# Difference between T-SQL and the Relational Model

Davor Lozić [1], Alen Šimec [1]

[1]*Tehničko veleučilište u Zagrebu, Vrbik 8, Zagreb, Croatia*

***Abstract* - T-SQL, dialect of SQL, is a language used for a relational database management system (primarily Microsoft's SQL Server), which in turn is based on the relational model. Understanding some of the key foundation principals can help in a better understanding of the language.**
**This paper introduces basics of the relational model and the difference between the relational model and the actual T-SQL implementation.Several tightly related topics like predicates, relations, and common misconceptions about databases are also discussed.**

***Keywords* – relational model, T-SQL, predicate, relation**

## 1 Introduction

The relational model is a mathematical model for data management and manipulation which was initially proposed by Edgar Codd in 1969 [1]. It's based on set theory and predicate logic (figure 1).

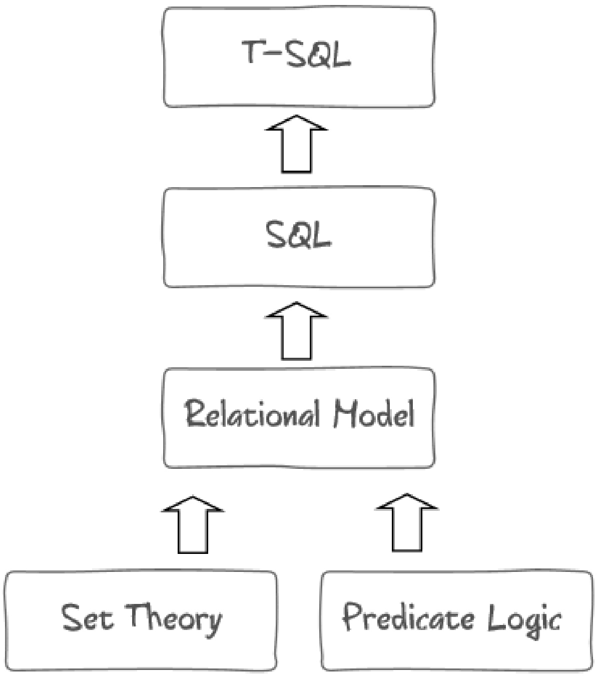


*Figure 1 - T-SQL.*

### 1.1 Set theory

George Cantor, the founder of set theory, defined a set as follows:

By an "aggregate" we are to understand any collection into a whole *M* of definite and separate objects *m* of our intuition or our thought. These objects are called the "elements" of *M*.

The German word for a set is Menge, which is the reason Cantor denotes a set by M and its elements by m. Menge is translated as an aggregate, but it has since become common to use the word set instead [2].

Fundamental concept to understand about sets is that they are determined *uniquely* by its members [3]. That means that sets {5, 5, 7, 2} and {5, 7, 2} are identical. Order of elements inside the set is not important so the set {2, 5, 7} is the same as the previous two.

### 1.2 Predicate logic

A predicate is a generalization of a propositional variable [4]. In other words, a predicate is an expression that, when attributed to some object, makes a proposition either true or false [5]. For example, when "has a car" predicate is evaluated against employee, it's a proposition. Predicates make filtering and data integrity possible. They can be even used for defining the data model itself. Example: invoice with ID 1128 has been created on 2015-01-01 by the user with ID 201. In this example, defining the data model is simple: invoiceID INT, invoiceCreated DATE, userID INT.

## 2 Common misconceptions

A common mistake is to think that "relational" in relational model has something to do with foreign keys and relationships between tables. Relation is what SQL calls a *table* but the two are not synonymous. **Table is an engineering approximation to the relational model**.

A relation has a heading with a set of attributes and a body with a set of tuples [6]. SQL attempts to represent attributes with columns and a set of tuples with rows. Attribute is identified with name and type name and each tuple is defined with heading (which corresponds to the heading of the relation) and values with a respective types (Figure 2).

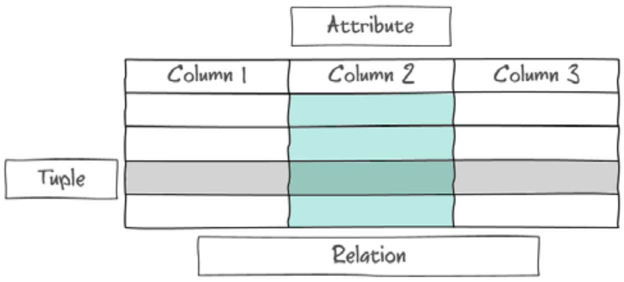
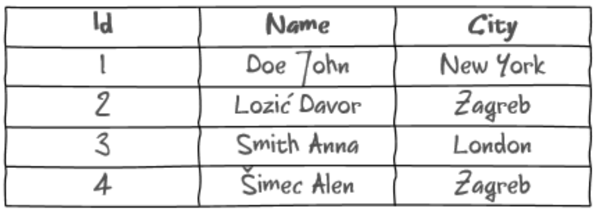


*Figure 2 - Relational model*

## 3 Differences

Main differences between T-SQL and relational model will be shown in this chapter. Figure 3 shows the data used for queries [7].



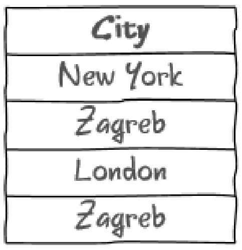
| Id | Name | City |
|---|---|---|
| 1 | Doe John | New York |
| 2 | Lozić Davor | Zagreb |
| 3 | Smith Anna | London |
| 4 | Šimec Alen | Zagreb |

*Figure 3 - Example data (Users)*

**Set has no duplicates** but the T-SQL doesn't enforce this rule. Example:

SELECT City FROM Users;

Figure 4 shows the query result. As shown, SELECT statement doesn't enforce unique rows in result.

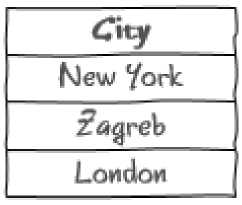


| City |
|---|
| New York |
| Zagreb |
| London |
| Zagreb |

*Figure 4 - Result does not contain unique rows*

T-SQL implements keyword DISTINCT which guarantees unique rows and returns relational result:

SELECT DISTINCT City FROM Users;



| City |
|---|
| New York |
| Zagreb |
| London |

*Figure 5 - Unique results returned by DISTINCT*

Another non-relational concept which T-SQL allows is having a column without defining a name. Often, there is a need for defining the result based on the expression:

SELECT Name + ‘‘ + City FROM Users;

This query creates attribute without the name but the relational model doesn't allow creation of attributes without giving them a unique name inside the relation. Attribute name uniqueness is another problem. Consider having a simple join between two tables where both tables have the attribute name. To solve both problems, T-SQL implemented AS clause which can assign the alias to the given target.[8] Next query perfectly respects all relational model rules:

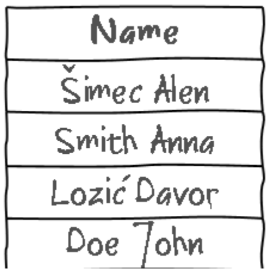
SELECT Name + ‘ ‘ + City ASFullDescriptionFROM Users;

T-SQL's implementation has another deviation from the relational model. Its predicate logic only implements TRUE, FALSE, and NULL but according to Edgar Codd, there should be another value. Example is when the user doesn't want to provide or when the user doesn't have an email. Both cases should have different value inside the relation.

For a query to be called relational, it needs to return a relation after executed. Query with the ORDER BY clause is not relational because the set theory clearly states that the order in which the elements of a given set are listed does not matter [2]. The next query guarantees ordering by name and that is the reason why the result of the query is not a relation:

SELECT Name FROM Users ORDER BY NameDESC;

Figure 6 shows a result which guarantees the order of rows.



| Name |
|---|
| Šimec Alen |
| Smith Anna |
| Lozić Davor |
| Doe John |

*Figure 6 - Example data (Users)*

After executing a query, result also returns a list attributes in a specified order. If the query contains SELECT *, T-SQL guarantees the same order in the result based on their order inside the table definition. The true relation doesn't care about the order of attributes and that is another deviation from the relational model.

In T-SQL, order of columns is defined and significant but the relational model requires there to be no significance to any ordering. It's not even easy to change the column order inside most of the SQL implementations. In SQL Server, when a user changes the column order, system recreates the table from scratch. SQL Management Studio will not allow such operation if the option "*Prevent saving changes that require table re-creation*" is checked (Figure 7).

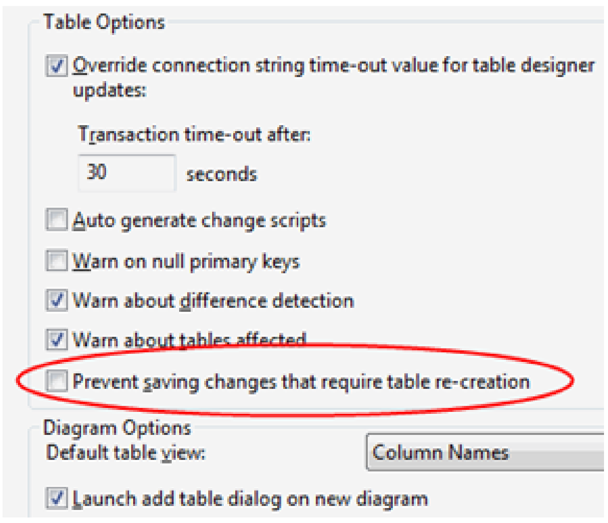


Table Options

[✓] Override connection string time-out value for table designer updates:

Transaction time-out after: 30 seconds

[ ] Auto generate change scripts

[ ] Warn on null primary keys

[✓] Warn about difference detection

[✓] Warn about tables affected

[ ] Prevent saving changes that require table re-creation

Diagram Options

Default table view: Column Names

[✓] Launch add table dialog on new diagram

*Figure 7 – Prevent saving changes that require table re-creation*

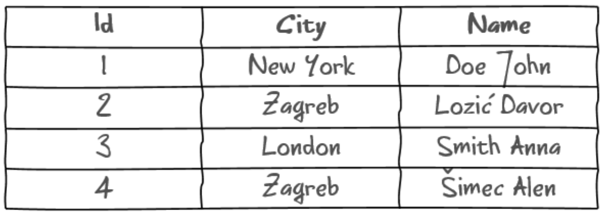
To prevent T-SQL from recreating a whole table, Views can be used. A view is a virtual table and it doesn't exist physically.It'sdata source is based on the result of an SQL statement. Consider the data in Figure 3. Requested column order is *Id*, *City*, *Name*, instead of *Id*, *Name*, *City*.

```
CREATEVIEWUsersView
AS
SELECT Id, City, Name FROM Users;
```

Getting the data is same as previous examples:

```
SELECT * FROMUsersView;
```

Result is shown in Figure 8.



| Id | City | Name |
|---|---|---|
| 1 | New York | Doe John |
| 2 | Zagreb | Lozić Davor |
| 3 | London | Smith Anna |
| 4 | Zagreb | Šimec Alen |

*Figure 8 – Result after getting data from a View*

## 4 Non-relational queries

This chapter will provide some of the non-relational queries seen in this paper, grouped in one place which could also be used as a reference for creating relational queries.

```
# this query doesn't guarantee
# uniqueness, use DISTINCT clause
SELECT City FROM Users;

# withDISTINCT clause
SELECT DISTINCT City FROM Users;
```

```
# no attribute name in this query
# AS must be used, also this query
# doesn't guarantee uniqueness,
# useDISTINCT clause
SELECT Name + ' ' + City FROM Users;

# withDISTINCT and AS clause
SELECT Name + ' ' + CityASUniqueNameFROM Users;
```

```
# this query contains ORDER BY
# in Set Theory, order is not
# important, also this query
# doesn't guarantee uniqueness,
# useDISTINCT clause
SELECT Name FROM Users ORDER BY Name DESC;
```

```
# withDISCTINCT and without
# ORDERBY clause
SELECT DISTINCT Name FROM Users;
```

## 5 Conclusion

Virtually all relational database management systems are based on SQL which is derived from a relational model. T-SQL and relational model is based on a strong mathematical foundations and understanding them can help in better understanding of the language. T-SQL deviates in a lot of ways from a relational model but almost all deviations could be avoided with appropriate language constructs.

## 6 References

[1] E. F. Codd, Relational Completeness of Data Base Sublanguages, IBM Research, 1972.

[2] G. Bezhanishvili and E. Landreth, An Introduction to Elementary Set Theory.

[3] F. Stephan, Set Theory, 2009 - 2010.

[4] A. Aho and J. Ullman, Foundations of Computer Science, 1992.

[5] I. Ben-Gan, D. Sarka and R. Talmage, Querying Microsoft SQL Server 2012, 2012.

[6] T. L. Saito, Silk: A Scalable Data Format In-Between Relations and Trees, Department of Computational Biology University of Tokyo, Japan.

[7] S. Misner and M. Ross, Introducing Microsoft SQL Server 2014, Washington: Microsoft Press, 2014.

[8] G. Fritchey, SQL Server 2012 Query Performance Tuning, New York: Apress, 2012.