# Object - Relational Database

"Third Manifesto" Criticism to SQL and Object Oriented Database System

## Table of Contents

Abstract .................................................................................................................................................... 2

Introduction ............................................................................................................................................ 3

### Part I
Relational Database & Object Oriented Database ........................................................................... 3
1.1. Relational Database System ......................................................................................................... 3
1.2. Object Oriented Database System ................................................................................................. 6

### Part II
"Third Manifesto" Revisited : A Criticism to SQL & OO Database Systems... 10
3.1. "Third Manifesto" – at a Glance ..................................................................................................... 10
3.2. Criticism to SQL (Relational Database) ........................................................................................ 11
3.3. Criticism to Object Oriented Database .......................................................................................... 12
3.4. "Third Manifesto" : Object – Relational Database .......................................................................... 12

### Part III
Further Discussion on SQL 3 and Object Relational Database ......................................................... 14
3.1. SQL ................................................................................................................................................. 14
3.2. Object-Relational DBMS ................................................................................................................. 15

### Part IV
Database Systems Nowadays : Where To Go? .................................................................................. 16

Conclusion ............................................................................................................................................... 18

References ............................................................................................................................................ 19
ABSTRACT

Relational database systems became a revolution in database management systems in the 1980s, and then respectively followed by the object oriented database systems in the 1990s. Those systems have complementary strengths. Relational database management systems (RDBMS) are good for managing large amounts of data; object oriented database management systems (OODBMS) are good at expressing complex relationships among objects. In one hand, RDBMS are good for data retrieval but provide little support for data manipulation. In the other hand OODBMS are excellent at data manipulation but provide little or no support for data persistence and retrieval. The main idea behind the RDBMS and OODBMS are the relational model and the object model, which are both fundamentally different.

Relational database systems are based on two-dimensional tables in which each item appears as a row. Relationships among the data are expressed by comparing the values stored in these tables. Languages like SQL allow tables to be combined to express relationships among the data. The object model is based on the tight integration of code and data, flexible data types, hierarchical relationships among data types, and references.

“The Third Manifesto” by Darwen and Date is trying to combine the two (relational and object database) in order to manage large amounts of data with complex relationships. It has criticised and drawn postulates to reject SQL that pretends to manifest RDBMS and object-oriented which wasted the ideals of the relational model. Oppositely, it attempts to join together the object-oriented and relational database by returning to the bosom of the pure relational model and Codd’s 12 rules.

Keywords:
Relational Database Management System (RDBMS), Object Oriented Database Management Systems (OODBMS), relation model, object model, Structured Query Language (SQL), database, data, relation, object, logical difference
INTRODUCTION

The First Manifesto of databases was the Object Oriented Database System Manifesto which attempts to define an object oriented database system as opposed to extending the relational model (Atkinson 1989). The Second Manifesto was The Third Generation Database System Manifesto also supports the introduction of a number of object oriented features, such as inheritance, support for complex objects and an extensible type system. However, it argues that these features should be incorporated on top of the existing relational model (Stonebraker 1991).

Respectively, the Third Manifesto was The Foundation of Object-Relational Database that also supports the retention of the relational model, but strongly opposes the retention of SQL and argues that it is a ‘perversion’ of the relational model (Darwen 1993 in Darwen 2000). In addition it is also believed that no change to the current relational model is required in order to support object oriented features.

This paper aims to have a deeper discussion in the progression of database systems concepts based on those manifestos and the future model of them.

Systematically, this paper comprises of four parts. Firstly, there will be a Technical Overview on Relational Database and Object Oriented Database in the first part. Then The “Third Manifesto” Revisited - Criticism to SQL and Object Oriented will follow in the second part. In the third part, the further discussions on SQL3 and Object Database will be presented in which it will lead to the last part of Database Nowadays. In addition, a conclusion will conclude and summarise this paper.

The sources used in this paper are mostly from the Internet that provides the newest reference. However, a very limited number of classic books on Fundamentals of Database Systems and Foundation of Object Oriented Database Systems are also referred to give a broader and deeper background to this paper. The slide-copies kindly given by Mr. Hugh Darwen are also where this paper refers.
1.1. RELATIONAL DATABASE SYSTEM
The relational database model was firstly created by E. F. Codd, a researcher in IBM in 1969. The model is based on set theory and predicate logic, a branch of mathematic. The basic idea of the relational model is that a database comprises of a series of unordered tables (or relations) that can be manipulated using non-procedural operations that return tables (Litwin, 2000).

The term “relational” has its roots in the terminology to define the relational model. The table was referred to as a relation (a related set of information). In fact in the relational database it is widely used the terms relations, attributes and tuples where most of people use the common terms as tables, columns and rows respectively\(^1\). The relational model can be applied to both databases and database management systems (DBMS) themselves.

1.1.1. Relational Database Design
There are numerous benefits of a relational database model. Some of them are:

- Data entry, updates and deletions will be efficient.
- Data retrieval, summarization and reporting will also be efficient.
- Since the database follows a well-formulated model, it behaves predictably.
- Since much of the information is stored in the database rather than in the application, the database is somewhat self-documenting.
- Changes to the database schema are easy to make.

The relational database design involves SQL (Structured Query Language) that is used to communicate with a database\(^2\). SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. SQL has the standard commands such as “Select”, “Insert”, “Update”, “Delete”, “Create”, and “Drop” that can be used to accomplish almost everything that one needs to do with a database.

1.1.2. Key concepts and terms in Relational Database Design
There are several key concepts and terms in RDB which will be briefly overviewed.

a. Tables, Uniqueness and Keys
Tables in the relational model are used to represent “things” in the real world. Each table should represent only one thing. These things (or entities) can be real-world objects or events. Tables are made up of rows and columns.

The relational model dictates that each row in a table be unique. The uniqueness can be established by designing a primary key, a column that contains unique values for a table. Each

---

\(^{1}\) Or the more physical terms files, fields and records

\(^{2}\) According to ANSI (American National Standards Institute), SQL is the standard language for relational database management systems. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc.
table can have only one primary key, even though several columns (or combination of columns) may contain unique values. All columns (or combination of columns) in a table with unique values are referred to as **candidate keys**, from which the primary key must be drawn. All other candidate key columns are referred to as **alternate keys**. Keys can be simple or composite. A simple key is a key made up of one column, whereas a composite key is made up of two or more columns.

### b. Foreign Keys and Domains

A **foreign key** is a column in a table used to **refer a primary key in another table**. **Domains** can be said as **user-defined column types** whose definition implies certain rules that the columns must follow and certain operations that can be performed on those columns.

### c. Relationships

**Relationships** between real-world entities can be quite complicated and involving many entities that have many relationships with each other as well. In a relational database, the relationships can be considered as between pairs of tables. These tables or relationships can be related in: **one-to-one**, **one-to-many** or **many-to-many**.

- **One-to-One Relationships**
  
  Two tables are related in a one-to-one (1–1) relationship if, for every row in the first table, there is at most one row in the second table.

- **One-to-Many Relationships**
  
  Two tables are related in a one-to-many (1–M) relationship if for every row in the first table, there can be zero, one, or many rows in the second table, but for every row in the second table there is exactly one row in the first table.

- **Many-to-Many Relationships**
  
  Two tables are related in a many-to-many (M–M) relationship when for every row in the first table, there can be many rows in the second table, and for every row in the second table, there can be many rows in the first table.

---

**Figure 1.**

Example of (1.) One-to-One Relationship Table, (2.) One-to-Many Relationship Table and (3.) Many-to-Many Relationship Table

(adapted from Litwin 2000)
d. Normalization

Normalization is the process of simplifying the design of a database so that it achieves the optimum structure\(^3\). Normalization delivers the concept of normal forms to optimise structure. The normal forms are:

- **First Normal Form** (1NF) states that all column values must be atomic (indivisible). 1NF requires that for every row-by-column position in a given table, there exists only one value, not an array or list of values.
- **Second Normal Form** (2NF) will be achieved if a table is in 1NF and every non-key column is fully dependent on the (entire) primary key. In other words, if the tables only stores data relating to one “thing” (or entity) that should be described by its primary key.
- **Third Normal Form** (3NF) will be gained if a table is in 2NF and if all non-key columns are mutually independent.
- **Higher Normal Forms** consists of Boyce Codd Normal Form, (4NF) and (5NF). Every higher normal form is a superset of all lower forms. Thus, if the design is in Third Normal Form, by definition it is also in 1NF and 2NF. If the database is normalised to 3NF, it is likely also to achieve Boyce Codd Normal Form and maybe 4NF or 5NF.

Un-normalised table:

![Un-normalised table](image)

…after 1NF, 2NF and 3NF …

![Un-normalised table and after its 3NF](image)

**Figure 2.**

Un-normalised table and after its 3NF
(adapted from Litwin 2000)

SQL allows users to access data in RDBMS such as Oracle, Sybase, Informix, Microsoft SQL Server, Access, and others, by allowing them to describe the data the user wishes to see. In addition, SQL also allows users to define the data in a database, and manipulate that data using the particular statements like SEARCH, FROM, WHERE (Hoffman 2000).

---

\(^3\) A well-designed database always follows the Normal Forms.
1.2. **OBJECT ORIENTED DATABASE SYSTEM**

The main idea behind the object-oriented database is that the databases consist of objects rather than relations, tables or other data structures (Subieta 1999). The concept of “object” is a kind of idiom or metaphor that addresses the human psychology and the way humans perceive the real world and think. There is such mechanism that is enabling someone to isolate objects in the environment, to name them, and to assign to them some properties and behaviour.

1.2.1. **Object Oriented Basic Idea**

The basic idea of object-oriented is following the natural human psychology. This is supposed to be achieved by reducing the gap between the human perception of the problem domain (such as business problem), an abstract conceptual model of the problem domain (such as a class diagram), and the programmer’s view of data structures and operations.

Minimizing the distance between these three views of designers’ and programmers’ thinking is referred to as “conceptual modelling”.

Object-oriented models offer notions that enable the analyst and designer to map the business problem to the abstract conceptual schema better. These notions include: complex objects, classes, inheritance, typing, methods associated to classes, encapsulation and polymorphism⁴.

There are several semi-formal notations and methodologies (such as OMT – Object Modelling Technique, UML – Unified Modelling Language, OPEN - Object-oriented Process, Environment and Notation) that make it possible to map efficiently a business problem to an object-oriented conceptual model. Moreover, object database systems offer similar notions from the side of data structures and hence the mapping between the conceptual model and data structures is much simpler than in the case of the traditional relational systems.

---

⁴ All will be briefly discussed further later
1.2.2 Object-oriented concepts
Object-oriented database models adopt the concepts of object-oriented programming languages. Actually, there is no agreement concerning their precise definition. The definitions presented below are most typical (Loom 1995 in Subieta 1999).

a. Complex objects, object identity.
The database should consist of objects having arbitrary complexity and an arbitrary number of hierarchy levels. Objects can be aggregates of (sub-) objects. Each database object has identity, i.e. a unique internal identifier (OID). Each object has one or more external names that can be used to identify the object by the programmer.

b. Relationships, associations, links.
Objects are connected by conceptual links. For example, the “Employee” and “Department” objects can be connected by a link “worksFor”. In the data structure, links are applied as logical pointers (bi-directional or uni-directional).

c. Encapsulation and information hiding.
The internal properties of an object are divided into two parts: public (visible from the outside) and private (invisible from the outside). The user of an object can usually refer only to public properties.

d. Classes, types, interface.
Each object is an instance of one or more classes. The class is defined as a blueprint for objects; i.e. (i) objects are installed according to the information presented in the class and (ii) the class contains the properties that are common for some collection of objects (objects' invariants). Each object is assigned a type. Objects are accessible through their interfaces, which specify all the information that is necessary for using objects.

e. Abstract data types (ADT): is a kind of a class, which assumes that any access to an object is limited to the predefined collection of operations.

f. Operations, methods and messages.
An object is associated with a set of operations (called methods). The object performs the operation after receiving a message with the name of operation to be performed (and parameters of this operation).

g. Inheritance.
Classes are organized in a hierarchy reflecting the hierarchy of real world concepts. For instance, the class Person is a superclass of the classes Employee and Student. Properties of more abstract classes are inherited by more specific classes. Multi-inheritance means that a specific class inherits from several independent classes.

h. Polymorphism, late binding, overriding.
The operation to be executed on an object is chosen dynamically, after the object receives the message with the operation name. The same message sent to different objects can appeal to different operations.

i. Persistence.
Database objects are persistent, that is they live as long as necessary. They can outlive programs, which created these objects.
1.2.3 Architecture of Object-Oriented DBMS

There are several concepts of the OODBMS architecture. The most well known is the ANSI/SPARC architecture which is also known as the three-schema. It is designed to separate the user applications and the physical database (Tsichritzis and Klug 1978, in Elmasri/Navathe 1999).

Another architecture is client/server architecture, where database applications are divided into two parts namely database server (which executes SQL statements sent by clients) and one or more clients sending requests to the server. More advanced architecture are the three-tier architecture and multi-tier architecture, where layers (tiers) of a user interface and a database are separated by one (or more) layers devoted to business logic.

From the functional point of view, a typical OODBMS architecture is presented below, which shows dependencies between basic functional components of a system.

(See the picture in the next page)

---

5 ANSI/SPARC stands for American National Standards Institute/Standards Planning And Requirements Committee
1.2.4. Functionality of Object-Oriented DBMS

OODBMS provides traditional database functionality (such as persistence, distribution, integrity, concurrency and recovery) based on the object model.

These functionalities typically provide permanent, immutable object identifiers to guarantee integrity. OODBMS also generally provides transparent distributed database capabilities (like transparent object migration, distributed transactions), and other advanced DBMS functionality (such as support for Web, support for workgroups, administrative tools).

In comparison to their relational peers, OODBMS are well suited for handling complex, highly interrelated data, particularly in cross-platform and distributed environment. There are also some benchmarks showing that performance of OODBMS is much better than RDBMS\(^6\) (Subieta 1999).

---

\(^6\) This is due to the new techniques, such as new caching methods, pointer swizzling, navigation via pointer links instead of performing joins, shifting processing on the client side, and others
Part II.

“THIRD MANIFESTO” Revisited:
A Criticism to SQL & OO Database Systems

Remark to this part:
It was an honour to have a presentation directly by Mr. Hugh Darwen, (one of) the famous author of the book “Third Manifesto – Foundation of Object-Relational Databases” [Addison Wesley Longman Publisher, 22 July 1998]
He was delivering his comprehensive thought in a very interesting and “inspirative” presentation at UMIST in CT-413 lecture on Thursday, 19 October 2000. Most of this part refers to the presentation.

The history of database manifestos started in mid 1980s, when E.F.Codd, the “father of the relational model”, published the 12 rules of a true relational system7 (Pepperdine 2000).

Following Codd, respectively came the first manifesto called, “The Object-Oriented Database System Manifesto” by Malcolm Atkinson et al8 and the second one called, “Third Generation Database System Manifesto” by Mike Stonebraker.9

“The Third Manifesto” by Darwen and Date postulates to reject both object-orientedness and SQL which wasted the ideals of the relational model and to return to the bosom of the pure relational model and 12 rules (Darwen 2000).

2.1. “THIRD MANIFESTO” – AT A GLANCE
While the First manifesto (namely the object oriented database model) essentially ignores the relational model entirely, the Second Manifesto in one hand agrees that the relational model must not be discarded, but in other hand decides that the relational model solely means SQL that must be involved with. It is implied therefore that there is an assumption that everything will grow out of SQL. The Third Manifesto definitely rejects this assumption.

A section in The Manifesto book, called “Back to the Future”10, says that to do the foundation right, the concept must get back to its roots, which is the relational model. By that way it is clear that Third Manifesto stays very firmly in the relational model. However, the Third Manifesto also says that there are two good ideas in object-oriented database. One is the data type concept that is the user-defined data use of arbitrary complexity and user-defined functions. The other is inheritance.

---

7 They are (1) The Information Rule, (2) Guaranteed Access Rule, (3) Systematic Treatment of Null Values, (4) Dynamic On-line Catalogue Based on the Relational Model, (5) Comprehensive Data Sub-language, (6) View Updating Rule, (7) High-level Insert, Update and Delete, (8) Physical Data Independence, (9) Logical Data Independence, (10) Integrity Independence, (11) Distribution Independence and (12) Nonsubversion Rule – according to them, up to now, no commercial RDBMS has been “truly relational” – look also at (Elmasri/Navathe 1999) -


9 By ACM SIGMOD Record 19,3 (September 1990) and also according to the on-line interview with CJ. Date at DBMS Online Interview – October 1994 (Kalman 1999)

10 According to Mr. Hugh Darwen in his presentation – because I do not read the book yet.
Therefore, the basic idea of the Third Manifesto is how to *marry object technology and relational technology*. The Third Manifesto concerns on how to integrate the good ideas of object-oriented database with relational ideas. The Third Manifesto has found that there is absolutely nothing to do with the relational model. The relational model is so solid, because it “needs no extension, no correction, no subsumption, and, above all, no perversion” (Atkinson 1989 in Kalman 1994). Therefore it certainly has the power to accommodate the good ideas of object oriented.

According to the Third Manifesto, the other way to say the same thing is that the good ideas of object-oriented are **completely orthogonal to the relational model**. It means that there is no need to destroy the relational model to get what the Third Manifesto wants. Then, the key question raised in bringing these technologies together is, “what in the relational world is the analog of the object class in the object world?” The answer is: the *domain* or *data type* (Darwen 2000).

Furthermore, Darwen explained that the key notion underlying the Third Manifesto is the equation that “domain = object class”. A domain, or an object class, is a *data type* that is *encapsulated*, which means that the only to operate on values of that type is through operators that are defined for the type. The users do not actually see the way the data is represented because it is not relevant. The users only know that there are certain functions they can perform. It might be a *primitive system-defined data type* that is going to be a *user-defined data type*. The values of these data types can be arbitrarily complex.

The Third Manifesto introduces a “mythical language”, called D. It serves as the basis for examples used to reinforce their proposals contained in the book. In addition, there is a comparison between D language to SQL3 and to those advocated by the Object Database Management Group (ODBMG) as well.11

### 2.2. CRITICISM TO SQL (RELATIONAL DATABASE)

The biggest misconception from the point of view delivered by the Second Manifesto is that the SQL is the same as the relational model.

Factually, when people look at relational products, they see SQL products and assume that SQL and the relational model are the same thing. However, the fact is that they are not the same thing. As its consequence, all the flaws of SQL are regarded as flaws of the relational model. The biggest flaw of SQL is that it does not implement the relational model.

More particularly, Darwen pointed out these fatal flaws of SQL:
- *anonymous columns* – a correctable flaw that is not bypassable
- *FROM clause restricted to named tables* – also a correctable flaw that is not generally bypassable. There is a note that while lack of support for this is a fundamental error; the suggested fix leads to queries that are difficult to write and to understand.
- *duplicate column names* – sometimes SQL generates two columns with the same name that are not duplicate columns. This shortcoming has the same correctability and bypassability as anonymous columns
- *duplicate rows* – a bypassable flaw but not a correctable one. For example, “declare at least one candidate key for every base table”, “always write DISTINCT after the word SELECT” and “never write the word ALL after UNION”.

---

11 As it has been told by Mr. Hugh Darwen in the presentation
- *nulls* -- SQL implementation of nulls is that SQL thinks that the sum of the empty set is null. Neither it is bypassable nor is it correctable.
- *failure to support degenerate cases (such as columnless tables)* -- from the fact that SQL does not acknowledge the empty set, therefore SQL cannot have a table with no columns, FROM clause cannot specify “no tables”, primary key and foreign keys cannot be empty and SQL cannot explicitly GROUP BY no columns.
- *updating views without checking option, lack of decent integrity model* and many more if the user are not careful.

### 2.3. CRITICISM TO OBJECT ORIENTED DATABASE

The essential role in the development of object DBMS was fulfilled by “The Object-Oriented Database System Manifesto” by Atkinson et al (Atkinson 1989 in Kalman 1994).

In one hand, there is a strong argument used by the relational database concept that there was no reasonable definition of the object oriented database concept. However in the other hand, the object oriented database manifesto has determined its basic rules, which contrary abandon the relational model.

The characteristics of an object-oriented DBMS were separated into three groups (Date in Kalman 1994):

- **Mandatory**: complex objects, object identity, encapsulation, types or classes, inheritance, overriding combined with late binding, extensibility, computational completeness, persistence, secondary storage management, concurrency, recovery and ad hoc query facilities.
- **Optional**: multiple inheritance, type checking and inferencing, distribution, design transactions and versions.
- **Open** (to decide by designers): the programming paradigm, the representation system, the type system and uniformity.

The object oriented database manifesto was unacceptable for the conservative point of views of the relational model. The “The Third Generation Database Systems Manifesto” postulates retaining all practically proven features of relational database systems (notably SQL, as an “intergalactic dataspeak”) and augmenting them modestly by new features, among them with some object-oriented concepts.\(^{12}\)

### 2.4. “THIRD MANIFESTO” : OBJECT – RELATIONAL DATABASE

According to Mr. Darwen, the Third Manifesto is intended as a blueprint (or a candidate for such a blueprint) for the future direction of data and database systems. Specifically, it consists of a model that acts as a basis for the design and implementation of DBMS, including object/relational DBMS (ORDBMS). The model should be one that is meant to be abstract, formal, robust, rigorous, and well defined.\(^ {13}\)

---

\(^{12}\) In the presentation, Mr. Darwen did not explicitly mention this, but inferred it. The more explicit views of Third Manifesto’s criticism to object oriented database ware found in the interview of CJ. Date, 1994 (Kalman 1994)

\(^{13}\) An extended version of that model provides a comprehensive and detailed proposal for type inheritance support.
The key technical insight behind the Third Manifesto is very simple. In essence that: *There is nothing to do with the relational model in order to achieve object functionality.* More specifically, the Third Manifesto points out that relational model already provide the capabilities it needs.

Then the Third Manifesto elaborates them together:

- The useful aspect of the object model is the ability for users to define their own data types, and that facility is provided by domains in the relational model.
- Hence, a relational DBMS that supported domains properly would be able to deal with all those complex kinds of data that object systems can handle but relational ones cannot: such as time-series data, biological data, financial data, engineering design data, office automation data, etc.
- Therefore, a true “object/relational” system would be nothing more or less than a true relational system.

Furthermore the Third Manifesto suggests that a database that is not relational is not really a proper database. Database according to the manifesto is a set of true statements. The manifesto left behind the existence of a “false fact” that might be a result from a proposition. So a database is, abstractly, just a collection of true propositions. And relational theory supports this view very directly, because tuples in relations (or rows in tables) are directly interpretable as such true propositions.

Thus conceptually, a database must be relational, by definition. It then follows the term “object/relational” it uses.
Part III.
Further Discussion on
SQL 3 and Object Relational Database

Evaluating what the Third Manifesto has concerned about, this part aims to further technically-but-very-briefly
discuss the SQL —one of the most important objections of the manifesto—and the Object-Relational Database
that the manifesto recommends. Neutral resources are referred to get a broader and fair discussion. In addition to
SQL discussion, it will also concern with SQL 3 which is now the latest version of SQL.

3.1. SQL
Originally, SQL was called SEQUEL (for Structured English QUEry Language) and was
designed and implemented at IBM research as the interface for an experimental relational
database system called SYSTEM R. (Elmasri/Navathe 1999). Variations of SQL have been
implemented by most commercial DBMS vendors.

A joint effort between ANSI and ISO has achieved to a standard of SQL, called SQL1 (ANSI
1986). A further expanded standard of SQL has been achieved as well which outcomes are SQL-
89 and SQL-92 – later called SQL2. The latest standard of SQL is SQL3 (ANSI 1994, Manola
1995). Different from the previous versions (SQL-86, SQL-89, SQL-92) SQL3 is also a
programming language with full computational power. The main data structure in the table is
equipped with a lot of options.

SQL3 has been equipped by the object-oriented database concepts. It supports user-defined
abstract data types (ADT), including methods, object identifiers, subtypes, inheritance and
polyomorphism. Some enhancements in SQL3 are: statements defining tables, in particular types
of rows, row identifiers, and specific inheritance between rows of families of tables. Further
facilities include control statements and parameterised types. SQL3 is compatible with SQL-92
and follows the SELECT – FROM - WHERE syntax (Subieta 1999).

Each SQL3 table has a predefined column named IDENTITY containing row identifiers. These
identifiers can be used as values in other rows. SQL3 therefore makes it possible to create
pointer-based, network data structures just like CODASYL DBTG 14. The ADT concept in
SQL3 enables the user to associate values stored in the tables with methods that can be written
in SQL3 or other languages, in particular in C++ and Java. SQL3 introduces a lot of extensions
to previous SQL standards, including the collections, triggers, transactions, cursor processing,
etc. It includes also new features such as support for user-defined aggregate operations, transitive
closures, and even deductive rules.

According to (Subieta 1999), there is a lot of controversy around the SQL3 standard. This paper
will only mention the three of them.

Firstly, the standard is extremely huge, currently ca. 1100 - 1600 pages. In the history of software,
no implemented language has specification longer than 1000 pages. Secondly, a controversy is

---

14 CODASYL DBTG: Computer Data Systems Language Data Base Task Group, which is the committee that specified
the network model and its language, particularly data definition language (DDL) and data manipulation language
(DML)
caused by the fact, that research and development of a new programming language is carried on in the standardisation office. The standardisation office’s status does not provide this kind of activity and lack of incompatible competence. Thirdly, some people doubt if the standardisation office (financed from public sources) can spend a lot of money on support of private ORDBMS vendors rather than pure OODBMS.

3.2. OBJECT-RELATIONAL DBMS

Object-relational DBMS (ORDBMS) are based on the idea that the relational model and SQL-92 implemented in the majority of RDBMS should be extended (Darwen 2000, Subieta 1999, Date in Kalman 1999).

There are actually many ways to extend the relational model, however, the one chosen (this includes by the Third Manifesto) is very relation-centric. A database still consists of a set of tables. A number of object features are provided as an extension to this core relational model, namely multi-row tables, references between rows, inheritance between tables, etc.

Nowadays, there is no standard for the object-relational model and each vendor has own object extensions. For example, the differences between Oracle-8, Informix Dynamic Server, UniSQL, and other object-relational products are significant. They exclude portability and even a common conceptual and didactic basis. Various products present the choice of quite random, redundant, limited and sometimes inconsistent features. A new ANSI/ISO standard SQL3 (discussed above) is supposed to unify the systems.

The motivation of the implementation of the object-relational database by the DB vendors is more the commercial profit rather than conceptual clarity or software engineering principles. The products are criticized as huge, complex, eclectic, and thus decadent (Subieta 1999).

The key products of this technology include: Informix Dynamic Server (formerly Postgres, Illustra and Informix Universal Server), IBM’s DB2 Universal Database, Oracle-8, UniSQL/X, OSMOS, Ingres II, Sybase Adaptive Server, and others. Every relational vendor has announced object-relational products. With exception of some new systems (Postgress, Montage, Informix DS, UniSQL), the systems are extensions of their relational predecessors. They are equipped with facilities for efficient application development. These are: support for multimedia and Web, spatial data, ADT, methods defined by the programmer, collection-valued attributes, and others. (Chamberlin 1996, Subieta 1999)
To further discuss the progression of database technology, a summary below might be helpful to trace-back the progression itself.

<table>
<thead>
<tr>
<th></th>
<th>1960s to mid-1970s</th>
<th>1970s to mid-1980s</th>
<th>1980s to early 1990s</th>
<th>Future</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Model</strong></td>
<td>Network</td>
<td>Relational</td>
<td>Semantic</td>
<td>Merging data models with knowledge representation</td>
</tr>
<tr>
<td></td>
<td>Hierarchical</td>
<td></td>
<td>Object-Oriented</td>
<td>Hybrid models</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td><strong>Database Hardware</strong></td>
<td>Mainframes</td>
<td>Mainframes</td>
<td>Faster PCs</td>
<td>Client-server configuration</td>
</tr>
<tr>
<td></td>
<td>Minis</td>
<td>Minis</td>
<td>Workstations</td>
<td>Parallel processing</td>
</tr>
<tr>
<td></td>
<td>PCs</td>
<td>PCs</td>
<td>Dbase machines</td>
<td>Optical memories</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Back ends</td>
<td></td>
</tr>
<tr>
<td><strong>User Interface</strong></td>
<td>None</td>
<td>Query languages</td>
<td>Graphics</td>
<td>Multimedia</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Forms</td>
<td>Menus</td>
<td>Natural languages</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query-by-forms</td>
<td>Speech input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Freehand text</td>
</tr>
<tr>
<td><strong>Program Interface</strong></td>
<td>Procedural</td>
<td>Embedded query</td>
<td>Standardised SQL</td>
<td>Generalised presentation managers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>languages</td>
<td>4GL</td>
<td>Distributed, heterogeneous data and knowledge processing with multimedia information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Logic programming</td>
<td></td>
</tr>
<tr>
<td><strong>Presentation and Display</strong></td>
<td>Reports</td>
<td>Report generators</td>
<td>Business graphics</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Image Output</td>
<td></td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Processing data</td>
<td>Information and transaction processing</td>
<td>Transaction processing</td>
<td>Parallel database management</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.
Database Technology Trends
(Source: Elmasri/Navathe 1999)

According to (Elmasri/Navathe 1999) as systematised in the table above, the trend of technology for the next several years will be dominated by themes such as:

**a. Distributed, heterogeneous environments**

Large centralised databases will be replaced by distributed databases. These databases might be implemented from conventional network, current relational DBMS, object oriented DBMS or combination of both. In addition, the client-server architecture is popular and will provide users with various options.
b. Open systems
Several standardisations in database are trying to improve the openness of future databases and applications. These include SQL Access Group, ANSI/X3.H7, OMG (Object Management Group), ODMG (Object Data Management Group), Microsoft’s ODBS, IDAPI and ODAPI (Borland). The systems that are compatible with those standards will be able to communicate each other easily. In addition, SQL2 and SQL3 standards are projected to play a major role in the open systems. The trend will be toward designing global applications that run on client’s side systems and access data from a various servers with standardised data access protocol.

c. More functionality
The functionality of future DBMS will go beyond the conventional define, store, access, query and report functions. The deductive capability, the processing meta-data and better specification of behaviour through methods will be included as well. In addition, the DBMS will be more responsive as it includes multimedia database management, better processing of temporal and spatial data, etc.

d. Parallel database management
A new DBMS based on MIMD (multiple instructions over multiple data paths) might come and be offered in commercial use. This system makes use of algorithm for parallel query strategies for data allocation and therefore becomes effective to minimise cost per transaction within the overall system.

The long-debate since the relational database, object oriented database to the new object-relational database becomes however fruitful that the database technology itself then progresses. Then, relating the future database to the topic of this paper, the book “Fundamentals of Future Database” (Darwen et al 1999 in Darwen 2000) concludes mainly that the Third Manifesto is the basis for the future database, in particular, the object-relational database.

However, many professionals consider ORDBMS as a temporary result of the evolution from the relational to the pure object-oriented technology. The current market, hardly accepts new database ideas and languages. Thus it may happen that the evolution of database will take tens of years (Subieta 1999).

So it becomes very hard to predict where this evolution will come.
Conclusion

The relational model is simple and elegant, but it is fundamentally different from the object model. Relational databases are not designed for storing objects. Relational databases are poorly adapted to express the basic attributes of objects like encapsulation, inheritance, polymorphism, object identity, references among objects, and user defined data types.

The object oriented databases greatly simplify object database management because they support the object model directly. The standard for the object database technology is organised in the ODMG (Object Database Management Group). The ODMG-93 standard covers the essentials for object database programming, especially the OO model and the OOP language bindings. This standard has already many advantages compared to extended relational database systems that do not comply with any standard.

The comparison of SQL and ODMG standards is illustrated below.

Third Manifesto is intended as a blueprint for the future direction of data and database systems. It consists of a model that acts as a basis for the design and implementation of DBMS, including object-relational DBMS (ORDBMS) drawn from relational and object-oriented model. The object-relational model should be one that is meant to be abstract, formal, robust, rigorous, and well defined (Darwen 2000).
References


8. (Kalman 1994). David M. Kalman, DBMS-Online Interview, “Moving forward with relational: looking for objects in the relational model, Chris Date finds they were there all the time”. Available online at [http://www.dbmsmag.com/int9410.html](http://www.dbmsmag.com/int9410.html)


15. Teradata ORDBMS SQL Online Reference at [http://www.info.ncr.com/kw-result.cfm](http://www.info.ncr.com/kw-result.cfm)