The South African Institute for Computer Scientists and Information Technologists

ANNUAL RESEARCH AND DEVELOPMENT SYMPOSIUM

23-24 NOVEMBER 1998
CAPE TOWN
Van Riebeeck hotel in Gordons Bay

Hosted by the University of Cape Town in association with the CSSA, Potchefstroom University for CHE and The University of Natal

PROCEEDINGS

EDITED BY
D. PETKOV AND L. VENTER

SPONSORED BY

ABSA Group

The South African Institute for Computer Scientists and Information Technologists

ANNUAL RESEARCH AND DEVELOPMENT SYMPOSIUM

23-24 NOVEMBER 1998
CAPE TOWN
Van Riebeeck hotel in Gordons Bay

Hosted by the University of Cape Town in association with the CSSA,
Potchefstroom University for CHE and
The University of Natal

GENERAL CHAIR : PROF G. HATTINGH, PU CHE

PROGRAMME CO-CHAIRS:
PROF. L VENTER, PU CHE (Vaal Triangle), PROF. D. PETKOV, UN-PMB

LOCAL ORGANISING CHAIR: PROF. P. LICKER, UCT - IS

PROCEEDINGS

EDITED BY
D. PETKOV AND L. VENTER

SYMPOSIUM THEME:

Development of a quality academic CS/IS infrastraucture in South Africa

SPONSORED BY

ABSA Group

Proceedings of the 1998 Annual Research Conference of the South African Institute for Computer Scientists and Information Technologists.
Edited by Prof. D. Petkov and Prof. L. Venter
Van Reebeck Hotel, Gordons Bay, 23-24 November 1998

ISBN: 1-86840-303-3

Keywords: Computer Science, Information Systems, Software Engineering.

Office of SAICSIT: Prof. J.M.Hatting, Department of Computer Science and information Systems, Potchefstroom University for CHE, Private Bag X6001, Potchefstroom, 2520, RSA.

Produced by the Library Copy Centre, University of Natal, Pietermaritzburg.

FOREWORD

The South African Institute for Computer Scientists and Information Technologists (SAICSIT) promotes the cooperation of academics and industry in the area of research and development in Computer Science, Information Systems and Technology and Software Engineering. The culmination of its activities throughout the year is the annual research symposium. This book is a collection of papers presented at the 1998 such event taking place on the 23rd and 24th of November in Gordons Bay, Cape Town. The Conference is hosted by the Department of Information Systems, University of Cape Town in cooperation with the Department of Computer Science, Potchefstroom University for CHE and and Department of Computer Science and Information Systems of the University of Natal, Pietermaritzburg.

There are a total of 46 papers. The speakers represent practitioners and academics from all the major Universities and Technikons in the country. The number of industry based authors has increased compared to previous years.

We would like to express our gratitude to the referees and the paper contributors for their hard work on the papers included in this volume. The Organising and Programme Committees would like to thank the keynote speaker, Prof M.C.Jackson, Dean, University of Lincolshire and Humberside, United Kingdom, President of the International Federation for Systems Research as well as the Computer Society of South Africa and The University of Cape Town for the cooperation as well as the management and staff of the Potchefstroom University for CHE and the University of Natal for their support and for making this event a success.

Giel Hattingh, Paul Licker, Lucas Venter and Don Petkov

Table of Contents Page

QUALITY CONSIDERATIONS OF REAL TIME ACCESS TO MULTIDIMENSIONAL MATRICES

F. Janssen
Rubico Engineering Server Products,
Postnet #22, Private Bag X87, Bryanston, 2021

Abstract

This paper investigates the quality considerations of real time access to multidimensional matrices. Researchers have emphasised the relationship between multidimensional matrices and data warehousing in the past; the multidimensional matrix concept may also be used to store real time transaction data. Different models can be used to implement multidimensional matrices. The application of the quality characteristics of Functionality, Reliability, Usability, Maintainability, Portability and Efficiency, as related to software developed for the Matrix Model is discussed. Efficient software must provide appropriate responses. Performance is an inherent problem when accessing multidimensional matrices. Different techniques can be used to optimise performance. Techniques such as aggregates and partitioning can provide outstanding performance gains. The importance of quality is often recognised more by it's absence at the end of a software project, than by its presence at the start of a new project. Quality cannot be added into a product after it has been developed, it must be built into it from the start.

1. Introduction

This paper investigates the quality considerations of real time access to multidimensional matrices. Researchers have emphasised the relationship between multidimensional matrices and data warehousing in the past; the multidimensional matrix concept may also be used to store real time transaction data. The main problem of real time access to multidimensional matrices is the slow response time of data access.

Section 1 contains a broad overview of multidimensional matrices. This section describes the Typical Dimensional Model, which is mostly used for data warehousing, and the Matrix Model, which may be used for both data warehousing and real time access. The Typical Dimensional Model contains a single fact table and a number of dimension tables. The Matrix Model consist of a fact table; one combined dimension table, which has relationships to a central repository; and a third table, where date related data is stored. The central repository takes advantage of leading-edge data repository concepts and technologies.

Section 2 examines the quality characteristics of Functionality, Reliability, Usability, Maintainability, Portability and Efficiency. The relevance of these characteristics to real time access will be identified.

The application of these six quality characteristics, as related to software developed for the Matrix Model is discussed in Section 3.

Section 4 briefly discusses the different techniques that can be used to optimise performance. Performance considerations correlate closely to the physical data model. Intelligent data modelling, through techniques such as aggregates and partitioning can provide outstanding performance gains and is covered in this last section.

2. Multidimensional Matrices concepts

Multidimensional Matrices is a new name for an old technique of making databases simple and understandable. Multidimensional matrices allow you to visualise the database as a "cube" of dimensions.

The different dimensions are intimately related and can be stored, viewed and analysed from different perspectives.

In Figure 1 there are three dimensions: Stores, Products and Time. We now have a 3 by 3 by 3 multidimensional matrix containing 27 cells. The Sales Volume figures are located at the intersections of the dimension positions.

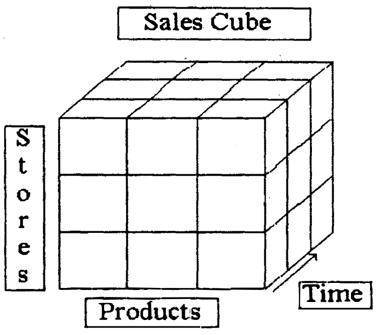

Figure 1: Sales Cube

Different models can be used to present multidimensional matrices in a database. The most common model used is the Typical Dimensional Model as illustrated in Figure 2 [2].

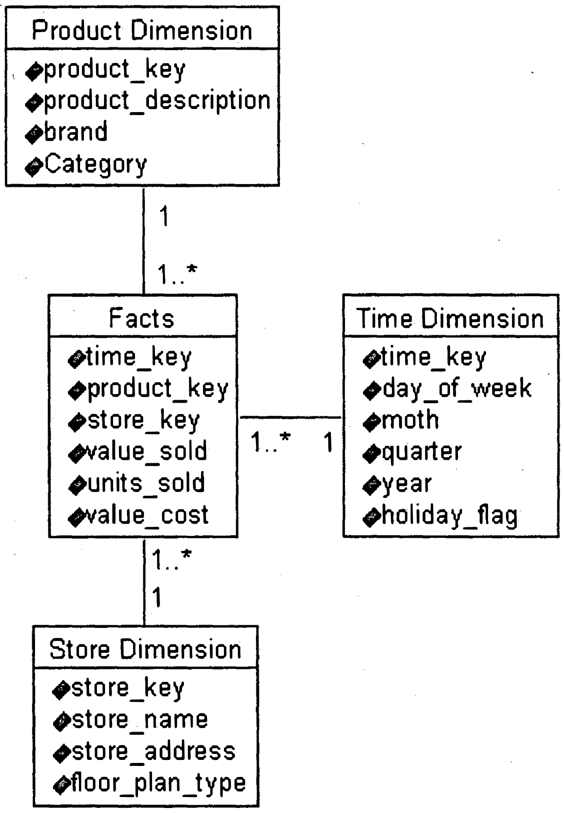

Figure 2: Typical Dimensional Model

The Typical Dimensional Model may also be referred to as the star join schema, because the model looks like a star. There is one large central fact table and a few smaller dimension tables. A dimension acts as an index for identifying values within the fact table. Every combination of dimension tables generates a record in the fact table; for example a particular product sold in a specific store, at a certain time, will generate one record in the fact table. The fact table, depicted above, may easily contain billions of records.

The fact table is used to store numerical measurement data. Each measurement is taken at the intersection of all three dimensions. In Figure 2 the measurement is the value sold, units sold and the cost price of the sold units. Relatively few dimension combinations usually exist in the fact table. In the above example it is very unlikely that all the products will be sold in all the different stores on the same date and time.

Dimension tables are used to store descriptions of the different dimension combinations. In the store dimension every store has a name, address and floor plan. By modifying the database table, other attributes such as manager and telephone number could be added. Dimension table attributes usually serve as constraints or row headers in queries.

The Typical Dimensional model is generally used for data warehousing but the enthusiasm for data warehousing is very dangerous. It is often assumed that a warehouse will solve all the problems of an organisation, when in reality, attention to operational systems might solve many problems. Building a data warehouse often only provides a "Band-Aid" solution for problems in operational systems. By using a suitable design, operational data may provide the same functionality as the data in a data warehouse, with the added advantage of real time access [1].

The Matrix Model can be used for both real time access, for example, access to product sales prices; and data warehousing, for example, storing sales history. This model consist of a fact table, one combined dimension table (which has relationships to a central repository) and a third table where date related data is stored. The central repository has a threefold purpose of providing:

- Structures as a grouping mechanism for items as illustrated in Figure 3. Structures may consist either of single groups of items, or of hierarchically related groups of items.

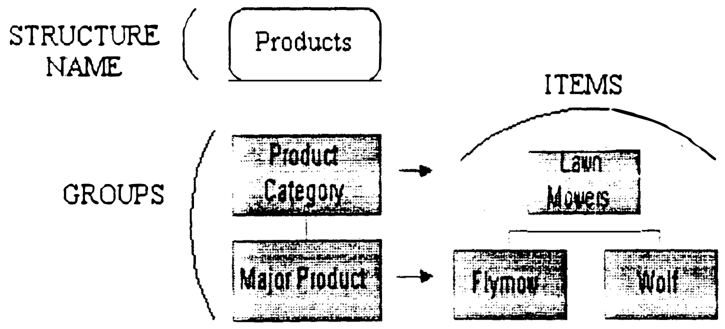

Figure 3: Structure

- A Virtual Database for holding attributes and their associated values as illustrated in Figure 4. An attribute can be string, numeric, boolean or a date.

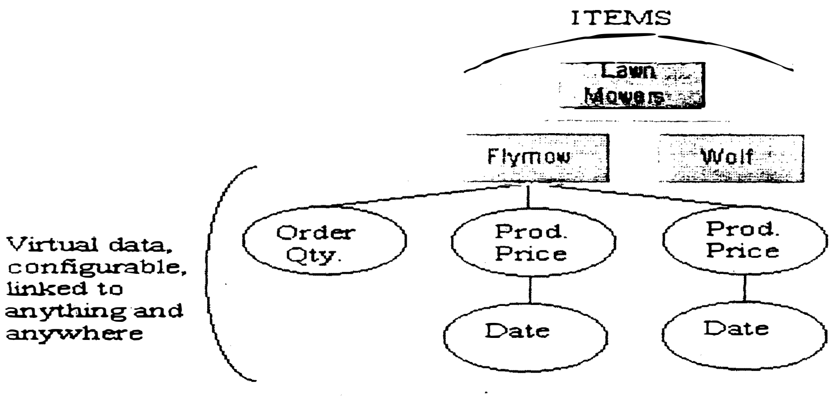

Figure 4: Virtual Database

- Configuration Capability to allow dynamic configuration of data requirements as per individual user's business rules.

Instead of separate dimension tables the Matrix Model, as illustrated in Figure 5, use one combined dimension table. The combined dimension table is used to group structure items together. The dimension table has a unique technical key (t_stvlnk). This key is a numeric attribute used as the primary key of the table.

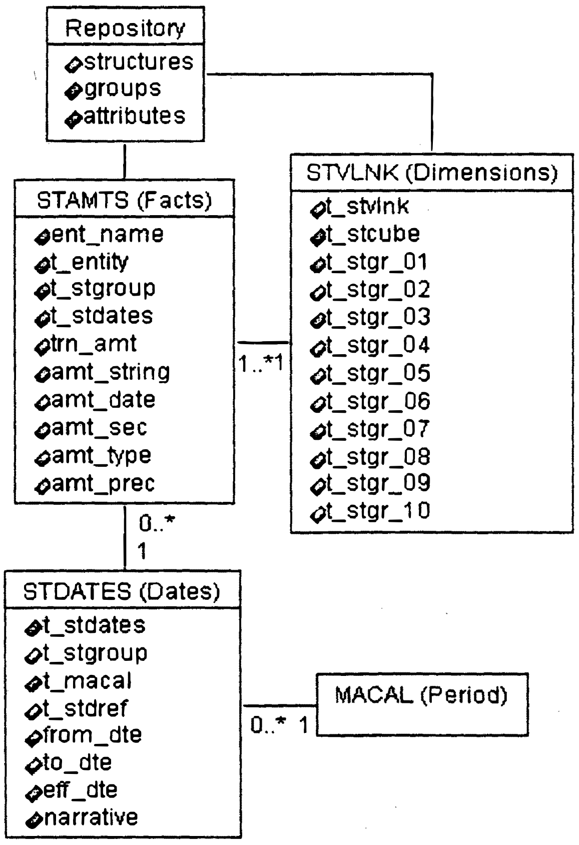

Figure 5: The Matrix Model

Different structure groupings or cubes can exist in the dimension table, for example one cube can consist of a product and store grouping while another can be a resource and location grouping. The dimension table is denormalized to improve performance; therefore the maximum number of structures that may by grouped together, as a cube, is ten. Each cube must be configured in the repository. A unique technical key is generated for every configured cube. This key is used to distinguish between the different cubes in the dimension table (t_stcube).

The combination of repository structure items that makes up an occurrence in the dimension table is referred to as a structure matrix. The technical keys of these items are stored in the t_stgr_*nn* columns. The number of the column relates to a specific structure in the grouping.

For each record in the dimension table one or more records can exist in the fact table. The fact table is used to store attributes for a specific structure matrix. The fact table has a combined technical key (ent_name, t_entity, t_stgroup, t_stdates). The ent_name is the name of the table to which the record is related, in this model it will be "STVLNK". The next field contains the primary key of the dimension table record that is linked to that specific, fact table record. Every, attribute in the central repository for example Sales Volume has a unique key (t_stgroup) and type. Valid types are string, date, numeric or boolean. Each record in the fact table is linked to a specific date (t_stdates) record in the date table.

The value of the attribute is stored in trn_amt if it is numeric, in amt_string if it is a string or boolean and in amt_date if it is a date. The attribute type is stored in the amt_type field. If it is a numeric attribute the precision will be stored in amt_prec.

The Dates table contains specific date ranges. Each date range has a start date (from_dte), end date (to_dte), effective date and description (narritive). A date range always falls in a period (t_macal) and is linked to an item in the calendar structure in the central repository. The date range makes it possible to store the same attribute against a structure matrix combination for different dates.

The Matrix Model has various advantages when compared to the Typical Dimensional Model.

- Different cubes can be stored in the combined Multidimensional Model: Instead of creating new database tables for every dimension, different structures can be grouped together. Every structure grouping has a unique identifier (t_stcube) in the combined dimension table.
- Attributes can be added to the fact table without modifying the physical database table.
- The Matrix Model is part of the operational system. This solves the problem of data inconsistency between the data warehouse and the operational system.
- The Matrix Model can be used for both real time access and data warehousing.

3. Quality Characteristics

Quality is often misunderstood when used within a software context. The concept of quality implies excellence of some sort, but this is not what is implied in a software context. Quality in a software context can be defined as the totality of features and characteristics of a product that satisfy specified or implied needs. Informally the quality of software can be stated as the extent to which the product satisfies its specification [3].

The quality characteristics of Functionality, Reliability, Usability, Maintainability, Portability and Efficiency are very important considerations when developing software. The relevance of these quality characteristics in real time access to multidimensional matrices can be identified as follow:

- Software must provide the required functionality. To ensure developed software is suitable for its intended application, the software must provide an appropriate set of functions. To ensure accuracy tests must be performed on the software. It is also important that the software can interact with other systems.

- A reliable system must have a high level of fault tolerance, which ensures that it maintains a high performance level in the event of software faults. To ensure availability, the system must be capable of performing a required function at any given point in time.
- User procedures must be uncomplicated and effective, to allow individuals to operate the system effectively.
- Software systems must be maintainable. Object Orientation techniques used in software development simplify maintainability. Modified software must be tested.
- Portability is an important quality characteristic. This ensures that the software can be transferred from one environment to another without any changes in the code. The developed software must be database and operating system independent.
- Developed software must be efficient to meet its required outputs satisfactorily.

The application of the six quality characteristics as related to software developed for the Matrix Model, is discussed in the following section.

4. Addressing of the quality characteristics

4.1 Functionality

It is important that the software provides an appropriate set of functions for specified tasks and user objectives. To ensure this, Requirements and Specification documents are created. The following functions to manipulate attribute values in the fact table were identified: accumulation, subtracting, adding, deletion, updating and reading. To manipulate records in the dimension table the "FETCH" functionality can be used. "FETCH" functionality checks if a specified record exist in the dimension table. If the record exist, the technical key of that record will be returned to the calling object. If the record doesn't exist, a new record will be created for the specified structure matrix and the created technical key will be returned. The user must also be able to step through the entire or specified dimension records and manipulate the fact table records. This functionality is referred to as the "STEP" functionality. A list of all dimension records for a specific profile can be obtained by using the "FETCHLIST" functionality. The data obtained by using the above functionality must be presented to the users. For this reason, a suite of visual objects was developed. The visual objects invoke different actions on non-visual objects by firing rule events. Rules consist of conditions, calculations and object executions and are called from events within objects. The Rules Engine manages the execution of rules.

The specification of a product clarifies what the product must do. Detail design in Unified Modelling Language (UML) is used to determine how to do it. Detail design may detect faults in the Specification document that can be resolved at this point.

The implementation of the product starts when the detail design is completed. The requirement, specifications and detail design of the product ensures that the correct functionality is implemented in the final product.

It is important that the developed software provides the correct and anticipated results. Testing, verification and validation must ensure this.

The Verification and Validation plan, created for the developed software, contains references to design documents, tests and contains a test plan for the product. The test plan includes the following:

- Test items
- Features to be tested
- Features not to be tested
- Approach (e.g. example static and dynamic testing)
- Item pass/fail criteria
- Test deliverables
- Testing tasks

- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Test Designs
- Test case specifications
- Test procedure
- Test reports

It is important to note that testing is not a separate phase in the development life cycle but an intrinsic component of every phase.

By referencing the requirements, specifications and detail design different integration and implementation testing, functionality groupings may be identified (testing to specifications). These groupings require that each path through the software must be executed at least once (functional testing). The relationships between the different groupings and the different paths of the groupings are modelled in the Verification and Validation Plan.

The developed software is tested with a test engine that executes test scripts. Different test scripts are created for every functionality grouping. A standard template is used to create the test script. Every test script as well as all the other documents must be maintained under version control. The tester specifies the action to execute, input for the action and expected or partial expected results in the test script. This information is gathered by referencing the requirements, technical documents and verification and validation plan. The test engine then reads the information from the file, executes the action with the specified input and compares the results with the specified expected results. All the error messages and output are dumped to a new text file. Non conforming issues, where the expected and true results are not the same, are dumped to another text file. It is the tester's responsibility to analyse and document all the non-conforming issues, in the verification and validation plan. The developer is responsible for the necessary maintenance to the code. The changed software will again be tested with the same test scripts. This process continues until no non-conforming issues remain.

To ensure that the developed software can interact with other systems the Open Interface (OI) architecture as illustrated in Figure 6 is used. The Open Interface establishes an environment that supports:

- Services to be deployed, using 'plug and play' architecture within different technologies.
- External Clients to gain access to services through a set of client interface adapters.

The OI architecture consists of client adapters, a broker and services. External clients; e.g. a Visual Basic or Java application may access services through a client dependent Client Interface Adapter (CIA). The CIA is dependent on the interfacing capabilities of the foreign technology (Level 1 API). The CIA encapsulates the data retrieval and marshalling specifics prescribed by each client type and converts it to the level 2 API.

The broker expects data in a level 2 Application Program Interface (API). The level 2 API consists of a service, error list and a list of parameters. The service can be an internal service; e.g. DynamicSQL, as illustrated in Figure 6, or an external service that can be plugged into the system. The broker will then invoke the service. Internal services are invoked using the level 3 API. This API consists of an error list and a list of parameters. The described multidimensional matrix functionality is embedded in the Cube Maintenance service.

The API levels show at which phases an external vendor or a new internal service may 'plug in'. Level 1 is the most straightforward, where the client specific CIA must be developed to convert the level 1 API call to a level 2 API call. The level 3 API allows new services to be added. By using this architecture, any application can interface with the developed multidimensional matrix software.

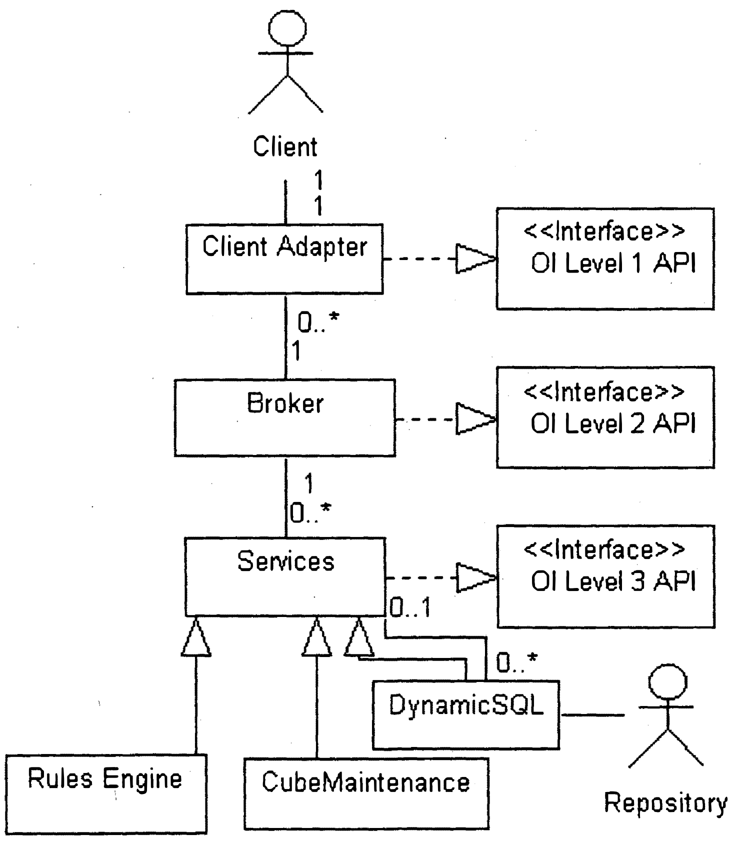

Figure 6: Open Interface Architecture

4.2 Reliability

It is impossible to develop reliable software without a set of standards and processes. The standards include certain software development standards for, e.g. naming conventions and error checking. Certain processes must be followed when developing software. These include the creation of a requirements document, detail design using Unified Modelling Language (UML), the creation of a verification and validation plan and the creation of technical documents.

Software vendors need to know if their products are reliable before they are delivered to customers. Reliability is a measure of the frequency and criticality of product failure. Product failure, under permissible operation conditions, is an unacceptable effect or behaviour that occurs as a consequence of a fault [4].

For system reliability, the system must have a high level of fault tolerance. This is to ensure that the system maintains a high level of performance during software faults or during infringements of its specified interface. Intensive error handling ensure this. Various corrective actions must be taken, depending on the severity of the faults.

For real time access to multidimensional matrices, it is important that the system is available, to perform a required function, at a given point in time, under stated conditions of use. Only a reliable system will always be available under the above circumstances. A disadvantage of a data warehouse based on the typical dimensional model is the downtime when new data is loaded into the database. Using the Matrix Model, downtime is minimised.

A test engine is one of the tools used to ensure that developed software is reliable. Different test scripts are executed and records are kept of the frequency of failures. Software with a high frequency of failures will not be released to customers.

There is no guarantee that using a test engine will identify all the faults in the software. It is possible to exercise every path without detecting every fault. A path can also only be tested if it is present in the software product. To further ensure that the software is reliable and that all the logical paths are present inspections are performed. The inspection process consists out of four steps: Overview, Preparation, Inspection and Rework [5].

- Step 1. The designer first describes the overall area being addressed and then the specific area he has designed in detail to the audit participants. Design documentation is distributed to all the inspection participants on conclusion of the overview.
- Step 2. The participants of the code inspection prepare individually using the design documentation to try and understand the design and logic of the product.
- Step 3. The presenter walks the inspection team through every line of code with the objective to find errors. The purpose is only to find and document the errors, not to correct them.
- Step 4. The following step is the rework where the designer or developer resolves all errors or problems noted in the inspection report. In the follow-up phase the moderator must ensure that all the issues are resolved.

Audits are performed on the software to ensure that:

- Coded software products adhere to design documentation.
- Testing requirements prescribed by the documentation are adequate for software product acceptance.
- Test data adhere to specifications.
- Software products have been successfully tested and meet their specifications.
- Test reports are correct and discrepancies between actual and expected results were resolved.
- User documentation complies with specified standards.
- The schedule adheres to the plans.

The Matrix Model architecture, test engine, code inspections and audits ensure that the developed software is reliable.

4.3 Usability

The visual suite of multidimensional matrix objects is designed to be very user friendly. The term "user friendliness" refers to the ease with which computer users can communicate with the software product. If users find it difficult to understand, learn and use the software, then the product will either be used incorectly or not at all.

Check boxes, push buttons and radio buttons are used to provide the users with easy to use front ends. All the fields have describing labels. When double clicking on text fields, list forms are displayed from where the user can select the correct item. User documentation is included in the help facility. Information is presented as a matrix of items and values. Future research must still be done to find different methods displaying the data to users, e.g. example tree views.

The developed objects can be configured according to business needs. Therefore it is very important that the persons responsible for the configuration is trained in cube concepts. Training sessions on multidimensional matrix concepts and the related software is often presented. Detailed user manuals and object technical documentation is used in these training sessions. Objects aren't released before the user manuals and technical documentation is created.

4.4 Maintainability

During the software life cycle most time is spent on maintenance. The total cost of maintaining a product over its lifetime is more that twice what it costs to develop that product. This makes it very important to develop maintainable software. Software is maintainable if corrective maintenance and enhancements can easily be performed on the software.

Standards, object orientated techniques, detailed documentation, detail design and the test techniques ensure that the developed software is maintainable. Development standards ensure that all the developed software has the same look and feel, which simplifies the maintenance process.

Object oriented concepts such as inheritance, polymorphism and encapsulation make it easy to identify problem areas. The problem areas can then be modified and only the modified objects have to be tested. Enhancements can be implemented by creating new objects; most of the time existing objects don't have to change.

Up to date detailed documentation and detail design for the product as a whole and for each individual object is available. This makes it easy to analyse deficiencies or causes of failures and to minimise regression faults. Regression faults occur when making a change to one part of the product may cause another part of the product to fail. Executing the original test scripts with the test engine performs testing for regression faults.

Version numbers are used to track changes in objects. Whenever an object is changed the version of the object is incremented and the reason for the change is documented.

The same techniques used to develop the original product that ensured that the quality characteristics are met must be applied when maintaining the software.

4.5 Portability

A product is considered to be portable if it is significantly less expensive to adapt the product to run in a new environment than to write a new product from scratch. This means that the product must be database and platform independent.

To ensure that the developed software is operating system independent the development tool like Dynasty is used. Dynasty is a fourth generation language that generates C code. Dynasty objects are partitionable, platform independent and target independent. Code can be generated for different platforms by changing a setting in the program object. These platforms include Solaris, OS2, Windows, WINNT and many more. This functionality makes it easy to deploy the software on different operating systems.

For the software to be database independent, open database connectivity (ODBC) is used. One central object is used to manage the database interaction. For this reason, modifying a single object may solve database incompatibilities. Database interaction is done with Structured Query Language (SQL) as defined by the American National Standards Institute (ANSI).

4.6 Efficiency

It is essential that the developed product provide the appropriate response to be efficient. Small single-table queries need to be performed instantaneous. Large join queries are expected to run for seconds or minutes. Performance is an inherent problem when accessing multidimensional matrices. The following techniques have been used to increase the response time of the developed software: Denormalization of database tables, Indexes, Temporary database tables and Stored procedures.

4.6.1 Denormalization

Denormalization improves performance by reducing the number of joins required during query execution. This technique was applied to the dimension table in the Matrix Model. Instead of joining the dimension table to an additional table containing the structure matrix technical keys, the dimension table was denormalized to contain these keys in the t_stgr_*nn* columns. (Refer to Figure 5)

4.6.2 Indexes

Performance is further improved by using indexes. In the dimension table there are indexes on the primary key (t_stvlnk) and a combined index on t_stcube and all the t_stgr_*nn* columns. Two indexes exist on the fact table. The first index on the primary key (ent_name, t_stdates, t_stgroup and t_entity) and the second index on ent_name, t_entity and t_stdates. All database queries in the developed software make use of the indexes.

The developed software creates temporary database tables. A specific subset of the data in the permanent dimension table is copied to a temporary table. Instead of joining the permanent dimension table to the fact table the smaller temporary table is joined to the dimension table.

4.6.3. Stored procedures

Stored procedures are used for database intensive queries. Instead of sending different SQL statements across the network to the database server a single statement to invoke the stored procedure is sent across. This concept reduces network traffic. In the database stored procedures, SQL cursors are used to join the dimension table record to its related fact table records, one at a time; instead of joining all the dimension table records to the related fact table records in one join.

5. Performance optimisation

Different techniques can be used to optimise the performance of multidimensional matrices. These techniques include partitioning, parallel processing, multidimensional servers and aggregates.

- In a partitioned data model, the data is segmented into logical areas. Data can, for example, be partitioned by month. Instead of having one fact table, different fact tables can exist for every date range.
- Parallel processing divides the workload among multiple processors. Therefore one processor is able to scan the database, while another sorts data.
- A specialised database can be created on a dedicated multidimensional server that will maintain the data in a format geared toward the type of maintenance and retrieval users will be making.
- Aggregates can be pre-computed on some subsets of dimensions and their corresponding hierarchies to speed op access to multidimensional data. These aggregates improve query response times [2].

6. Conclusion

Future research will be done to further optimise the software developed for the Matrix Model. Different tools and techniques will be used to do this.

The importance of quality is often recognised more by it's absence at the end of a software project, than by its presence at the start of a new project. Quality cannot be added into a product after it has been developed, it must be built into it from the start. In this paper it is shown how it is possible to develop software with a high degree of excellence that provides real time access to multidimensional matrices.

A supporter of data warehousing can argue that transaction processing and decision support activities must be separated into different databases. In this paper we have proved that the Matrix Model can hold years of history, require relatively little additional disk space, have integrated data and can provide high quality real time data without being separated into a different database.

References

1. J. Giles. "Is data warehousing only First Aid", Database programming & Design, **11**:34-39,47,1998
2. K. Kimball. The Data Warehouse Toolkit. JOHN WILEY & SONS, INC, New York, 1996.
3. J. Knight. Software Quality: Principles. SEAL, Johannesburg, 1994.
4. S.R. Schach. Software Engineering. 2nd edition. IRWIN, Burr Ridge, 1993.
5. D. Wheeler, B. Brykczynski and N. Meeson. Software Inspection an Industry Best Practice. IEEE Computer Society Press, Los Alamitos, 1996.