALLINT                      ,
```

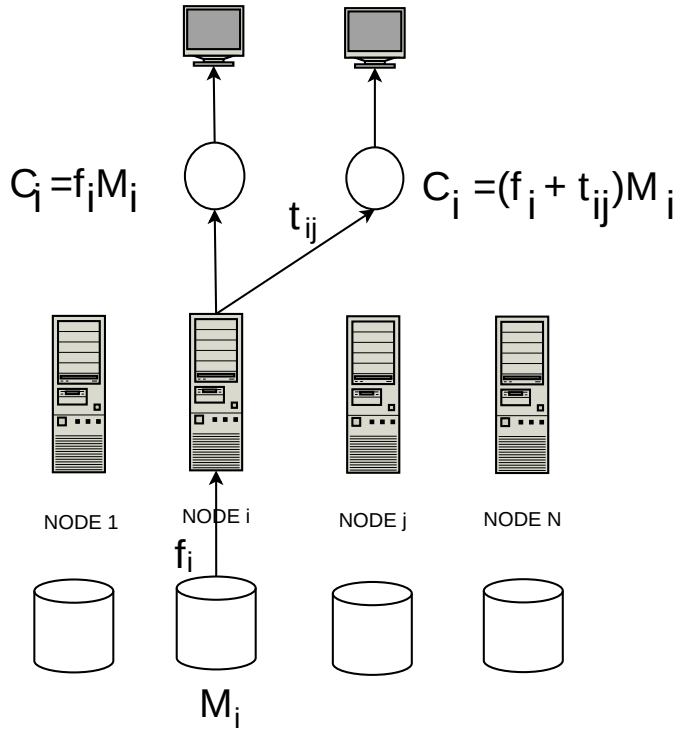

Figure E.1: The scheme of the test of the remote and local data retrieval.

.

```
    WT          SMALLINT                    ,
    W           VARCHAR(2000)
) IN TO INDEX IN IO@
```

The number of records in the table is 10^8 simulated so that I1 runs values from 0 to $10^8 - 1$.

E.3 Request

The SQL statement

```
SELECT * FROM TO_MAIN where I1>=0 and I1< I_MAX
```

was used to select data into the cursor and fetch them into application's variables, where I_MAX varies from 100 records to 7754700 records with 100 records step. Simultaneously the same statement was executed on the node 1 of the cluster.

E.4 Results

Results were collected for 46 days of the test's duration. The stable part of the results were selected (the difference between two neighbors points must be smaller than 20 %) and the TCF was fitted in the form

$$TCF(N_{rec}) = A_0^i + A_1^i \, N_{rec},$$

where i due to the local/remote data retrieval (0 for the local, 1 for the remote) and N_{rec} is a number of records.

Table E.1: The linear form of the TCF.

The start count	The end count	$A_0^1 \pm \sigma_{A_0^1}$	$A_1^1 \pm \sigma_{A_1^1}$ 10^{-5}	$A_0^2 \pm \sigma_{A_0^2}$	$A_1^2 \pm \sigma_{A_1^2}$ 10^{-5}	A_1^2/A_1^1
1518800	1533400	-429.92 ± 46.75	30.00 ± 3.06	-251.02 ± 44.74	18.14 ± 2.93	0.60
1533500	1573800	12.02 ± 4.81	1.16 ± 0.31	-39.94 ± 7.47	4.32 ± 0.48	3.71
1580000	1777900	19.47 ± 0.91	0.78 ± 0.05	-32.50 ± 1.19	3.90 ± 0.07	5.01
1902500	1914200	-9.08 ± 26.57	1.47 ± 1.39	-96.97 ± 65.90	5.87 ± 3.45	3.98
1962700	2006100	-7.84 ± 5.50	1.40 ± 0.28	-22.92 ± 8.80	2.44 ± 0.44	1.75
2006200	2398900	-0.02 ± 0.21	1.00 ± 0.01	0.74 ± 0.38	1.26 ± 0.02	1.26
2399000	2463300	-2.79 ± 3.59	1.11 ± 0.15	-15.97 ± 6.78	1.95 ± 0.28	1.75
2470700	2498300	-7.98 ± 13.78	1.32 ± 0.55	-7.82 ± 29.85	1.10 ± 1.20	0.83
2524400	2539000	-24.89 ± 22.93	1.96 ± 0.91	-38.77 ± 80.97	2.32 ± 3.20	1.18
2617700	2627900	-9.71 ± 43.08	1.35 ± 1.64	-40.86 ± 160.84	2.35 ± 6.13	1.75
2628000	2814400	-5.12 ± 0.61	1.16 ± 0.02	1.47 ± 2.00	0.73 ± 0.07	0.62
2860900	3491600	-0.16 ± 0.19	1.01 ± 0.01	-0.04 ± 0.36	1.30 ± 0.01	1.29
3491700	3827400	16.16 ± 0.52	0.55 ± 0.01	-2.45 ± 1.51	0.84 ± 0.04	1.54
3848200	3907000	-194.33 ± 8.43	6.02 ± 0.22	2.61 ± 26.75	1.22 ± 0.69	0.20
3907100	3922200	2.39 ± 181.26	1.71 ± 4.63	-1026.47 ± 146.90	27.53 ± 3.75	16.12
3925800	3940000	-3859.58 ± 277.39	99.77 ± 7.05	-1465.58 ± 359.61	38.93 ± 9.14	0.39
3979800	3993200	-181.83 ± 220.03	6.34 ± 5.52	-187.16 ± 201.11	5.48 ± 5.04	0.87
4050400	4062500	-1118.76 ± 120.50	28.58 ± 2.97	-224.41 ± 228.69	6.33 ± 5.64	0.22
4536400	4555300	5.11 ± 32.64	0.86 ± 0.72	-102.01 ± 176.44	3.06 ± 3.88	3.54
4555400	4583000	-46.80 ± 61.78	2.03 ± 1.35	-95.75 ± 93.89	2.88 ± 2.05	1.42
4656700	4691000	-428.84 ± 110.20	10.93 ± 2.36	22.52 ± 68.92	0.29 ± 1.47	0.03
4693800	4705600	-5076.67 ± 425.27	109.76 ± 9.05	-57.22 ± 355.61	1.99 ± 7.57	0.02
4706400	4717100	-5122.60 ± 522.08	110.45 ± 11.08	-305.69 ± 418.27	7.27 ± 8.88	0.07
4815600	4831700	-78.93 ± 42.88	2.65 ± 0.89	-380.79 ± 223.06	8.69 ± 4.62	3.28
4831900	4846300	-65.09 ± 54.79	2.36 ± 1.13	-5121.38 ± 625.26	106.70 ± 12.92	45.23
4877800	4890100	-504.28 ± 93.73	11.30 ± 1.92	-4595.28 ± 747.47	94.94 ± 15.30	8.40
4977300	4990700	-13.11 ± 56.91	1.24 ± 1.14	-1031.53 ± 291.89	21.89 ± 5.86	17.71
5008600	5078500	-2.20 ± 5.15	1.02 ± 0.10	-248.26 ± 24.01	6.18 ± 0.48	6.08
5082300	5116900	-16.15 ± 15.07	1.29 ± 0.30	-3.41 ± 83.59	0.84 ± 1.64	0.65
5122000	5171600	-18.69 ± 9.04	1.34 ± 0.18	0.62 ± 48.31	0.77 ± 0.94	0.58
5171700	5567100	-13.87 ± 0.75	1.24 ± 0.01	-4.43 ± 2.35	0.86 ± 0.04	0.70
5567200	5603300	-1590.28 ± 90.10	29.93 ± 1.61	-131.40 ± 93.38	3.13 ± 1.67	0.10
5685100	5714800	-435.22 ± 139.13	9.30 ± 2.44	24.89 ± 123.42	0.35 ± 2.17	0.04
5822800	5846200	-183.58 ± 63.95	4.17 ± 1.10	-39.27 ± 190.66	1.46 ± 3.27	0.35
5880400	5896800	50.62 ± 318.62	0.65 ± 5.41	-2897.68 ± 558.36	50.76 ± 9.48	78.38
5896900	5943700	21.94 ± 38.54	1.13 ± 0.65	-914.07 ± 99.10	17.27 ± 1.67	15.32
5951200	5964500	-9.42 ± 209.79	1.66 ± 3.52	-542.07 ± 303.76	10.60 ± 5.10	6.40
5986500	6002700	-595.22 ± 229.05	11.55 ± 3.82	-56.93 ± 230.10	2.88 ± 3.84	0.25
6005000	6015400	51.67 ± 748.34	0.76 ± 12.45	-20.88 ± 676.02	2.27 ± 11.25	2.97
6126700	6155500	-14.98 ± 121.29	1.30 ± 1.98	-264.98 ± 189.38	6.18 ± 3.08	4.75
6185400	6214400	-133.48 ± 124.24	3.39 ± 2.00	-1364.96 ± 146.83	23.89 ± 2.37	7.06
6240700	6264600	-226.06 ± 93.06	4.86 ± 1.49	19.73 ± 216.30	1.61 ± 3.46	0.33
6264700	6278200	-1433.58 ± 323.27	24.11 ± 5.15	-747.60 ± 489.28	13.87 ± 7.80	0.58
6278300	6294400	-285.68 ± 233.31	5.81 ± 3.71	-712.74 ± 311.17	13.29 ± 4.95	2.29
6311000	6341600	-258.31 ± 70.55	5.34 ± 1.12	82.30 ± 148.18	0.02 ± 2.34	0.00
6381600	6399000	-447.24 ± 164.28	8.41 ± 2.57	-152.63 ± 212.77	3.95 ± 3.33	0.47
6399100	6435100	17.84 ± 25.89	0.74 ± 0.40	-276.90 ± 68.88	5.88 ± 1.07	7.99
6493400	6508700	-7032.55 ± 615.06	109.70 ± 9.46	-320.82 ± 282.82	6.54 ± 4.35	0.06
6540300	6558400	-285.45 ± 246.24	6.02 ± 3.76	26.74 ± 259.31	0.87 ± 3.96	0.14
6652400	6674100	-723.64 ± 205.37	11.89 ± 3.08	-653.19 ± 171.06	11.75 ± 2.57	0.99
6753100	6770900	-1111.43 ± 151.52	17.45 ± 2.24	-876.27 ± 394.44	14.24 ± 5.83	0.82
6779800	6798300	44.75 ± 122.55	0.37 ± 1.81	-306.73 ± 316.82	5.81 ± 4.67	15.54
6820600	6840200	-89.07 ± 55.96	2.28 ± 0.82	-108.21 ± 245.28	2.85 ± 3.59	1.25
6849300	6866000	-321.99 ± 172.64	5.73 ± 2.52	-64.23 ± 428.10	1.74 ± 6.24	0.30
6936600	6949500	-3099.84 ± 553.72	45.66 ± 7.98	-1378.68 ± 653.91	20.64 ± 9.42	0.45
7197800	7212900	-219.73 ± 144.66	4.09 ± 2.01	-600.89 ± 555.33	9.14 ± 7.71	2.23
7442900	7457200	-3254.86 ± 743.41	45.11 ± 9.98	-430.03 ± 588.04	6.79 ± 7.89	0.15
7668100	7738000	-23.55 ± 15.03	1.35 ± 0.20	-506.64 ± 61.24	7.60 ± 0.79	5.64
7746300	7762900	11.72 ± 108.02	0.89 ± 1.39	-2239.45 ± 611.22	29.94 ± 7.88	33.68

Appendix F

The dependence of the TCF on the size of the table in case of the local data retrieval

To To check the form of the TCF one table was created at the single node. Two applications retrieved data from the table : one locally, at the same node where the table was placed and one remotely, from the other node of the cluster.

F.1 Test PC

The test was made on the same PC and with the same configuration as in Test 1.

F.2 Test database

The test tables were created at the single node with data and indexes in separate DMS tablespaces.

```
create tablespace T1_BDB managed by database
using (FILE '/home/data/bdb_t1' 50000)@

create tablespace I1_BDB managed by database
using (FILE '/home/data/bdb_it1' 50000)@


create table T1
(
    I1          INTEGER                not null ,
    primary key (I1)
) IN T1_BDB INDEX IN I1_BDB
@
```

```
create tablespace T2_BDB managed by database
using (FILE '/home/data/bdb_t1' 50000)@

create tablespace I2_BDB managed by database
using (FILE '/home/data/bdb_it1' 50000)@


create table T2
(
    I1          INTEGER               not null ,
    I2          INTEGER                        ,
    primary key (I1)
) IN T2_BDB INDEX IN I2_BDB
@

create tablespace T3_BDB managed by database
using (FILE '/home/data/bdb_t1' 50000)@

create tablespace I3_BDB managed by database
using (FILE '/home/data/bdb_it1' 50000)@


create table T3
(
    I1          INTEGER               not null ,
    S1          SMALLINT                       ,
    D1          DOUBLE                         ,
    primary key (I1)
) IN T3_BDB INDEX IN I3_BDB
@

create tablespace T4_BDB managed by database
using (FILE '/home/data/bdb_t1' 50000)@

create tablespace I4_BDB managed by database
using (FILE '/home/data/bdb_it1' 50000)@


create table T4
(
    I1          INTEGER               not null ,
    D1          DOUBLE                         ,
    W           CHAR(8)                        ,
    primary key (I1)
) IN T4_BDB INDEX IN I4_BDB
```

```
@
create tablespace T5_BDB managed by database
using (FILE '/home/data/bdb_t1' 50000)@

create tablespace I5_BDB managed by database
using (FILE '/home/data/bdb_it1' 50000)@


create table T5
(
    I1          INTEGER               not null ,
    D1          DOUBLE                         ,
    W           VARCHAR(8)                        ,
    primary key (I1)
) IN T5_BDB INDEX IN I5_BDB
@
```

The number of records in the table is 10^5 simulated so that I1 runs values from 0 to $10^5 - 1$.

F.3 Request

The SQL statement

```
SELECT * FROM TEST_STAB where I1>=0 and I1< I_MAX
```

was used to select data into the cursor and fetch them into application's variables, where $I\_MAX \in [1, 10, 10^2, 10^3, 10^4, 10^5]$.

F.4 Results

The TCF was fitted in the form

$$TCF(N_{rec}, size(Table)) = A_0 + A_1\, size(Table),$$

where N_{rec} is a number of records and $size(Table)$ is a size of the table in B for [T1, T2, T3, T4] or [T1, T2, T3, T5] .

Table F.1: The dependence of the response time on the size of the table.

Number of records	Table				
	T1 4B	T2 8B	T3 14B	T4 20B	T5 20B(VARCHAR)
1	$(9.2 \pm 7.8)\,10^{-2}$	$(7.5 \pm 0.2)\,10^{-2}$	$(7.9 \pm 0.5)\,10^{-2}$	$(8.0 \pm 0.7)\,10^{-2}$	$(7.9 \pm 0.6)\,10^{-2}$
10	$(7.4 \pm 0.2)\,10^{-2}$	$(7.5 \pm 0.2)\,10^{-2}$	$(7.9 \pm 0.2)\,10^{-2}$	$(8.0 \pm 0.2)\,10^{-2}$	$(7.9 \pm 0.2)\,10^{-2}$
10^2	$(7.7 \pm 0.2)\,10^{-2}$	$(7.8 \pm 0.2)\,10^{-2}$	$(8.1 \pm 0.2)\,10^{-2}$	$(8.2 \pm 0.2)\,10^{-2}$	$(8.1 \pm 0.2)\,10^{-2}$
10^3	$(10.2 \pm 0.2)\,10^{-2}$	$(10.5 \pm 0.2)\,10^{-2}$	$(11.0 \pm 0.2)\,10^{-2}$	$(11.3 \pm 0.2)\,10^{-2}$	$(11.0 \pm 0.2)\,10^{-2}$
10^4	$(36.1 \pm 0.4)\,10^{-2}$	$(37.2 \pm 0.2)\,10^{-2}$	$(39.6 \pm 0.3)\,10^{-2}$	$(41.2 \pm 0.3)\,10^{-2}$	$(40.0 \pm 0.1)\,10^{-2}$
10^5	2.95 ± 0.01	3.04 ± 0.03	3.16 ± 0.05	3.37 ± 0.01	3.45 ± 0.09

Table F.2: The dependence of the TCF on the tables' size.

Number of records	with T4		with T5	
	$A_0 \pm \sigma_{A_0}$	$A_1 \pm \sigma_{A_1}$	$A_0 \pm \sigma_{A_0}$	$A_1 \pm \sigma_{A_1}$
1	$(8.67 \pm 0.09)\,10^{-2}$	$(-0.05 \pm 0.07)\,10^{-2}$	$(8.72 \pm 0.82)\,10^{-2}$	$(-0.05 \pm 0.06)\,10^{-2}$
10	$(7.24 \pm 0.09)\,10^{-2}$	$(0.04 \pm 0.01)\,10^{-2}$	$(7.28 \pm 0.12)\,10^{-2}$	$(0.03 \pm 0.01)\,10^{-2}$
10^2	$(7.48 \pm 0.07)\,10^{-2}$	$(0.04 \pm 0.01)\,10^{-2}$	$(7.53 \pm 0.12)\,10^{-2}$	$(0.03 \pm 0.01)\,10^{-2}$
10^3	$(9.99 \pm 0.10)\,10^{-2}$	$(0.07 \pm 0.01)\,10^{-2}$	$(10.11 \pm 0.21)\,10^{-2}$	$(0.05 \pm 0.02)\,10^{-2}$
10^4	$(34.78 \pm 0.22)\,10^{-2}$	$(0.33 \pm 0.02)\,10^{-2}$	$(35.29 \pm 0.65)\,10^{-2}$	$(0.26 \pm 0.05)\,10^{-2}$
10^5	2.83 ± 0.03	$(2.57 \pm 0.25)\,10^{-2}$	2.80 ± 0.06	$(3.06 \pm 0.50)\,10^{-2}$

Appendix G

The dependence of the TCF on the size of the table in case of the remote data retrieval

To check the form of the TCF one table was created at the single node.

G.1 Test PC

The test was produced on the one node of the linux cluster installed in Astronomisches-Rechen Institut. The description of the node configuration was done in the Test C.

G.2 Test database

The test tables were created at the single node with data and indexes in separate DMS tablespaces.

```
create nodegroup NG09 on nodes (9) @

create tablespace T09_BDB IN NODEGROUP NG09
managed by database using (FILE '/work/data2/t09.t1' 512000,
FILE '/work/data2/t09.t2' 512000,FILE '/work/data2/t09.t3' 512000)@

create tablespace I09_BDB IN NODEGROUP NG09
managed by database using (FILE '/work/data2/t09.i1' 512000,
FILE '/work/data2/t09.i2' 512000,FILE '/work/data2/t09.i3' 512000)@

create table T1
(
    I1          INTEGER
) IN T09_BDB INDEX IN I09_BDB
```

```
@

create index IN_T1 on T1 (I1 ASC)@

create table T2
(
    I1          INTEGER,
    I2          INTEGER
) IN T09_BDB INDEX IN I09_BDB
@

create index IN_T2 on T2 (I1 ASC)@

create table T3
(
    I1          INTEGER,
    I2          INTEGER,
    D1          DOUBLE

) IN T09_BDB INDEX IN I09_BDB
@

create index IN_T3 on T3 (I1 ASC)@

create table T4
(
    I1          INTEGER,
    I2          INTEGER,
    D1          DOUBLE,
    D2          DOUBLE,
    D3          DOUBLE,
    D4          DOUBLE
) IN T09_BDB INDEX IN I09_BDB
@

create index IN_T4 on T4 (I1 ASC)@

create table T5
(
    I1          INTEGER,
    I2          INTEGER,
    D1          DOUBLE,
    D2          DOUBLE,
    D3          DOUBLE,
    D4          DOUBLE,
    C1          CHAR(100)
```

```
) IN T09_BDB INDEX IN I09_BDB
@

create index IN_T5 on T5 (I1 ASC)@

create table T6
(
    I1          INTEGER,
    I2          INTEGER,
    D1          DOUBLE,
    D2          DOUBLE,
    D3          DOUBLE,
    D4          DOUBLE,
    C1          VARCHAR(100)
) IN T09_BDB INDEX IN I09_BDB
@

create index IN_T6 on T6 (I1 ASC)@

create table T7
(
    I1          INTEGER,
    I2          INTEGER,
    D1          DOUBLE,
    D2          DOUBLE,
    D3          DOUBLE,
    D4          DOUBLE,
    C1          VARCHAR(500)
) IN T09_BDB INDEX IN I09_BDB
@

create index IN_T7 on T7 (I1 ASC)@

create table T8
(
    I1          INTEGER,
    I2          INTEGER,
    D1          DOUBLE,
    D2          DOUBLE,
    D3          DOUBLE,
    D4          DOUBLE,
    C1          VARCHAR(1000)
) IN T09_BDB INDEX IN I09_BDB
@

create index IN_T8 on T8 (I1 ASC)@
```

Table G.1: The dependence of the TCF on the tables' size.

Number of records	coefficients of the TCF	
	$A_0 \pm \sigma_{A_0}$	$A_1 \pm \sigma_{A_1}$
1	$(2.80 \pm 0.35)\,10^{-2}$	$(0.2 \pm 0.8)\,10^{-5}$
10	$(8.70 \pm 1.73)\,10^{-2}$	$(-6.2 \pm 3.9)\,10^{-5}$
10^2	$(51.32 \pm 11.22)\,10^{-2}$	$(-25.7 \pm 25.1)\,10^{-5}$
10^3	2.94 ± 0.12	$(13.4 \pm 25.8)\,10^{-5}$
10^4	26.09 ± 0.23	$(52.5 \pm 50.6)\,10^{-5}$
10^5	254.14 ± 1.26	$(1.03 \pm 0.28)\,10^{-2}$
10^6	2546.17 ± 19.48	$(43.40 \pm 4.37)\,10^{-2}$

The number of records in the table is 10^5 simulated so that I1 runs values from 0 to $10^6 - 1$.

G.3 Request

The SQL statement

```
SELECT * FROM TEST_STAB where I1>=0 and I1< I_MAX
```

was used to select data into the cursor and fetch them into application's variables, where $I\_MAX \in [1, 10, 10^2, 10^3, 10^4, 10^5, 10^6]$.

G.4 Results

The TCF was fitted in the form

$$TCF(N_{rec}, size(Table)) = A_0 + A_1\, size(Table),$$

where N_{rec} is a number of records and $size(Table)$ is a size of the table in B for [T1, T2, T3, T4, T5, T7, T8].

Table G.2: The response time for tables of various sizes.

Number of records	Table							
	T1 4B	T2 8B	T3 16B	T4 40B	T5 140B	T6 140B (VARCHAR)	T7 540B (VARCHAR)	T8 1040B (VARCHAR)
1	$(3.66 \pm 0.02)\ 10^{-2}$	$(3.23 \pm 0.40)\ 10^{-2}$	$(2.83 \pm 1.11)\ 10^{-2}$	$(1.98 \pm 1.00)\ 10^{-2}$	$(2.77 \pm 0.98)\ 10^{-2}$	$(0.79 \pm 0.31)\ 10^{-2}$	$(1.91 \pm 1.24)\ 10^{-2}$	$(3.54 \pm 1.10)\ 10^{-2}$
10	$(12.66 \pm 0.01)\ 10^{-2}$	$(11.87 \pm 0.61)\ 10^{-2}$	$(9.83 \pm 3.11)\ 10^{-2}$	$(5.31 \pm 3.54)\ 10^{-2}$	$(2.85 \pm 1.38)\ 10^{-2}$	$(2.36 \pm 1.07)\ 10^{-2}$	$(3.14 \pm 1.73)\ 10^{-2}$	$(4.07 \pm 2.00)\ 10^{-2}$
10^2	$(90.30 \pm 14.20)\ 10^{-2}$	$(60.10 \pm 29.08)\ 10^{-2}$	$(32.82 \pm 15.83)\ 10^{-2}$	$(49.86 \pm 43.27)\ 10^{-2}$	$(22.05 \pm 90.21)\ 10^{-2}$	$(29.05 \pm 12.80)\ 10^{-2}$	$(21.99 \pm 0.94)\ 10^{-2}$	$(36.10 \pm 0.88)\ 10^{-2}$
10^3	3.38 ± 1.09	2.82 ± 0.87	2.94 ± 0.57	2.62 ± 0.55	2.95 ± 0.58	2.69 ± 0.52	3.05 ± 0.46	3.07 ± 0.52
10^4	26.01 ± 3.14	26.54 ± 4.18	26.17 ± 3.66	25.32 ± 3.93	26.26 ± 2.59	25.30 ± 3.36	26.91 ± 4.36	26.37 ± 5.65
10^5	253.10 ± 9.31	253.10 ± 5.00	256.90 ± 6.31	255.00 ± 6.46	257.30 ± 4.17	255.30 ± 5.73	255.20 ± 6.67	267.00 ± 6.18
10^6	2568.00 ± 27.75	2552.00 ± 46.74	2567.00 ± 25.01	2564.00 ± 23.45	2611.00 ± 94.08	2564.00 ± 33.68	2699.00 ± 244.9	3039.00 ± 81.12

Appendix H

The stability of the F+T matrix

The **F+T** matrix must be calculated for the number of records which will be retrieved in the real application which will use data partitioning. In this test the dependence of the **F+T** on the number of records used in the test query will be studied.

H.1 Test PC

The test was produced on two nodes of the cluster of Astronomisches-Rechen Institut. See Test C for the description. Only 3 nodes of the cluster were used.

H.2 Test database

The benchmark database described in the Chapter 6 was used for the test.

H.3 Request

Two SQL statements were executed consequently:

```
SELECT * FROM TEST_MAIN where I1>=0 and I1< I_MAX

SELECT * FROM TEST_BAD where I1>=0 and I1< I_MAX
```

where $I\_MAX \in [1, 10, 10^2, 10^3, 10^4, 10^5]$. Measurements for each point were made 20 times.

H.4 Results

Results in Table H.1 and in Fig. 7.3 shows the stability of the **F+T** matrix from some number of records retrieved.

Table H.1: The **F**+**T** matrix with dispersion.

Number of records	The **F**+**T** matrix, sec, for element $(f+t)^i_j$		
	00	01	02
1	$(1.302 \pm 2.284)\,10^{-3}$	$(8.523 \pm 10.390)\,10^{-2}$	(1.402 ± 4.201)
10	$(1.201 \pm 0.624)\,10^{-4}$	$(1.736 \pm 0.994)\,10^{-4}$	$(1.911 \pm 1.311)\,10^{-2}$
10^2	$(5.228 \pm 0.721)\,10^{-5}$	$(1.077 \pm 0.110)\,10^{-2}$	$(1.088 \pm 0.124)\,10^{-2}$
10^3	$(4.409 \pm 0.095)\,10^{-5}$	$(1.010 \pm 0.012)\,10^{-2}$	$(1.011 \pm 0.001)\,10^{-2}$
10^4	$(4.278 \pm 0.200)\,10^{-5}$	$(1.040 \pm 0.083)\,10^{-2}$	$(1.027 \pm 0.005)\,10^{-2}$
10^5	$(4.113 \pm 0.006)\,10^{-5}$	$(1.017 \pm 0.009)\,10^{-2}$	$(1.015 \pm 0.005)\,10^{-2}$
10^6	$(4.088 \pm 0.016)\,10^{-5}$	$(1.018 \pm 0.001)\,10^{-2}$	$(1.018 \pm 0.001)\,10^{-2}$
	10	11	12
1	$(6.791 \pm 5.906)\,10^{-2}$	$(2.445 \pm 5.030)\,10^{-2}$	$(9.693 \pm 13.600)\,10^{-2}$
10	$(1.589 \pm 0.586)\,10^{-2}$	$(2.476 \pm 5.021)\,10^{-3}$	$(1.939 \pm 1.336)\,10^{-2}$
10^2	$(1.061 \pm 0.055)\,10^{-2}$	$(1.784 \pm 3.622)\,10^{-4}$	$(1.087 \pm 0.139)\,10^{-2}$
10^3	$(1.006 \pm 0.005)\,10^{-2}$	$(5.661 \pm 3.678)\,10^{-5}$	$(1.151 \pm 0.321)\,10^{-2}$
10^4	$(1.002 \pm 0.001)\,10^{-2}$	$(4.047 \pm 0.283)\,10^{-5}$	$(1.019 \pm 0.039)\,10^{-2}$
10^5	$(1.019 \pm 0.009)\,10^{-2}$	$(4.057 \pm 0.036)\,10^{-5}$	$(1.019 \pm 0.004)\,10^{-2}$
10^6	$(1.047 \pm 0.018)\,10^{-2}$	$(3.832 \pm 0.017)\,10^{-5}$	$(1.059 \pm 0.001)\,10^{-2}$
	20	21	22
1	$(6.795 \pm 6.260)\,10^{-2}$	$(6.794 \pm 6.263)\,10^{-2}$	$(5.702 \pm 0.560)\,10^{-4}$
10	$(1.579 \pm 0.627)\,10^{-2}$	$(1.578 \pm 0.627)\,10^{-2}$	$(2.476 \pm 5.323)\,10^{-3}$
10^2	$(1.058 \pm 0.060)\,10^{-2}$	$(1.066 \pm 0.081)\,10^{-2}$	$(2.809 \pm 5.139)\,10^{-4}$
10^3	$(1.008 \pm 0.011)\,10^{-2}$	$(1.008 \pm 0.008)\,10^{-2}$	$(6.652 \pm 5.227)\,10^{-5}$
10^4	$(1.003 \pm 0.002)\,10^{-2}$	$(1.019 \pm 0.037)\,10^{-2}$	$(4.174 \pm 0.593)\,10^{-5}$
10^5	$(1.019 \pm 0.007)\,10^{-2}$	$(1.018 \pm 0.004)\,10^{-2}$	$(4.234 \pm 0.048)\,10^{-5}$
10^6	$(1.014 \pm 0.002)\,10^{-2}$	$(1.017 \pm 0.004)\,10^{-2}$	$(5.486 \pm 0.003)\,10^{-5}$

Appendix I

The choice of the data upload strategy. The direct insert and the use of DB2 utility

The presented test shows the difference between the use of the INSERT statement for the data load and the use of the DB2 utility db2split and `LOAD FROM FILE` statement. In the last case we use a preformatted (by db2split) database page to insert it directly in the database's tablespace.

I.1 Test PC

Hardware: Pentium 4 2,4 GHz PC with 512 MB RAM and 40 GB IDE disk.

Software: OS : Mandrake linux 9.1 with 2.4.21-0.13mdk core. DBMS : DB2 UDB EEE v7.2 in one-node mode. C/C++ compiler : gcc version 3.2.2 (Mandrake Linux 9.1 3.2.2-3mdk). Java compiler : gij 3.2.2.

I.2 Test database

The test database simulates the raw data table and contains the only table placed in a separate SMS tablespace with indexes in a separate SMS tablespace as well.

```
create tablespace TSI_BDB managed by database
using (FILE '/home/data/bdbi_sp' 200000)@

create tablespace ISI_BDB managed by database
using (FILE '/home/data/bdbi_isp' 200000)@


create table TEST
(
```

```
    I1          INTEGER
) IN TSI_BDB INDEX IN ISI_BDB
@

create index IND1 on TEST (I1 ASC) @
```

The number of records in the table is 10^6 simulated such that the value of I1 runs from 0 to $10^6 - 1$.

I.3 Request

The SQL statement for the direct insert is

```
INSERT INTO TEST VALUES (:load_val).
```

The ESQL/C program reads the `:load_val` from the file with data. In the second case the csv file (DEL format in terms of DB2) is processed by db2split utility and loaded by the `LOAD FROM <filename> OF DEL INSERT INTO TEST` command.

I.4 Results

Table I.1: INSERT and LOAD.

Number of records	$t^{INSERT} \pm \sigma_t^{INSERT}$, sec	$t^{LOAD} \pm \sigma_t^{LOAD}$, sec
1	$(5.78 \pm 0.20)\,10^{-4}$	$(4.27 \pm 2.91)\,10^{-1}$
10	$(1.87 \pm 2.98)\,10^{-3}$	$(4.14 \pm 2.35)\,10^{-1}$
10^2	$(6.74 \pm 0.50)\,10^{-3}$	$(4.16 \pm 2.29)\,10^{-1}$
10^3	$(9.81 \pm 1.32)\,10^{-2}$	$(5.53 \pm 2.55)\,10^{-1}$
10^4	1.07 ± 0.02	$(7.96 \pm 3.00)\,10^{-1}$
10^5	10.84 ± 0.10	2.72 ± 0.24
10^6	107.80 ± 0.53	14.52 ± 7.46

Results are summarized in the Table I.1 and Fig. 6.4. The use of the db2split utility and `LOAD` statement decreases the time required to upload the data, but in case of the massive data load only (more than 10^4 records).