Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik
2003-05-27

Are SQL Server, DB2, and Oracle really relational?

Abstract

Today, we have a huge quantity of information in the world. The development of information systems makes it possible to manage our needs. Database Management Systems (DBMS) have played a great role in the success of information systems. The most influential theory in DBMS is the Relational Database Model (RDM), which was established by E. J. Codd in the early 1970s.

In this thesis, three database products, SQL Server, DB2, and Oracle are compared, which is unique because in earlier studies they have only been evaluated one database at a time. They are called Relational Database Management Systems (RDBMS), based on RDM. Their total market share is more than 80 % of the whole RDBMS. Codd points out important requirements for RDBMS, known as Codd's twelve rules. The purpose of this study is to evaluate if and how closely each database vendor has followed Codd's rules when developing their products. In summary, eight of the rules are put into practice by all of the vendors. The most difficult rules to implement are: 1. *modification of the view table* and 2. *handling of null values*. Our conclusions are that all three databases qualify as relational, and they have equal difficulties with the *update view* function and handling of null values.

Key words: Database, RDM, Codd, View, Null

Author: Michiko Itoi Blomgren
Supervisor: Alan B. Carlson
Master Thesis, 20 p

Acknowledgements

I would like to express my sincere thanks to Mathias Johansson who was my supervisor and to the staff at Volvo Information Technology for providing a positive atmosphere and sharing their professional experiences.

I would like to thank to my supervisor, Alan B. Carlson for good advice and support.

Table of contents

1. Introduction

Because of the technological improvements, especially the Internet and mobile technologies, we can get virtually any information that we need instantly, and for example book tickets all over the world easily. We can send e-mail using our mobile telephone or PDA (Personal Digital Assistant). Many companies are globalising in conjunction with e-Commerce and m-Commerce. Nowadays it is popular to use Data warehousing and Data mining, integrating data to be able to use strategies and analyse data. (Connolly & Begg & Strachan, 1999, chap.25-26) Such techniques increase the quantity of data and complexity of data management. In our modern society it is difficult to manage without Database Management Systems (DBMSs).

DBMS software is based on a data model. The data model is a collection of concepts that can be used to describe the structure of a database. The data model describes the structure of a database such as data types, relationships, and constraints that should hold on the data. (Elmasri & Navathe, 2001, p.24)

The history of DBMSs began in the 1960s Apollo moon-landing project. They needed to manage vast amounts of information to support the project. In the 70s, hierarchical database system (HDBMS), the first of which was called Information Management Systems (IMS), was launched from IBM as a result from the Apollo project. Almost at the same period, network database system (NDBMS) from General Electric was introduced. This product was called Integrated Data Store (IDS). A relational database system (RDBMS) was introduced by Edger F. Codd (IBM) in the 70s and was launched in the early 80s. (Connolly et al., 1999, pp.24-26)

Firstly, a hierarchical data model is designed in a tree (or parent-child) structure and allows only one-to-one or one-to-many relationships between entities. The model is fast when it comes to getting information, but it is not a flexible structure. Sometimes the role of the entity (parent or child) is not clear and unsuitable for a hierarchical model. Secondly, a network data model has a more flexible structure than the hierarchical model and allows many-to-many relationships between the entities, but it easily becomes complex and difficult to manage. Thirdly, the relational data model is more flexible than the hierarchical model and easier to manage than a network model. The relational data model is the most widely used model today. (Wallace &Cox, 2002)

Data migration means the process of translating data from one format to another. This usually requires converting the data into some common format that can be output from the old database and input into the new database. Since the new database may be organised differently, it may be necessary to write a program that can process the migrating files (searchCIO.com).

A DBMS is expensive and to change the DBMS software from one vendor to another is time-consuming and economically demanding. Craig S. Mullins (2003, May) has written in ‘Complexity complicates database upgrades and migrations’ that a complex database environment increases the difficulty of upgrading and data migration. As the number of database servers, instances, applications, and users increase, the complexity grows.

It is critical to choose the right DBMS software for an organization from the beginning, but the choice of DBMS software vendor is not easy because the DBMS software consists of several components and they are large and complex. It is not always clear which DBMS software is better than the other. The *Open System Databases* (OSD) department and the *Mainframe Databases & Transaction Systems* (MDTS) at Volvo Information Technology (Volvo IT) in Gothenburg needs to know the differences between three DBMS software products, SQL Server, DB2 and Oracle to be able to assist their clients in their choice of database.

OSDs are subdivided into two main groups, depending on which database software they work with, SQL Server or Oracle. The MDTS department works with DB2. Their main task is to see to that there are databases in all the Gothenburg offices within the Volvo group. They even have other clients round world, for example ASSA, Nobel Biocare and Skolförvaltningen, Södra skogsgägarna, etc.

Generally a system developer contacts the OSD or MDTS department to ask for assistance in creating a database, or to ask technical questions. The client usually has a database which is ready for use and the role of the department is to help performing a smooth database transfer in the working environment. There is also another case where clients need help to design a database and to create it directly at Volvo IT's server-hotel. In many cases, the clients want to buy the consultants' time and they often need to give some advice regarding the best choice of platform or database engine for the particular client. The consultant also usually needs to give advice regarding handling of different data types, normalization of databases, etc. SQL Server can be run only on the Windows platform. Oracle can be run on both Unix and Windows platforms today, but they are planning to launch Linux and Mainframe versions later this year. In Volvo DB2 is run only on Mainframe.

It is important to consider many aspects when it comes to the choice of DBMS. It constitutes very complex software and usually has far-reaching effects on all groups and individuals in any organisation. The criteria that have to be met to fulfil the needs of an organisation must be clarified and characterised in detail. There are thousands of criteria and benchmarks (commercial and freeware) to support the choice of DBMS. We can measure which SQL constraints are allowed, response speed for a question, and maximum capacity for users at the same time, etc. In this thesis we will examine SQL Server, DB2, and Oracle, which are the leading RDBMS software products and cover more than 80 percent of the DBMS market share (Boss Consulting).

The user requirements are the starting points for the choice of criteria (Connolly, 1999, p.135). However, in this study we do not have a specific user and hence there are no specific requirements to consider. The choice of benchmarking methods is one of the most difficult parts when it comes to DBMS evaluation. (Robson, 1997, p.426) Usually, vendors choose benchmarks that suit their products, or adjust benchmarks. We decided to use Codd’s twelve rules as our criteria to compare the databases in a general way. (Date, 1991, p.391) In 1985, Edgar F. Codd published a set of 12 rules, which he defined as an evaluation scheme for a software product that claimed to be a Relational DBMS. They still serve as a foundation for a definition of a Relational DBMS (Bull, M., 2002). Codd’s rules implicated that satisfying the rules was by and large a technically feasible proposition, and there were clear practical benefits to the user if the system in fact did satisfy the rules. (Date, 1991, p.391) Codd’s rules are not just a theoretical model but a practical tool that can be used for evaluation of DBMS.

The main question is:

Are SQL Server, DB2 and Oracle really relational DBMS software?

The relational data model is the most powerful. E. F. Codd is the founder of the relational data model. The relational data model was introduced already in the 70s, but still is the most common model for DBMS. We examine how close the three DBMS software products fulfil the requirements of Codd’s twelve rules for relational models. This study is unique because earlier studies have evaluated only one database at a time, using Codd’s twelve rules. A comparative study like the present one has not been performed.

2. Methods and Materials

The main method in this thesis was to study printed books and electronic documents from each database vendor. In addition, interviews were performed with the experienced staff at Volvo IT. In the next sections, it is explained what kind of literature and documents were used and how the interviews were accomplished.

2.1. Literature studies and other documents

Because of the character of the study, database theories are the major issue. *An Introduction to Database Systems* (Date, 1991, 2000), *Fundamentals of Database Systems* (Elmasri and Navathe, 2000) and *Database Systems* (Connolly et al., 1999) are used as the theoretical foundation for information about database systems. Date's books are mainly used to explain the relational data model concept for RDBMS. Date is a previous employee at IBM and was one of the first persons to recognize and support Codd's pioneering work on the relational model. Date's book became a bible for RDBMS. In Date's book (1991) there are some explanations of Codd's twelve rules, but the explanations are quite simple and do not provide enough detail to fully understand how to implement them. The homepages from the following companies or academic institutions were used to supplement the explanations of Codd's twelve rules: http://www.kingston.ac.uk, http://engr.smu.edu, http://www.itworld.com, http://www.dbmsmag.com, http://www.webopedia.com. All four books provided information about concepts and structures of database systems, and helped review the results. The purpose of using several resources was to get more comprehensive information about the same concepts and to compare between sources.

DBMS software vendors' homepages and documents were also used. The three vendors have huge libraries in their web sites. They were used to find information regarding whether the systems comply with Codd's rules or not. The main resources for DB2 came from DB2 Technical Support for version 8 and DB2 UDB for OS/390 and z/OS Version 7, which consist of many organized online books (D14). There are more than 50 books which consist of 200 to 1500 pages each in the DB2 Technical Support site. The books are categorized into nine subject headings. In version 7, DB2's search engine can fetch articles in every related book if we ask a specific question. This was very convenient if we could not find the information easily in the DB2 Technical Support site. Oracle also provides an organized library (Oracle 9i Database List of Books (Release 2 (9.2)), 2003). It includes close to 150 books which we could either download as PDF files or view the same information in HTML-files. Of course, the Oracle site provides good search engines, too. SQL Server has different structure of its library somehow. Even though there is a search engine and many books, it was not so easy to find some information from our specific point of view. DB2 and Oracle have more detailed information in their web sites than SQL Server does. The documents for SQL Server are easy to read with abundant pictures, but in general it was more difficult to find, for example, information regarding specific problems or questions. The documents are suitable for the general user.

However, the other two vendors have prepared different levels for users with different knowledge about the database systems.

2.2. Interview

Opportunities were given to ask questions to the experienced staff from three different groups working specifically with SQL Server or Oracle at the Open System Databases department or DB2 at the Mainframe Databases & Transaction Systems department at Volvo IT. The interviews were informal. The staff members were asked to ascertain to what extent each of Codd’s rules was implemented according to their empirical knowledge. They were also asked if the accuracy of the statements from each vendor’s manual or homepage could be confirmed.

As described above, we combined literature studies with interviews. For example, Codd’s first rule says that all information has to be represented in a tabular form. We found some documents from each vendor’s homepage and asked if the statements and the practical experience of the staff were in agreement. In another case, as we could not find a proper answer for the eighth rule, the physical independence rule, from neither their homepages nor their papers, the interviews were very valuable. All the vendors describe that the primary key constraint is an option, but this is the very core of RDBMS, so we questioned whether this really could be an option. After the interviews with each group, it was clear to us that in practise, tables without a primary key are very unusual or even non-existent. We were also given the opportunity to ask questions about very fundamental database concepts, for example, how the locking concept works in DBMS.

3. Theories

This chapter is about DBMS theories in general which are used for comparing the DBMS software products.

3.1. Database systems

In this section we will explain the limitations of the file-based system which was the general repository solution for data management before the advent of databases. Then, we will describe definitions about the database and DBMS, the structure of a DBMS and entity-relationship model (ER model), which are needed to understand the definitions. Eventually other important database terms will be illustrated.

3.1.1. File-based system vs. a database system

The concept of databases has created the need for more flexible and effective systems than the inflexible file-based system (Connolly et al., 1999, p.11). Because the data in the file-based system are isolated in separate files, duplication of data occurs easily, and several files have to be changed if data need to be updated or deleted. It leads to difficulties for maintenance. The file-based system defines the physical space and storage of the data files and records in the application program. It causes a program-data dependency (Connolly et al., 1999, p.13), *i. e. incompatible file formats between different application programs and queries may cause problems. The systems are not flexible. For example, the upgrading of an application program or the increment of another software or hardware is always difficult because of the dependency on the application programs. A database approach is a way to solve these problems.*

3.1.2. Definition of a database and a DBMS

Connolly et al. (1999, pp.14-15) defines that *a database is a shared collection of logically related data (and a description of this data), designed to meet the information needs of an organisation*. The definition of 'shared' in a database context is that many departments and users simultaneously can use a single, large repository of data as a common resource. The database includes not only the organisation's data but a description of this data, which is called the system catalog or meta-data. This feature facilitates program-data independence. The data and meta-data are stored in different files in a DBMS system, not in an application program. For example if one of the length of the fields needs to be changed because a postcode becomes longer than before, we need only to change the meta-data and the application programs can be used without interruption.

An entity-relationship model (ER model) is used to analyse and design the information needs of the organization. The model consists of three main terms, entity, attribute, and relationship. An entity is a distinct object in the organisation that is to be represented in the database. An attribute is a property that describes some aspect of the object that we wish to record, and a relationship is an association between several entities. More detail about ER model is explained in the next section 3.1.4. By means of the relationship in the ER model, 'logically related data' is stored in the database. (Connolly et al., 1999, p.15)

A database management system (DBMS) is a *software system that enables users to define, create, and maintain the database and provides controlled access to this database through the application program*. A DBMS uses Data Definition Language (DDL) and Data Manipulation Language (DML) in the DBMS. (Connolly et al., 1999, p.16)

The DDL is used that users to define and name the entities required for the application and the relationships that may exist between the different entities (Connolly et al., 1999, p.45) and DML is used that users insert, update, delete and retrieve data from the database. Connolly (1999, pp.54-58) gives ten expected functions from a DBMS.

1) Data storage, retrieval, and update;
2) User-accessible catalog – a DBMS must store information about data in a system catalog and user can access meta-data;
3) Transaction support – it means that data should be consistent, that is, ensure that either all the updates corresponding to a given transaction are made or that none of them are made;
4) Concurrency control – a DBMS is made for several users simultaneously. The DBMS must manage that the users can access data without interfering with another users even though the same data are needed;
5) Recovery services – a DBMS must have some preparation for different kinds of damages (hard disk or software damages);
6) Authorization services – a DBMS must have control that data is accessed by only authorized users;
7) Support for data communication – a DBMS must integrate with communication software;
8) Integrity services – a DBMS must ensure that both the data in the database and changes to the data follow certain rules. There are some rules for consistency of stored data. See section 3.2.10. R10 Integrity independence in detail;
9) Services to promote data independence – a DBMS must facilitate to support the independence of program from the actual structure of the database. See the beginning of this section 'program-data independence';
10) Utility services – a DBMS should be provide a set of utility services such as statistical analysis programs, monitoring facilities import facilities, garbage collection and reallocation.

3.1.3. Structure of a DBMS

The program-data independence is mentioned in the previous section 3.1.2. The three-level ANSI-SPARC architecture supports the concept and is a common structure for databases. The following points will describe the three levels in detail (Connolly et al., 1999, pp.41-42).

- External level – the user's view of the database. This level is concerned with the way the data is seen by individual users.

- Conceptual level – the community view of the database. This describes what data is stored in the database and the relationships among the data. This level includes all entities, their attributes, and their relationship, the constraints on the data, semantic information about data, and security and integrity information.

- Internal level – the physical representation of the database on the computer. This level describes how the data is stored in the database. They are storage space allocation for data and indexes, record descriptions for storage (with stored sizes for data items), record placement, and data compression and data encryption.

There are external, conceptual, internal schemas, which define the respective level. The DBMS performs mappings between these three different types of schemas to relate them correctly. We give an example how different levels and mapping works together. As shown in figure 1, corresponding data items can have different names at different levels and different individual views in the external level. If a change is made to the physical definition, the conceptual/internal mapping must be changed, but the conceptual schema can remain untouched. Transformation between the users' view (external level) and the physical storage (internal level) serves to hide the complexity on the physical level. This increases the flexibility and possibilities of adaptation.

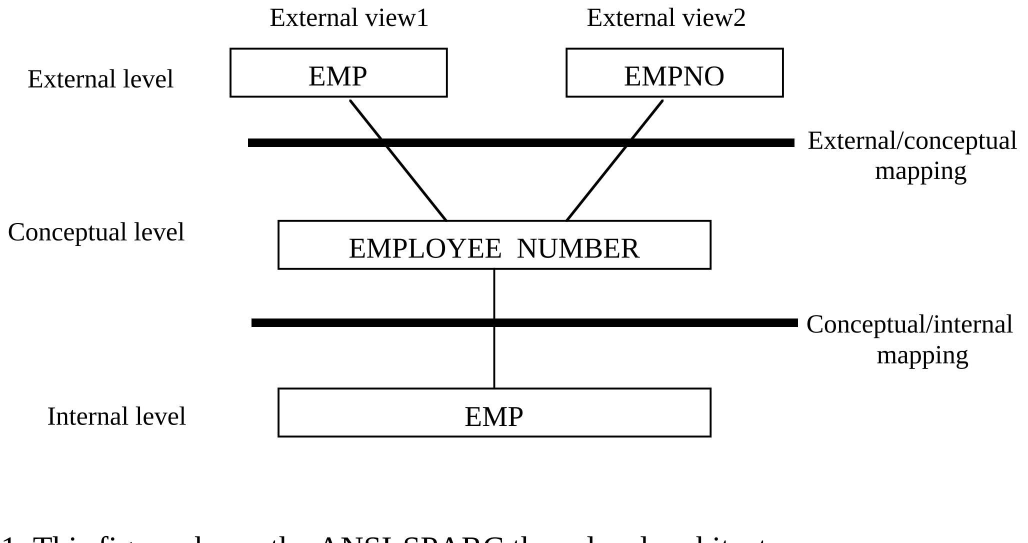

Fig. 1. This figure shows the ANSI-SPARC three-level architecture.

3.1.4. Relational data model and ER model

When we construct a database, we use some kind of data models to represent data, to make the data understandable and to design the database afterwards. A model is an abstraction or a representation of the real world. A data model represents the organization itself. It should help users clearly and accurately to communicate their understanding of the organizational data. (Connolly et al., 1999. p49)

The data model consists of three parts, that is, *entities*, *attributes*, and *relationships*. An entity represents a real-world object or concept, such as an employee or a project that is described in the database. An attribute represents some property that describes some aspect or condition of the object that we wish to record, such as the employee's name, or salary. A relationship among two or more entities represents an interaction among the entities; for example, a relationship between an employee and a project. The data models help to understand 'logically related' data. (Elmasri & Navathe, 2000, p.25)

The relational data model uses two-dimensional tables which represent entities and consist of columns and rows. A column is an attribute which is property and field. A row is a ***tuple*** which is an instance of an entity or relationship or whatever is represented by the relation. Usually one of the columns in a table is a primary key which always has a unique value (Brown, The Relational Model).

The entity-relationship model (ER model) is a high-level conceptual data model. It is a set of concepts that describe the structure of a database and the associated retrieval and update transactions on the database (Connolly et al., 1999, p.150). The purpose of the database system is to show users an abstract view of data, hiding certain details of how data is stored and manipulated (Connolly et al., 1999, p.39). The ER model supports an abstract and general description of the information requirements of the organization that is to be represented in the database.

The ER model uses three important terms, entities, attributes, and relationships to know the information of the organization. An entity is an object or concept that is identified by the enterprise as having an independent existence (Connolly et al., 1999, p.150). An attribute is a property of an entity or a relationship type. A relationship is a meaningful association among entity types.

3.2. Codd's Twelve Rules

Thanks to Codd's work, DBMS became flexible and still easier to control, but Codd was very anxious about DBMS software vendors which called their own products 'relational' without really relational DBMS features (Date, 1991, p.376). Maybe their products could make tables, but that was not enough to call them "relational" databases.

3.2.1. R1: The information rule

The information rule simple requires all information in the database to be represented in one and only one way, namely by values in column positions within rows of tables (Date, 1991, p.392). All information means that all data from application program, meta-data or information about the database and must be stored in relational tables.

3.2.2. R2: The guaranteed access rule

This rule is essentially a restatement of the fundamental requirement for primary keys. It says that every individual scalar value in the database must be logically addressable by specifying the name of the containing table, the name of the containing column, and the primary key value of the containing row (Date, 1991, p.392). It says that user should only access through a combination of the table name, primary key value, and column name.

A primary key is the core for the relational database model. It is a column or set of columns that uniquely identify all the rows in a table. Only one primary key must be in a table and is not allowed to take NULL values, which means the field is empty (The term NULL will be explained the next paragraph in more detail.).

3.2.3. R3: Systematic treatment of null values

The DBMS is required to support a representation of “missing information and inapplicable information” that is systematic, distinct from all regular values, and independent of data type. It is also implied that such representations must be manipulated by the DBMS in a systematic way (Date, 1991, p.392).

A null value means that a field is empty because the value of that field is unknown, missing, or inapplicable. The null values are not same such as an empty character string, a string of blank characters, or a zero or any other number and should be distinct from them. The null values must be allowed in the relational database. It can't apply to primary keys and most database implementations support the concept of a non-null field constraint that prevents null values in a specific table column. (Parkhurst**,** 2002)

3.2.4. R4: Active online catalog based on the relational model

The system is required to support an online, inline, relational catalog that is accessible to authorised users by means of their regular query language (Date, 1991, p.392). A relational database should be self-describing (Moore).

The catalog is the place where – among other things – all of the various schemas (external, conceptual, internal) and all of the corresponding mappings (external/conceptual, conceptual/internal) are kept. In other words, the catalog contains detailed information (sometimes called descriptor information or meta-data) regarding the various objects that are of interest to the system itself (Date, 2000, p.69).

3.2.5. R5: The comprehensive data sub-language rule

The system must support at least one relational language that

1. has a linear syntax,
2. can be used both interactively and within application programs, and
3. supports;
    - data definition operations (including view definitions),
    - data manipulation operations (update as well as retrieval),
    - security and integrity constraints, and
    - transaction management operations (begin, commit, and rollback).

(Date, 1991, p.392)

Rule 5 mandates using a relational database language, such as SQL, although SQL is not specifically required. The language must be able to support all the central functions of a DBMS. (Moore).

3.2.6. R6: View updating rule

All views of the data which are theoretically updatable must be updatable in practice by the DBMS (Date, 1991). Each view should support the same full range of data manipulation that direct access to a table has available (Parkhurst, 2002).

A view is a virtual table, which derives from at least a base table and creates a suitable look for certain users. The base tables are a concrete representation in the hardware. Because the view does not store data in the database but store query, that is, some SQL-statements, it is called ‘virtual’ table (Johnson, 1997, p216).

Date discusses two important principles when updatability of view is concerned. *The Principle of Interchangeability* is that there must be no arbitrary or unnecessary distinctions between base and derived table (view). The updatability of the database must not depend on the essentially arbitrary decision as to which tables we decide should be base ones and which we decide should be views. *The Principle of Database Relativity* is that those tables are all base tables from the user’s point of view, by definition for the database itself the choice of which database is the “real” one is arbitrary, just so long as the choice are all information-equivalent. (Date, 2000, p.295)

3.2.7. R7: High-level insert, update, and delete

Data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table. (Parkhurst, 2002)

3.2.8. R8: Physical data independence rule

The application programs are immune to changes made to storage representations or access methods (Avery). This means that the physical data structure, which is for example, hardware or disk storage methods, should not affect the user's ability to work with the data.

3.2.9. R9: Logical data independence rule

A user view's data should not be affected even in the case of changes of the logical structure of the database (for example, the growth of tables from addition of columns or from addition of tables) (Avery).

Date defines the logical data independence as *the immunity of users and user programs to changes in the logical structure of the database*. There are two perspectives, growth and restructuring about changing the logical structure. None of them should have any effect on users or users programs at all.

Growth
As the database grows to incorporate new kinds of information, the definition of the database must obviously grow also. There are two types of growth that can happen.

1. the expansion of an existing base relvar to include a new attribute, corresponding to the addition of new information concerning some existing type of object (the addition of discount attribute).
2. the inclusion of a new base relvar, corresponding to the addition of a new type of object (the addition of project information to the suppliers and parts database).

Restructuring
Occasionally it might become necessary to restructure the database in some way such that, although the overall information content remains the same, the logical placement of information changes – i.e., the allocation of attributes to base relvars is altered in some way. (Date, 2000, pp.293-294)

3.2.10. R10: Integrity independence rule

Integrity constraints must be specified separately from application programs and stored in the catalog. It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications (Date, 1991, p.393). Integrity constraints are rules that the DBMS enforces to keep the database from becoming inconsistent. According to Date (2001, p.251), the golden rule for integrity independence is that "*No update operation must ever be allowed to leave any relvar in a state that violates its own predicate. Likewise, no update transaction must ever be allowed to leave the database in a state that violates its own predicate*".

We examined five integrity constraints for each vendor.

NOT NULL constraints

This constraint is the ISO standard in the CREATE and ALTER TABLE statements. The NOT NULL constraints prohibit a database value with null, which means that the column with this constraint is not allowed to be empty. (Connolly et al., 1999, p433)

UNIQUE clause

The UNIQUE clause identifies a set of one or more columns that uniquely identify each row of the table. Again, every column that appears in the UNIQUE clause must also be declared as NOT NULL. SQL rejects any INSERT or UPDATE operation that attempts to create a duplicate value within each candidate key. Only one PRIMARY KEY is allowed in a table. UNIQUE clause is used if the primary key is chosen and another column is needed to be unique. (Connolly et al., 1999, p457)

PRIMARY KEY (entity integrity) clause

Primary key of a table must contain a unique, non-null value for each row. The ISO standard supports entity integrity with the PRIMARY KEY clause in the CREATE and ALTER TABLE statements. The PRIMARY KEY clause can be specified only once per table. (Connolly et al., 1999, p454)

FOREIGN KEY (referential integrity)

A foreign key is a field in a table that matches the candidate key column of another table. The value of the foreign key must be a key value in the corresponding parent table. The table constraining the foreign key is called the referencing, foreign, or child table, while the table containing the candidate key is called the referenced, primary, look-up, or parent table. (Rennhackkamp, 1996)

Referential Integrity: the database must not contain any unmatched foreign key values. ‘Unmatched foreign key’ means a foreign key value in some referencing table for which there does not exist a matching value of the relevant candidate key in the relevant referenced table. In other words, the constraint simply says: If B references A, then A must exist. Date says that it will never be feasible to provide declarative syntax for all conceivable responses. In general, therefore, it should be possible to specify a referential action of the form of “CALL proc(…), “ where proc is a user-defined procedure (Date, 2000, pp. 263-265).

Database updates are always atomic, which means all or nothing, even if under the covers they involve several updates on several relvars because if e.g., a CASCADE referential action (Date, 2000, p266).

The ISO standard supports the definition of foreign keys with the FOREIGN KEY clause in the CREATE and ALTER TABLE statements. SQL rejects any INSERT or UPDATE operation that attempts to create a foreign key value in a child table without a matching candidate key value in the parent table. (Connolly et al., 1999, p454)

For example, suppose Table B has a foreign key that points to a field in Table A. Referential integrity would prevent you from adding a record to Table B that cannot be linked to Table A. In addition, the referential integrity rules might also specify that whenever you delete a record from Table A, any records in Table B that are linked to the deleted record will also be deleted. This is called *cascading* delete. Finally, the referential integrity rules could specify that whenever you modify the value of a linked field in Table A, all records in Table B that are linked to it will also be modified accordingly. This is called cascading update (Webopedia, Referential Integrity).

SQL supports options regarding the action to be taken (Connolly et al., 1999, p455):

1. CASCADE: Delete the row from the parent table and automatically delete the matching rows in the child table. Since these deleted rows may themselves have a candidate key that is used as a foreign key in another table, the foreign key rules for these tables are triggered, and so on, in a cascading manner.
2. SET NULL: Delete the row from the parent table and set the foreign key column(s) in the child table to NULL. This is only valid if the foreign key columns do not have the NOT NULL qualifier specified.
3. SET DEFAULT: Delete the row from the parent table and set each component of the foreign key in the child table to the specified default value. This is valid only if the foreign key columns have a DEFAULT value specified.
4. NO ACTION: Reject the delete operation from the parent table. This is the default setting if the ON DELETE rule is omitted.

CHECK constraints

There are two types of CHECK constraints. One of them is called a domain constraint which decides a set of allowable values for one or more attributes. The other is an enterprise constraint which is needed to fulfil some conditions. (Connolly et al, 1999, p.74, p.452)

The ISO standard provides for specifying domains in the CREATE and ALTER TABLE statements. The CHECK clause allows a constraint to be defined on a column or the entire table. In a column constraint, the CHECK clause can reference only the column being defined. (Connolly et al, 1999, p.74, p.453)

3.2.11. R11: Distribution independence rule

Existing applications should continue to operate successfully (a) when a distributed version of the DBMS is first introduced; (b) when existing distributed data is redistributed around the system. (Date, 1991). The application operates from a logical point of view as if the data were all managed by a single DBMS on a single machine (Date, 1991, p.617).

Date (p 651, 2000) defines that *a distributed database system consists of a collection of sites, connected together via some kind of communications network, in which each site is a full database system site in its own right, but the sites have agreed to work together so that a user at any site can access data anywhere in the network exactly as if the data were all stored at user's own site*. "Each site is a database system site in its own right" means that each site has its own local 'real' database, its own local users, its own local DBMS and transaction management software, etc. He mentions twelve rules for a distributed database and they are:

1. Local autonomy – the sites in a distributed system should be autonomous.

2. No reliance on a central site – all sites must be treated as equals.

3. Continuous operation – the distributed systems should provide greater reliability and greater availability.

4. Location independence – users should not have to known where data is physically stored as if the data were all stored at their own local site.

5. Fragmentation independence – users should be able to behave, as if the data were in fact not fragmented at all.

6. Replication independence – users should be able to behave, as if the data were in fact not replicated at all.

7. Distributed query processing – supporting multi queries and optimisation.

8. Distributed transaction management – recovery control and concurrency control.

9. Hardware independence.

10. Operating system independence.

11. Network independence.

12. DBMS independence – it is about supporting homogeneity.

(Date, 2000, p.656 – p.663)

The rules cover a huge field, so we examined only rule number eight, Distributed transaction management. We chose especially to investigate a locking system for concurrency control issue.

3.2.12. R12: Non-subversion rule

If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system by (e.g.) bypassing a relational security or integrity constraint (Date, 1991, p.393).

If the database has any means of handling a single record at a time, that low-level of working must not be able to subvert or avoid the integrity rules which are expressed in a higher-level language which handles multiple records at a time.

There should be no way to modify the database structure other than through a multiple row database language like SQL. (Parkhurst, 2002)

4. Results

4.1. Comparison with Codd's Twelve Rules

4.1.1. Result of R1: The information rule

All of the database software vendors have system catalogs which store information about both the logical and physical structure of the database such as user tables, views, and indexes. The system catalog includes also the security information about the authority and integrity constraints (Glossary-5, Oracle9i Concepts [OC], 2001; Chap 1, DB2 Administration Guide: Planning v8 [DPL], 2002; Glossary [MS Glossary]).

4.1.2. Result of R2: The guaranteed access rule

Rule number two is also fully supported by all of the vendors. Each vendor has the same conditions such as a table that must have only one primary key, a primary key is not allowed to take NULL values, and must uniquely identify rows in a table. The vendors say that every table should have a primary key though it is not required, but in practice, there is no table without a primary key. Because of this fact, data that we want to get can be accessed by means of SQL statements which means that rule number two is fulfilled. (Chap 21, OC, 2001; Chap1, DB2 SQL Reference Volume 1 v8 [DSQL1]; [MS PK])

4.1.3. Result of R3: Systematic treatment of null values

It is not supported completely by any of the vendors, but the three-valued logic (3VL) is used by all of them. The 3VL is the management of NULL values in a Boolean expression. The use of Boolean AND, OR, and NOT operators return not only TRUE or FALSE but UNKNOWN also. The following tables show the results of the operators which all of the vendors use same logic.

(2.22. DB2 Search condition (2001). DB2 UDB for OS/390 and z/OS V7 SQL Reference Document Number: SC26-9944-02 [D1]; Chap 5, Oracle9i SQL Reference [OSQL]; [MS NULL])

AND and OR truth table

P predicate	**Q predicate**	**P AND Q**	**P OR Q**
True	True	True	True
True	False	False	True
True	Unknown	Unknown	True
False	True	False	True
False	False	False	False
False	Unknown	False	Unknown
Unknown	True	Unknown	True
Unknown	False	False	Unknown
Unknown	Unknown	Unknown	Unknown

Table 1 – The table shows the results of 3VL, using AND and OR operators. (Note that a null may not be equal to another null. The result becomes Unknown.)

NOT truth table

P predicate	**NOT**
TRUE	FALSE
UNKNOWN	UNKNOWN
FALSE	TRUE

Table 2 – The table shows the results of 3VL, using NOT operator.

However, the 3VL is not enough to cover the null value concept. Codd proposes that the relational model should be extended to include at least two types of nulls, “value unknown” and “value not applicable”. The three database systems supported only “unknown” one, but not “inapplicable” ones. (Date, 2000 p.585)

The next paragraphs we will explain each vendor in detail.

Null values in SQL Server

SQL Server manages NULL as an unknown value and NULL value is different from an empty or zero value. No two null values are equal. That is why comparisons between two null values return unknown, likewise, comparison between a NULL and any other value also. NULL values usually indicate data that is unknown, not applicable, or to be added at a later time. Naturally, some constraints, for example, NOT NULL, UNIQUE, and PRIMARY KEY are not allowed to null values. (MS NULL)

NULL values are eliminated if certain calculations are executed. One should be aware of this fact because some calculations, for example, an average can be inaccurate with NULL columns. One should control if NULL columns are eliminated or transformed into some other value. SQL Server states one should avoid using NULL values.

Null values in DB2

All data types include the null value (2.8. Data types. (2001). DB2 UDB for OS/390 and z/OS V7 SQL Reference: COVER Document Number: SC26-9944-02.) Distinct from all non-null values, the null value is a special value that denotes the absence of a (non-null) value. All data types except for some special values include the null value. A special values means that for example, constants, columns that are defined as NOT NULL, and special registers cannot contain null values; the COUNT (returns the number of rows or values in a set of rows or values., 3.1.2. COUNT. (2001). *IBM DB2*, DB2 UDB for OS/390 and z/OS V7 SQL Reference Document Number: SC26-9944-02 [D3]) and COUNT_BIG (It is similar to COUNT except that the result can be greater than the maximum value of an integer., 3.1.3. COUNT_BIG. (2001). *IBM DB2*, DB2 UDB for OS/390 and z/OS V7 SQL Reference Document Number: SC26-9944-02 [D4]) functions cannot return a null value; and ROWID columns (uniquely and permanently identify rows in a DB2 subsystem., 3.2.75. ROWID, SQL Reference v7) cannot store a null value although a null value can be returned for a ROWID column as the result of a query.

Null values in Oracle

Nulls can appear in columns of any data type that are not restricted by NOT NULL or PRIMARY KEY integrity constraints. Null is used when the actual value is not known or when a value would not be meaningful. Oracle treats a character value with a length of zero as null. However, they are working on it, and recommend that users do not treat empty strings as nulls (Chap2, OSQL, 2002).

4.1.4. Result of R4. Active online catalog based on the relational model

All of the vendors have a system catalog and they are created automatically when the database is created, and are updated during the course of normal operation. The authorised users can retrieve the meta-data with the same relational language as the ordinary regular data. (MS Glossary; DPL, 2002, p.7; OC, 2001, Chap.4)

4.1.5. Result of R5. The comprehensive data sub-language rule

MS SQL Server and Oracle use a sub-language, based on ANSI SQL-92 standard. PL/SQL is Oracle’s procedural language extension to SQL (Oracle, p1-13, Concepts). The ‘procedural’ language means that a language that allows the user to tell the system what data is needed and exactly how to retrieve the data (Connolly et al., 1999, p.46). SQL is a non-procedural language that does not have to describe how the data is retrieved. PL/SQL combines the both features. (Transact-SQL Tips [MS T-SQL]

Transact-SQL (T-SQL) in SQL Server corresponds to PL/SQL. DB2 does not have a procedural language extension of their own. Thus, SQL statements are embedded within the source files of a standard programming language like C, C++, COBOL (The Complete Reference DB2, 2001, p.600). All of the products fulfil the requirements described in section 3.2.5.

Since data from external and conceptual levels are organized using the same type of structure, the same sub-language is applied to data and meta-data in all of the products.

4.1.6. Result of R6. View updating rule

When the view tables have to be updated, all of the vendors supported this only partially. The products do not apply the principle of interchangeability and the principle of database relativity (see section 3.2.6) fully. It means that we cannot update the view table in same way as the original tables.

In general, the products of the three database vendors cannot update the view, if the view has aggregate functions, without an INSTEAD OF TRIGGER statement. The INSTEAD OF TRIGGER is a procedure that runs implicitly when an INSERT, UPDATE, or DELETE statement is issued against the associated view.

Aggregate functions operate on a single column of a table and return a single value. The ISO standard defines five aggregate functions (Connolly et al., 1999, p.401).

- COUNT - returns the number of values in a specified column.
- SUM - returns the sum of the values in a specified column.
- AVG - returns the average of the values in a specified column.
- MIN - returns the smallest value in a specified column.
- MAX - returns the largest value in a specified column.

The following paragraphs will explain each vendor’s limitations and effort regarding solutions for updating the view.

View in SQL Server

Views in MS SQL Server 2000 can be updateable in two ways. (CREATE VIEW [MS VIEW])

- **INSTEAD OF Triggers:** If an INSTEAD OF trigger exists for a view on a given data modification statement (INSERT, UPDATE, or DELETE), the corresponding view is updatable through that statement.

- **Partitioned Views:** If the view is of a specified form called 'partitioned view,' the view is updatable, subject to certain restrictions. When needed, SQL Server will distinguish **Local Partitioned Views** as the views in which all participating tables and the view are on the same SQL Server, and **Distributed Partitioned Views** as the views in which at least one of the tables in the view resides on a different (remote) server.

If a view does not have either an INSTEAD OF trigger or a partitioned view, then it is updatable only if the following conditions are satisfied:

- The SELECT statement has no aggregate functions in the select list and does not contain the TOP, GROUP BY, UNION (unless the view is a partitioned view), or DISTINCT clauses. Aggregate functions can be used in a subquery in the FROM clause as long as the values returned by the functions are not modified.

- The SELECT statement has no derived columns in the select list. Derived columns are result set columns formed by anything other than a simple column expression, such as using functions or addition or subtraction operators.

- The FROM clause in the SELECT statement references at least one table. The SELECT statement must have more than non-tabular expressions, which are expressions not derived from a table.

There are some limitations when we create the view with a SELECT clause in SQL Server. It is not about UPDATE the view, but we point out the limitations of creating a view. We cannot create a CREATE VIEW statement with: (MS VIEW)

- COMPUTE or COMPUTE BY clauses.

  COMPUTE clause produces multiple result sets. One type of result set contains the detail rows for each group containing the expressions from the select list (Summarizing Data Using COMPUTE and COMPUTE BY [MS COMPULTE]). The other type of result set contains the sub-aggregate for a group, or the total aggregate for the SELECT statement. The select list can contain expressions other than the grouping columns or aggregate functions. The aggregate functions are specified in the COMPUTE clause, not in the select list. The COMPUTE clause takes the following information:
  - The optional BY keyword, which calculates the specified row aggregate on a per column basis.
  - A row aggregate function name; for example, SUM, AVG, MIN, MAX, or COUNT.
  - A column to perform the row aggregate function upon.

- ORDER BY clause, unless there is also a TOP clause in the select list of the SELECT statement.

- The INTO keyword.

- Reference a temporary table or a table variable.

View in DB2

A view cannot use INSERT, UPDATE, or DELETE statements if the following points are included in the statements in DB2 (5.39 Create View (2001). DB2 UDB for OS/390 and z/OS V7 SQL Reference Document Number: SC26-9944-02 [D6];

- The first FROM clause identifies more than one table or view, or identifies a table function.

- The first SELECT clause specifies the keyword DISTINCT.

- The outer SELECT contains a GROUP BY clause.

- The outer SELECT contains a HAVING clause

- The first SELECT clause contains an aggregate (DB2 calls it a column function).

- It contains a subquery such that the base object of the outer SELECT, and of the subquery, is the same table.

- The first FROM clause identifies a view with these mentioned points.

INSTEAD OF TRIGGER can be used to perform a delete, insert, or update request on behalf of a view, which is not inherently updateable (p125, Chap2, Administration: Implementation v8). Applications taking advantage of this type of trigger are able to write update operations against views just as if the view were a table.

View in Oracle

In order to be inherently updatable, a view cannot contain any of the following constructs (OC, 2001, Chap. 11 Updatable Join View):

- A SET operator - Set operators combine the results of two component queries into a single result. Queries containing set operators are called compound queries. UNION, INTERSECT and MINUS operators are examples for set operators.

- A DISTINCT operator.

- An aggregate or analytic function.
    - PL/SQL lets you use all the SQL functions including the following aggregate functions, which summarize entire columns of Oracle data: AVG, COUNT, GROUPING, MAX, MIN, STDDEV, SUM, and VARIANCE. Except for COUNT(*), all aggregate functions ignore nulls.

- A GROUP BY, ORDER BY, CONNECT BY, or START WITH clause.
    - CONNECT BY – Specify a condition that identifies the relationship between parent rows and child rows of the hierarchy.
    - START WITH - Specify a condition that identifies the row(s) to be used as the root(s) of a hierarchical query. Oracle uses as root(s) all rows that satisfy this condition. If you omit this clause, then Oracle uses all rows in the table as root rows. The START WITH condition can contain a subquery, but it cannot contain a scalar subquery expression.

- A collection expression in a SELECT list.

- A subquery in a SELECT list.

- Joins (with some exceptions).

Views that are not updatable can be modified using INSTEAD OF triggers (OC, 2001, Chap. 17 Triggers). It provides a transparent way of modifying views that cannot be modified directly through DML statements (INSERT, UPDATE, and DELETE). You can write normal INSERT, UPDATE, and DELETE statements against the view and the INSTEAD OF trigger is fired to update the underlying tables appropriately. INSTEAD OF triggers are activated for each row of the view that gets modified.

In Oracle, there are *materialised views* that can summarise, compute with aggregated functions because they store summarised data (OC, 2001, Chap 11 Materialized view). The materialised view is used to replicate data at distributed sites and synchronized updates done at several sites with conflict resolution methods in distributed environments. In mobile computing environments, it is used to download a subset of data from central servers to mobile clients, with periodic refreshes from the central servers and propagation of updates by clients back to the central servers.

4.1.7. Result of R7. High-level insert, update, and delete

All of the vendors use *cursors* to fulfil this rule fully. Cursor is short for **cur**rent **s**et **o**f **r**ecords (Webopedia, Cursor) and the cursors are used to retrieve multiple rows from a result set that is returned by a SELECT statement. Date (2000, p.93) describes that the cursors provide to access the rows in the set one by one.

The following paragraphs describe each vendor's cursors in detail.

Cursor in SQL Server

Cursors in SQL Server support results as below. (Cursors [MS CURSOR])

- The cursors position at specific rows of the result set.
- The cursors retrieve one row or block of rows from the current position in the result set.
- The cursors support data modifications to the rows at the current position in the result set.
- The cursors support different levels of visibility to changes made by other users to the database data that is presented in the result set.
- The cursors provide Transact-SQL statements in scripts, stored procedures, and triggers

SQL Server provides the most different types of cursors among the vendors. Below is the SQL-92 syntax and T-SQL extended syntax which SQL Server uses to indicate how large the variety is.

SQL-92 Syntax

```
DECLARE cursor_name [ INSENSITIVE ] [ SCROLL ] CURSOR
```

Transact-SQL Extended Syntax

```
DECLARE cursor_name CURSOR
[ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
[ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
[ TYPE_WARNING ]
```

(DECLARE CURSORS [MS DECLARE CURSOR])

Cursor in DB2

Cursors in DB2 can be used to fetch, update, or delete a row of a table, but you cannot use them to insert a row into a table. Scrollable cursors allow random access to a result table. Applications can use a powerful set of SQL to fetch data using a cursor in random order. Scrollable cursors are especially useful for screen-based applications. You can specify that the data in the result table is to remain static, or you can perform the data updates dynamically. DB2's cursor declaration syntax is below. (2.3.4.2.3 Retrieving a set of rows (2001). DB2 UDB for OS/390 V7 An Introduction to DB2 for OS/390: 2.3.4.2.3 Document Number: SC26-9937-02)

```
DECLARE cursor_name CURSOR
[ INSENSITIVE | SENSITIVE]
[STATIC]
[SCROLL]
```

Cursor in Oracle

Execution of a cursor puts the results of the query into a set of rows called the result set, which can be fetched sequentially or nonsequentially. Scrollable cursors are cursors in which fetches and DML operations do not need to be forward sequential only. Interfaces exist to fetch previously fetched rows, to fetch the n^{th} row in the result set, and to fetch the n^{th} row from the current position in the result set. (OC, 2001, Chap 16 Scrollable Cursors)

4.1.8. Result of R8. Physical data independence rule

The vendors all support the eighth rule fully. Physical data management includes (Connolly et al., 1999, p.42):

- Storage space allocation for data and indexes
- Record descriptions for storage (with stored sizes for data items)
- Record placement
- Data compression and data encryption

We have interviewed a couple of staff members from each of the three respective groups (DB2, Oracle and SQL Server) at the ODS Department at Volvo IT, asking their opinion whether it is possible to change the index without altering the application program. In summary, all of the products can manage the points that we mentioned above.

4.1.9. Result of R9. Logical data independence rule

None of the vendors supports this rule completely. Rule number six, which is not fulfilled either by any of the vendors, influences also rule number nine, as explained below in more detail.

Logical limitation in SQL Server

As mentioned in section 4.1.6.1, there are some limitations when a view is used. Furthermore, if a view depends on a table (or view) that was dropped, SQL Server produces an error message if anyone tries to use the view (MS VIEW). If a new table (or view) is created, and the table structure does not change from the previous base table, to replace the one dropped, the view again becomes usable. If the new table (or view) structure changes, then the view must be dropped and recreated. There is a restructuring problem in the system.

Logical limitation in DB2

When a column is added to the base table after the view is created, the column does not include in the view. In other words, the changing of the logical schema effects the external schema which we have change separately. DB2 has not solved the growth aspect of the logical independence. (1.12 View. (2001). DB2 UDB for OS/390 and z/OS V7 SQL Reference Document Number: SC26-9944-02).

DB2 offers an option which can store the result of a query called a materialised query table. It is possible to use INSERT, UPDATE, and DELETE operations on the materialised query table, but there is no control against the underlying base tables and it is still the users' responsibilities to be aware of this. Inconsistencies may arise from changing the views. (DB2 UD Administration Guide: Implementation [DIM], 2002, p.137)

Logical limitation in Oracle

When 'SELECT * FROM table_name' is used in a view, we have to be aware that some invalid conditions can occur (OC, 2001, Chap 15 View and Base Tables).

- If a base table or view of a view is altered, renamed, or dropped, then the view is invalidated, but its definition remains in the data dictionary along with the privileges, synonyms, other objects, and other views that reference the invalid view. It means that changing the external view is necessary.

- All base tables referenced by the defining query of a view must exist. If a base table of a view is renamed or dropped, the view is invalidated and cannot be used. References to invalid views cause the referencing statement to fail. The view can be compiled only if the base table is renamed to its original name or the base table is re-created.

- The view cannot be validated if the base table is re-created with *new columns* and the view references columns no longer contained in the re-created table. It is especially relevant in the case of views defined with a `SELECT * FROM` *table* query, because the defining query is expanded at view creation time and permanently stored in the data dictionary. The growth problem arises.

Even though an INSTEAD OF TRIGGER is used to insert, delete, and update the view, the result of modifying the view can be ambiguous:

- Deleting a row in a view could either mean deleting it from the base table or updating some values so that it is no longer selected by the view.

- Inserting a row in a view could either mean inserting a new row into the base table or updating an existing row so that it is projected by the view.

- Updating a column in a view that involves JOINS might change the semantics of other columns that are not projected by the view.

4.1.10. Result of R10. Integrity independence rule

NOT NULL constraints are supported by all of the database vendors. There are no differences between them. (MS NULL; DPL, 2002, p.16; OC, 2001, Chap23) Likewise PRIMARY KEY constraints are fully supported as it is described in the section 4.1.2. Result of R2. 'The guaranteed access rule' (MS PK; DPL, 2002, p.16; OC, 2001, Chap. 23)

When UNIQUE constraints are concerned, there are subtle differences between the products. All of the vendors state that a unique constraint ensures that no duplicate values are allowed in specified columns on a non-primary key. Further, a composite UNIQUE KEY which consists of more than one column is applied by the all of the database systems. SQL Server and DB2 support multiple UNIQUE KEYS in a table, but not Oracle. (Creating a Unique Constraint [MS UNIQUE]; 1.6.1 Unique constraints. (2001). DB2 UDB for OS/390 and z/OS V7 SQL Reference: Document Number: SC26-9944-02) (OC, 2001, Chap 23)

Even though UNIQUE KEY constraints are generally used with NOT NULL constraints in a statement, SQL Server and Oracle allow having NULL values in UNIQUE constraints. However, DB2 states that it must be NOT NULL in it.

FOREIGN KEY constraints have some variations among the vendors, especially DML options in DELETE and UPDATE statement. The following table shows the summary of the result.

X = the option can be used. # = no information found.

DML Statement	DML option	SQL Server	DB2	Oracle
INSERT		#	X	X
DELETE	CASCADE	X	X	X
	NO ACTION	X	X	X
	RESTRICT		X	
	SET NULL		X	X
UPDATE	CASCADE	X		
	NO ACTION	X	X	X
	RESTRICT		X	

Table 3 – the table shows DML options for FOREIGN KEY constraints for each product.

There is no option regarding the INSERT statement. A non-null INSERT value of the foreign key must match some value of the parent key of the parent table. The value of a composite foreign key is null if any component of the value is null. This rule is implicit when a foreign key is specified.

The DELETE rule of a referential constraint applies when a row of the parent table is deleted. More precisely, the rule applies when a row of the parent table is the object of a delete or propagated delete operation (defined below), and that row has dependents in the child table of the referential constraint.

DELETE statement's options for FOREIGN KEY constraints

- CASCADE – when rows with parent key values are deleted, causing all rows in child tables with dependent foreign key values to also be deleted.

- NO ACTION – If any row in the child table does not have a corresponding parent key when the DELETE statement is completed, the deletion is rejected when the NO ACTION is used in the DELETE statement. NO ACTION means that a non-null delete value of a foreign key must match some value of the parent key of the parent table when the DELETE statement is completed.

- RESTRICT - If any row in the child table matches the original value of the key, the deletion is rejected when the RESTRICT option is applied.

- SET NULL – When rows containing parent key values are deleted, causing all rows in child tables with dependent foreign key values to set those values to null.

UPDATE statement's options for FOREIGN KEY constraints

- CASCADE – When rows containing parent key values are updated, causing all rows in child tables with dependent foreign key values to also be updated.

- NO ACTION - If any row in the child table does not have a corresponding parent key when the update statement is completed, the update is rejected when the update rule is NO ACTION. NO ACTION means that a non-null update value of a foreign key must match some value of the parent key of the parent table when the update statement is completed.

- RESTRICT - If any row in the child table matches the original value of the key, the update is rejected when the update rule is RESTRICT.

(Create Table [MS CREATE TABLE]; Table relationships Visual Database Tools for SQL Server [MS TABLE RELATION]; DPL, 2002, p.81; OC, 2001, Chap.23)

All of the database vendors have CHECK constraints. A table in all of the database systems can have an arbitrary number of CHECK constraints and a check constraint can be defined using the CREATE TABLE or the ALTER TABLE statement. A CHECK constraint must be a Boolean expression evaluated using the values in the row being inserted or updated and cannot contain subqueries which are SELECT statements within the WHERE or HAVING clause of another SQL statement or nested SQL statements. (1.6.3 Check constraints. (2001). DB2 UDB for OS/390 and z/OS V7 SQL Reference Document Number: SC26-9944-02 [D10]; DPL, 2002, p.84; OC, 2001, Chap 23; Check Constraints [MS CHECK])

When one or more table CHECK constraints is defined in the ALTER TABLE statement for a table with existing data, the existing data is checked against the new condition before the ALTER TABLE statement completes.

4.1.11. Result of R11. Distribution independence rule

All of the vendors support two and three architectures for distributed database (Three-tier Application Model [MS THREE-TIER]; 3.1.1 Architectural characteristics of Web applications. (2001). DB2 UDB for OS/390 V7 An Introduction to DB2 for OS/390 Document Number: SC26-9937-02; OC, 2001, Chap2).

A two-tier architecture is known as Client/Server. A client application takes responsibility about business roles and user interface and a database server performs the data retrieval and manipulation. A three-tier architecture uses an application server between a client application and a database server. The application server serves the purpose to check that all of the business processing is done correctly. The three-tier architecture allows modifications without having to change the other two parts.

(5.7.4.1.1 Definition. (2001). DB2 UDB for OS/390 and z/OS V7 Administration Guide Document Number: SC26-9931-02 [D11]; Understanding Locking in SQL Server [MS LOCKING]; 5.7.4.3.3 Modes of table, partition, and table space locks. (2001). DB2 UDB for OS/390 and z/OS V7 Administration Guide: 5.7.4.3.3 Document Number: SC26-9931-02 [D12])

A locking granularity shows the size of data items.

Locking Granularity	**SQL Server**	**DB2**	**Oracle**
Row	X	X	X
Key (row lock within an index)	X		
Page/block	X	X	
Extent	X		
Segmented Table		X	
Entire Table	X	X	X
Table space		X	
DB	X		

Table 4 – the table shows the locking granularity for each database product.

Description of Locking Granularity

- Row - a single row within a table.
- Key – a row lock within an index. Used to protect key ranges in serialisable transactions.
- Page /block – In SQL Server 8 kilobyte (K) data page or index page. In DB2, a page size is a choice of 4K, 8K, 16K or 32K (Adamache, DB2 Storage Trivia).
- Extent - contiguous groups of eight data pages or index pages.
- Segmented table (space?) - A table space that is divided into equally sized groups of pages called segments. Segments are assigned to tables so that rows of different tables are never stored in the same segment.
- Entire table - Entire table, including all data and indexes.
- Table space - A page set that is used to store the records in one or more tables.
- DB- Entire database.

A lock mode means what access to the locked object is permitted to the lock owner and to any concurrent processes (D12, 2001).

Lock Mode	**SQL Server**	**DB2**	**Oracle**
Shared (S)	S	S	S
Update (U)	U	U	
Exclusive (X)	X	X	X
Intent shared (IS) / Row share table locks (RS)	IS	IS	RS
Intent exclusive (IX)/ Row exclusive table locks (RX)	IX	IX	RX
Shared with intent exclusive (SIX)/ Share row exclusive table locks	SIX	SIX	SRX

Table 5 – the table shows the lock mode for each database product.

Description of Lock Mode

Shared Locks - Used for operations that do not change or update data (read-only operations), such as a SELECT statement. No other transactions can modify the data while shared (S) locks exist on the resource.

- Update Locks - Used on resources that can be updated. Prevents a common form of deadlock that occurs when multiple sessions are reading, locking, and potentially updating resources later. To avoid this potential deadlock problem, update (U) locks are used. Only one transaction can obtain an update (U) lock to a resource at a time. If a transaction modifies a resource, the update (U) lock is converted to an exclusive (X) lock. Otherwise, the lock is converted to a shared-mode lock.

- Exclusive Locks - Used for data-modification operations, such as INSERT, UPDATE, or DELETE. Ensures that multiple updates cannot be made to the same resource at the same time.

Intent Locks

When a page or row is locked, the table, partition, or table space containing it is also locked. In that case, the table, partition, or table space lock has one of the intent modes: IS, IX, or SIX. The modes S, U, and X of table, partition, and table space locks are sometimes called *gross* modes. In the context of reading, SIX is a gross mode lock because you don't get page or row locks; in this sense, it is like an S lock.

An intent lock indicates that SQL Server wants to acquire a shared (S) lock or exclusive (X) lock on some of the resources lower down in the hierarchy. For example, a shared intent lock placed at the table level means that a transaction intends on placing shared (S) locks on pages or rows within that table. Setting an intent lock at the table level prevents another transaction from subsequently acquiring an exclusive (X) lock on the table containing that page. Intent locks improve performance because SQL Server examines intent locks only at the table level to determine if a transaction can safely acquire a lock on that table. This removes the requirement to examine every row or page lock on the table to determine if a transaction can lock the entire table.

- Intent shared (IS) - Indicates the intention of a transaction to read some (but not all) resources lower in the hierarchy by placing S locks on those individual resources. The lock owner can read data in the table, partition, or table space, but not change it. Concurrent processes can both read and change the data. The lock owner might acquire a page or row lock on any data it reads. The Oracle function Row Share Table Locks (RS) appears to serve the same function as Intent Shared Locks in DB2 and SQL Server.
- Intent exclusive (IX) - The lock owner and concurrent processes can read and change data in the table, partition, or table space. The lock owner might acquire a page or row lock on any data it reads; it must acquire one on any data it changes.
- Shared with intent exclusive (SIX) - Indicates the intention of the transaction to read all of the resources lower in the hierarchy and modify some (but not all) resources lower in the hierarchy by placing IX locks on those individual resources. Concurrent IS locks at the top-level resource are allowed. For example, an SIX lock on a table places an SIX lock on the table (allowing concurrent IS locks), and IX locks on the pages being modified (and X locks on the modified rows). There can be only one SIX lock per resource at one time, preventing updates to the resource made by other transactions, although other transactions can read resources lower in the hierarchy by obtaining IS locks at the table level.

Oracle does not use Update lock which the other vendors have, but solves this another way called Deadlock Detection. (OC, 2001, Chap 22) It finds deadlock situations automatically and resolves them by rolling back one of the statements involved in the deadlock, thereby releasing one set of the conflicting row locks.

4.1.12. Result of R12. Non-subversion rule

There are administration tools, which help to manipulate the system catalog, but the base of the tools uses also SQL–statements, so actually we cannot manage the catalog without SQL-statements. Oracle does not allow the user to manipulate the system catalog. They are read-only. (DPL, 2002, p.7; OC, 2001, Glossary-5)

5. Discussion

The Relational Data Model is the foundation used to compare the three DBMS in this thesis. In summary, evaluated using Codd's twelve rules, none of the DBMSs can be said to be more 'relational' than the others. All of the products fulfil the requirements of Codd's rules quite well, and we can say that nine rules out of twelve are fully supported.

		SQL Server	DB2	Oracle
R1	The information rule	X	X	X
R2	The guaranteed access rule	X	X	X
R3	Systematic treatment of null values	Partially	Partially	Partially
R4	Active online catalog based on the relational model	X	X	X
R5	The comprehensive data sub-language rule	X	X	X
R6	The view updating rule	Partially	Partially	Partially
R7	High-level insert, update, and delete	X	X	X
R8	Physical data independence rule	X	X	X
R9	Logical data independence rule	Partially	Partially	Partially
R10	Integrity independence rule	X	X	X
R11	Distribution independence rule (locking)	X	X	X
R12	Non-subversion rule	X	X	X

Table 4 - the table shows the results of this study.
(X = fully supported, Partially = partially supported)

Another way to depict the results is to use Connolly's five rules, which are based on Codd's twelve rules (1999, p.103). They are:

(a) Foundational rules (R12)

(b) Structural rules (R1, R6)

(c) Integrity rules (R3, R10)

(d) Data manipulation rules (R2, R4, R5, R7)

(e) Data independence rules (R8, R9, R11)

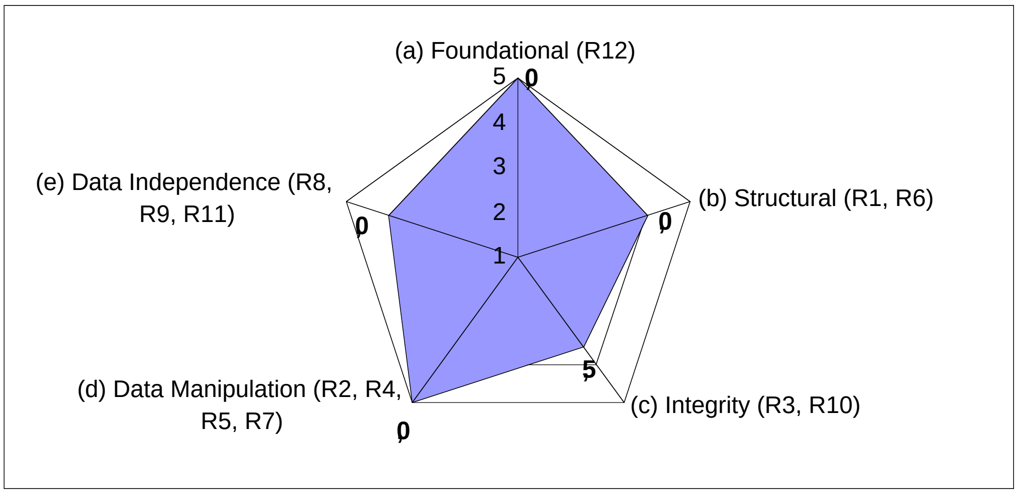

Fig. 2. The graph shows the results after applying Connolly's 5 rules to the three DBMS. All three systems produced the same score, so the single graph is representative of any one of them.

The scores between 1 and 5 demonstrated in fig. 2 are grades of achievements in each category. All DBMS got the same score using these five rules. The fifth grade in the chart shows that the category is supported fully by the vendor in question and the first grade means it is not supported at all. The numbers themselves have no quantitative meaning, they merely serve as a qualitative estimate based on how well they fulfil Codd's rules. There is not a radar chart or each DBMS, because their differences are very subtle and the levels are quite similar. The next paragraphs look through each category in detail.

(a) Foundational Rules (R12)

In the radar chart, this rule is given five full points because R12 is supported fully by all the vendors. The twelfth rule has some connection to first rule and the data manipulation rules. Because all of the information in the database is managed in tabular forms, it is possible to fulfil the twelfth rule. (DPL, 2002, p.7; OC, 2001, Glossary-5, MS Glossary) All of the database products implement PRIMARY constraints, well-developed sub-languages which provide services such as automated data definition operations, data manipulation, security and integrity constraints, and transaction management.

A violation of the twelfth rule can cause unnecessary problems. If a user, for example, changes information directly without using SQL statements, the integrity constraints can be violated. It is essential for the vendors to follow this rule fully because of that risk.

(b) Structural Rules (R1, R6)

R1 is the very a fundamental rule in Relational Data Model. In 1982, Codd published a paper, "Relational Database: A Practical Foundation for Productivity." (CACM). He proposed two minimal requirements for a relational database and pointed out the importance of the information rule at the time. Because the first rule is basic and important, it is necessary for all of the vendors to support this rule fully.

On the other hand, none of the products has succeeded to fulfil the View Update rule. One of the reasons for the restrictions of update views is not to violate some integrity constraints (p312, Date, 2000). Because a view is a virtual table, the result of which is not stored in the database, it can cause some ambiguity in the condition. Johnson (p217, 1997) gives a simple example of the ambiguity of updating a view from one base table without a primary key. A view is allowed to create without the presence of the primary key. However, if we want to delete specific tuples using SELECT statement with a WHERE clause from the view, the obscurity occurs.

Suppose that we have two base tables, "Student" and "Contact information".

Student (a base table)

Sno	Fname	Lname	Cno
1	Charlie	Orange	11
2	Black	White	22
3	Killer	White	33
4	Charlie	Orange	44

Attributes:

Sno	Student id (primary key)
Fname	First name
Lname	Last name
Cno	Contact information id (foreign key)

Contact information (a base table)

Cno	Home tel	Mobile tel	e-mail
11	111-1111	010-111-1111	Charlie@…
22	111-2222	010-111-2222	Flipper@…
33	111-3333	010-111-3333	Killer@…
44	111-4444	010-111-4444	Charlie_o@.

Attributes:

Cno	Contact information id (primary key)
Home tel	Home telephone number
Mobile tel	Mobile telephone number
E-mail	E-mail address

We create a view for Lname, based on Student table and the statement is:

```
CREATE VIEW Lastname_view AS
```

```
                    (SELECT Lname FROM Student);
```

The lastname_view table looks like this.

Lastname_view (a view)

Lname
Orange
White

As you see, there is no primary key in the view, lastname_view. Then, suppose that we want to delete the attribute Sno 1 from the Student table via the Lastname_view, but it is not safe to delete Sno 1 because there is no primary key in the view and there are two attributes Lname which are called ‘Orange’ in the base table, and it is not sure what the result would be. Johnson says that some products take the safe route and allow no data modifications through views. The three database products studied in this thesis all allow modifying tuples in the view under some restrictions, but they cannot support the update view issue completely.

As the updating view is only supported partially, it violates the principle of interchangeability and the principle of database relativity (section 3.2.6 in this paper). If an organisation uses such a DBMS, the end users in the organisation have more restrictions to use the views than the base tables, or the database administrators or the programmers in the organisation have to handle the problems so that the end users can use both the views and the base tables in the same manner. This creates an inflexible system and extra work for the administrator.

Elmasri and Navathe (2000, p.280) state that modification from a view can cause complex queries which are time-consuming to execute. Some vendors try to solve the problem using so called **view materialisation**, including DB2 and Oracle (OC, 2001, Chap. 11; DIM, 2002, Chap. 2). The view materialization in Oracle is more developed than in DB2. DB2’s materialization of the view is still primitive and the user has to take responsibility for the integrity of data while Oracle provides almost the same quality of services as the base table. SQL Server does not have a view materialization system. See also Data Independence Rules below.

In the radar chart, the structural rules (b) are given four points because all the vendors support R1 fully, but R6 is supported only partially by them.

(c) Integrity Rules (R3, R10)

Null value rules are supported partially by all the vendors in a similar way. There is often missing information in the real world and data from the database systems, which reflect the part of the real world, may also be missing. It is important to manage the missing values in the system because the missing values affect, for example, the result of average from a column. The vendors try to solve this problem using the UNKNOWN expression, but that is not enough to fulfil the integrity rules perfectly. Codd proposes that DBMSs should use four-valued logic (4VL). However, Date (2000, p.585) has argued against Codd's idea, because the complexity of the Boolean logic increases and it become difficult to implement. Null values in a table can cause unexpected results (Elmasri and Navatye, 2000, p.511). When a NATURAL JOIN operation is used to retrieve or modify specific data in a view or table, tuples with null values are not included in the result table. The NULL values have distinctive features against the other values. Therefore, the users, especially, the database administrators in the organisation have to be aware of how NULL values affect the results of the calculations.

Date (2000, p.250) points out that few products provide declarative integrity support and complains that some of the database vendors give advice to use stored or triggered procedures where the users have to take responsibility for the constraints. Date says that it is impossible to create 100 % potential constraints on the vendors' side, but recommends that the DBMS should provide declarative integrity support so that as much as 90 % of a typical database definition would consist of constraints. It helps to relieve application programmers of a considerable burden and would allow them to become significantly more productive. Of course, it is important for the vendors to provide the stored or triggered procedure, but especially, triggers are more powerful than constraints and there is a danger in violating the important rules (DPL, p.18).

As Date says, the lack of support for integrity rules will become a burden for the organisation because the database administrators have to make the rules and take care of them by themselves. Violations of integrity are easily caused, making it difficult to manage the integrity rules.

The FOREIGN KEY constraints among the five constraints indicate differences among the database products DML options regarding the DML statements for each database product. DB2 has more options than the others, so DB2 can be said to support the FOREIGN KEY constraints more fully than the others.

In the radar chart, we give three and a half points for the integrity rules (c). One of the reasons is that Codd recommends 4VL support instead of 3VL, and null values can cause inconsistencies in the tables. The other reason is that none of the vendors support the integrity independence rules as much as Date would wish.

(d) Data Manipulation Rules (R2, R4, R5, R7)

Data manipulation rules are supported fully by all of the vendors. In general, the data manipulation is the very core of the database systems. Because the first rule (nothing but table) is strictly complied with, the requirements of rules two, four, and five are easy to achieve. As we described in section 3.2.2, storing meta-data is one of the advantages of the DBMS. The tabular forms which are used in relational data models facilitate the management of the meta-data. We assume that the database vendors are aware of the importance of the development of this aspect. The result of these rules is five points in the radar chart, because all the vendors support them fully.

(e) Data Independence Rules (R8, R9, R11)

Date (p35, 2000) explains that the conceptual and external levels are relational while the physical level is not. ‘Relational’ means here that objects visible on a particular level are relational tables and the operators will be relational operators, including restrict (select) and project operators, and the information on the external and conceptual levels are controlled only on the relational level. From the interview, a physical change on the internal level does not necessarily have to influence a user’s application program.

Oracle has come closest to solving this because their materialized views are supported without many limitations (OC, 2001, Chap. 11). DB2 needs to solve the consistency between the view and the underlying base table (DIM, 2002, p.137). Presumably, because SQL Server has more limited storage space than the other products and the view materialization consumes much storage space, they do not use the view materialization technique.

The lack of the R9 causes a need of changing the application programs if the logical structure is changed. The more end users in the organisation, the more the updating tasks increase.

We can see differences in the fineness of locking granularity among the vendors. Date states that the finer the granularity, the greater the concurrency; the coarser, the fewer locks need to be set and tested and the lower the overhead (p486, Date, 2000). SQL Server and DB2 use a finer granularity than Oracle’s (MS LOCKING; 5.7.4.1.3 General effects of size (2001)**.** DB2 UDB for OS/390 and z/OS V7 Administration Guide Document Number: SC26-9931-02)
Oracle has only two grades of locking granularities and the finest is a row size. The reason for this is that Oracle avoids deadlock problems rather than increases the concurrency (22-20 Oracle*9i* Database Concepts). One of the positive effects when using coarse granularity is that deadlocks occur infrequently in Oracle. A possible disadvantage would be that fewer users can work simultaneously.

As Date says, the locking size issues have advantages and disadvantages, one way or another. A fine locking granularity means that more users can be active simultaneously, but deadlock problems occur more often. Conversely, a coarse locking granularity can avoid the deadlocks better than the fine one, but fewer users can be accommodated. It is difficult to say which type is better than the other, so both are assigned the same score.

In the radar chart, four points are given for these rules. One point reduction because changing in a base table causes inconsistency between the conceptual level and the external levels and the referencing view. This means that application programs have to be changed manually.

6. Conclusions

From the viewpoint of the relational database model, we have tried to find differences between the three database products DB2, Oracle and SQL Server. The results show that all the products are quite similar. They have similar problems with update view and the management of null values. However, Oracle seemed to have come closer to solving the update view issue using the so-called materialized views, which are integrated in the base table. DB2 can also support the update view fully, if the integrity between a view and its base table can be maintained. We suppose SQL Server needs to solve the view problem in a different way because of the more limited physical capacities. Furthermore, DB2 has the advantage of providing more options regarding FOREIGN KEY constraints. In summary, all three products can be considered relational, even though they do not completely fulfil all Codd's 12 rules.

There is no RDBMS that supports the rules update views (R6), management of null values (R3), and logical data independence (R9) fully. Violations of the rules cause unnecessary additional work for the end users as well as for the administrators. For example, the end users cannot update a view, based on two base tables, or get wrong calculation because the DBMS manages null values in an unexpected way. As a new column is created in a table, every application program that uses the table also has to be changed. Data inconsistency can occur easily because a view and its base table have to be controlled by the users separately. It is important for the users to fulfil the rules to get rid of such troublesome tasks.

Method Evaluation

In this thesis, examination of the DBMS vendors' documentation is used as the main method, combined with some interviews with experienced DBMS users at Volvo IT. Because all the database vendors provide detailed and organised documentation, and the experienced DBMS users were very helpful, the main questions of the study could be answered.

An alternative method could have been used. For example, we could have chosen some specific questions for each of Codd's rules, run the three different DBMSs and check the results. Documentation, the method used for this thesis, is a more general method, whereas testing specific questions would be more limited when it comes to investigating Codd's rules because it is impossible to test all aspects. Alternatively, both documentation and testing could have been used. It would be interesting to see if this would yield the same or different results, but this type of comparative study would have required much more time.

Future Research

We can develop this study to cover three new questions. The first one is to review R11 completely. We state that R11, the “distribution independence rule” is supported by all the vendors. However, we could only review one of Date’s 12 criteria for distributed DBMS, because the distributed database management systems cover huge areas. Penetrating all twelve criteria might yield a different result. Secondly, we can study different algorithms for the view update more closely to understand where the difficulties lie, and design strategies for improvement. The third and last future research topic would be the effects of the null values. It would be interesting to compare how each DBMS manages calculations using null values.

7. References

Literature References

Backman, J. (1998). *Rapporter och uppsatser*. Lund: Studentlitteratur.

Connolly, T., Begg, C., Strachan, A. (1999). *Databababase Systems: A Practical Approach to Design, Implementation, and Management* (2nd ed.). Harlow: Addison-Wesley.

Date, C.J. (1991). *An Introduction to Database Systems* (5th ed.). CA: Addison-Wesley.

Date, C.J. (2000). *An Introduction to Database Systems* (7th ed.). CA: Addison-Wesley.

[DIM] IBM DB2 Universal Database. (2002). Administration Guide: Implementation

[DPER] IBM DB2 Universal Database. (2002). Administration Guide: Performance version 8.

[DPL] IBM DB2 Universal Database. (2002). Administration Guide: Planning version 8.

[DSQL1] IBM DB2 Universal Database. (2002). SQL Reference Volume 1 version 8.

Elmasri, R., Navathe, S.B. (2000). *Fundamentals of Database Systems* (3rd ed.). CA: Addison-Wesley

Johnson, J.L. (1997). *Database: Model, Languages, Design*. NY: Oxford University Press.

Melnyk, R.B., Zikopoulos, P.C. (2001). *DB2: The Complete Reference*. CA: McGraw-Hill

Oracle. (2001, July). Oracle*9i* Database Concepts Release 1 (9.0.1). Part No. A88856-02

Oracle. (2002). Oracle*9i* SQL Reference Release 2 (9.2). Part No. A96540-1

Robson, W. (1997). *Strategic Management & Information Systems (2nd ed)*. Pitman.

Internet references

Adamache, B., DB2 Storage Trivia.
http://www7b.software.ibm.com/dmdd/library/techarticle/adamache/0621_adamache.html.

Avery, B. *the Relational Model*, Kingston University, [PDF document] URL http://www.kingston.ac.uk/~ku12492/MBIT/model.pdf.

Boss Consulting, http://www.bossconsulting.com/sybase_dba/sublevels/rdbms.marketshare.

Brown, C.E. *The Relational Model*, Database learning module
http://www2.bus.orst.edu/faculty/brownc/lectures/db_tutor/relational_model.htm#3.1%20Relational%20Data%20Model%20Concepts.

Bull, M., (2002, Oct.). *Codd's Rules for RDBMS*, MB-online publication. West Yorkshire, URL http://hometown.aol.com/mbaddenda/art120.html.

[D1] 2.22. DB2 Search condition. *IBM DB2,* DB2 UDB for OS/390 and z/OS V7 SQL Reference. Document Number: SC26-9944-02. (2001).
http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/dsnsqh12/2.22?ACTION=MATCHES&REQUEST=null+unknown&TYPE=FUZZY&SHELF=dsnshh11&DT=20020827055824&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT.

[D2] 2.8. Data types, *IBM DB2*, DB2 UDB for OS/390 and z/OS V7 SQL Reference COVER Document Number: SC26-9944-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/dsnsqh12/2.8?ACTION=MATCHES&REQUEST=null+value&TYPE=FUZZY&SHELF=dsnshh11&DT=20020827055824&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT.

[D3] 3.1.2. COUNT, *IBM DB2*, DB2 UDB for OS/390 and z/OS V7 SQL Reference COVER Document Number: SC26-9944-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/DSNSQH12/CCONTENTS?SHELF=dsnshh11&DN=SC26-9944-02&DT=20020827055824.

[D4] 3.1.3. COUNT_BIG, *IBM DB2*, DB2 UDB for OS/390 and z/OS V7 SQL Reference Document Number: SC26-9944-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/DSNSQH12/CCONTENTS?SHELF=dsnshh11&DN=SC26-9944-02&DT=20020827055824.

[D5] 3.2.75. ROWID, *IBM DB2*, DB2 UDB for OS/390 and z/OS V7 SQL Reference Document Number: SC26-9944-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/DSNSQH12/CCONTENTS?SHELF=dsnshh11&DN=SC26-9944-02&DT=20020827055824.

[D6] 5.39. Create View, *IBM DB2*, DB2 UDB for OS/390 and z/OS V7 SQL Reference Document Number: SC26-9944-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/dsnsqh12/5.39?SHELF=dsnshh11&DT=20020827055824#HDRRCVIE.

[D7] 2.3.4.2.3 Retrieving a set of rows. *IBM DB2,* DB2 UDB for OS/390 and z/OS V7 An Introduction to DB2 for OS/390: 2.3.4.2.3 Document Number: SC26-9937-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/dsnith12/2.3.4.2.3?ACTION=MATCHES&REQUEST=retrieve+rows+cursor&TYPE=FUZZY&SHELF=dsnshh11&DT=20020826113947&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT.

[D8] 1.12 View. *IBM DB2,* DB2 UDB for OS/390 and z/OS V7 SQL Reference: Document Number: SC26-9944-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/dsnsqh12/1.12?ACTION=MATCHES&REQUEST=view&TYPE=FUZZY&SHELF=dsnshh11&DT=20020827055824&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT.

[D9] 1.6.1 Unique constraints. *IBM DB2,* DB2 UDB for OS/390 and z/OS V7 SQL Reference: Document Number: SC26-9944-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/dsnsqh12/1.6.1?SHELF=dsnshh11&DT=20020827055824.

[D10] 1.6.3 Check constraints. *IBM DB2,* DB2 UDB for OS/390 and z/OS V7 SQL Reference: Document Number: SC26-9944-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/dsnsqh12/1.6.3?ACTION=MATCHES&REQUEST=check+constraint&TYPE=FUZZY&SHELF=dsnshh11&DT=20020827055824&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT.

[D11] 3.1.1 Architectural characteristics of Web applications. (2001). *IBM DB2,* DB2 UDB for OS/390 and z/OS V7 An Introduction to DB2 for OS/390: 2.3.4.2.3 Document Number: SC26-9937-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/dsnith12/3.1.1?ACTION=MATCHES&REQUEST=three+tier+architecture&TYPE=FUZZY&SHELF=&DT=20020826113947&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT.

[D12] 5.7.4.3.3 Modes of table, partition, and table space locks. *IBM DB2,* DB2 UDB for OS/390 and z/OS V7 Administration Guide Document Number: SC26-9931-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/dsnagh12/5.7.4.3.3?ACTION=MATCHES&REQUEST=lock+mode&TYPE=FUZZY&SHELF=dsnshh11&DT=20020823131906&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT.

[D13] 5.7.4.1.3 General effects of size. *IBM DB2,* DB2 UDB for OS/390 and z/OS V7 Administration Guide Document Number: SC26-9931-02. (2001). http://publib.boulder.ibm.com/cgi-bin/bookmgr/FRAMESET/dsnagh12/5.7.4.1.3?ACTION=MATCHES&REQUEST=General+effects+of+size&TYPE=FUZZY&SHELF=dsnshh11&DT=20020823131906&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT.

[D14] DB2 Technical Support. http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v8pubs.d2w/en_main

Moore, F.L. *Codd's twelve rules. SMU School of Engineering.* http://engr.smu.edu/~fmoore/notes/12rules.htm.

[MS CHECK] *MSDN Library, Microsoft SQL Server.* Check Constraints. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vdtsql/dvovrcheckconstraints.asp

[MS COMPUTE] *MSDN Library, Microsoft SQL Server,* Advanced Query Concepts, *Summarizing Data.* Summarizing Data Using COMPUTE and COMPUTE BY. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/acdata/ac_8_qd_08_4ego.asp.

[MS CREATE TABLE] *MSDN Library, Microsoft SQL Server*, Create Table, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sqlce/htm/_lce_create_table.asp.

[MS CURSOR] *MSDN Library, Microsoft SQL Server.* Cursors. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/acdata/ac_8_con_07_5lpz.asp.

[MS DECLARE CURSOR] *MSDN Library, Microsoft SQL Server.* DECLARE CURSOR. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_de-dz_31yq.asp.

[MS Glossary] *MSDN Library, Microsoft SQL Server*. Glossary. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/glosssql/glossary_frame.asp.

[MS LOCKING]] *MSDN Library, Microsoft SQL Server.* Understanding Locking in SQL Server. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/acdata/ac_8_con_7a_7xde.asp.

[MS NULL] *MSDN Library, Microsoft SQL Server*, Tr*ansact-SQL syntax Elements Experessions*. NULL Values. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/acdata/ac_8_qd_02_8pwy.asp.

[MS PK] *MSDN Library*, *Microsoft SQL Server*. Primary key. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/createdb/cm_8_des_04_888k.asp.

[MS TABLE RELATION] *MSDN Library*, *Microsoft SQL Server.* Table Relationships Visual Database Tools for SQL Server (SQL Server). http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vdtsql/dvcontablerelationships.asp.

[MS THREE-TIER] *MSDN Library, Microsoft SQL Serve.* Three-tier Application Model. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnproasp2/html/threetierapplicationmodel.asp.

[MS T-SQL] *MSDN Library, Microsoft SQL Server, Advanced Query Concepts*. Transact-SQL Tips. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/acdata/ac_8_con_05_6e2b.asp.

[MS UNIQUE] *MSDN Library, Microsoft SQL Server.* SQL Server Creating a Unique Constraint. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vdtsql/dvhowcreatinguniqueconstraint.asp.

[MS VIEW] *MSDN Library, Microsoft SQL Server.* CREATE VIEW. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_create2_30hj.asp.

Mullins, C.S. (2003, May). *Complexity complicates database upgrades and migrations* The DBA Corner, Database Trends and Applications, http://www.craigsmullins.com/dbta_020.htm

Oracle 9i Database List of Books (Release 2 (9.2)). (2003). http://otn.oracle.com/pls/db92/db92.docindex?remark=homepage

Parkhurst, T. (2002, Feb. 9). *Codd's 12 rules*. *DATA MANAGEMENT STRATEGIES*, http://www.itworld.com/nl/db_mgr/09022002/pf_index.html

Rennhackkamp, M. (1996 June). *Relational Integrity Control*, DBMS online Server side. http://www.dbmsmag.com/9606d17.html.

searchCIO.com Definitions, *Migration* http://searchcio.techtarget.com/sDefinition/0,,sid19_gci214624,00.html.

Wallace, J., Cox, E. (2002, April 17). *Database Management Systems and GIS*. http://www.geocities.com/emmacoxca/project.htm#_Toc6854073/.

Webopedia, *Referential Integrity*. http://www.webopedia.com/TERM/r/referential_integrity.html.

Webopedia, *Cursor*. http://www.webopedia.com/TERM/c/cursor.html.