BASIC RELATIONSHIP DATABASES CONCEPTS AND PRINCIPLES

Stefan Cliff 2019/230449

# Introduction to relational databases, Basic concept and principles

A **Relational (Relationship) Database** is a digital representation of data in a **Relational Model**.

Relational Model is a style of data organization where data is put into one or more tables (“*relations*”), which are then put into columns and rows. Each row is identifiable with their unique key. Rows are often also referred to as tuples or records, whereas columns can also be called attributes. Generally, each table servers to represent one instance of an entity where the rows represent the instances of the entity and the columns are the values.

One example of this model of information being represented:

* Let’s say we have a table of workers for some company. So, the rows would be something along the lines of: -first name; -last name; -personal ID number; -date of birth and so on. The columns would then be filled corresponding to the rows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | First Name | Last Name | Personal ID | Date of Birth |
| 1 | Stefan | Cliff | 1111111111111 | 03.06.2000 |
| 2 | Pera | Perovic | 2222222222222 | 01.01.2000 |
| 3 | Jovan | Jovanovic | 3333333333333 | 12.03.1999 |
| 4 | Marija | Marjanovic | 1122334455664 | 06.05.1997 |
| 5 | Aleksa | Dimitrijevic | 1354422423155 | 19.09.2001 |

We can connect this table with another table called Work Sector/Division, where one worker can be a part of a sector but one sector can have many workers.

This kind of relationship is called “**one to many**”. This is one of the three main relationship types that exist, the other two being: “**one to one**”, “**many to many**”. Although **many to many** does exist, it is usually avoided or worked around by the addition of another table to better and cleaner connect two or more tables. These extra tables are made in the other two styles.

Using this example, we can begin to talk about “**keys**”. A key is some data that serves as an identifier for that row in a table. These keys, called **unique keys**. Each table has one **primary key** which serves as the main point/anchor when used to define relationships among tables. When a primary key migrates to another table, it becomes a **foreign key**.

Many relational database systems have the option to manipulated or updated via queries using **Structured Query Language** (***SQL***). As of 2009, most commercial relational databases employ **SQL** as their standard query language.

# Short overview or history of relationship of databases

**Edgar Frank Codd** in 1970 first used the term relational database while working for **IBM**. He invented this term in a paper he was writing called: “*A Relational Model of Data for Large Shared Data Banks*”, which was the first stepping stone of many that held his definition of the term “*relational*”. One of the more well-known definition can be found in “*Codd’s 12 rules*”, which actual holds **thirteen rules as to what constitutes as a relational database**.

Following his papers, there actually hasn’t really been a commercial implementation of a relational database whose form conforms to all of Codd’s rules. Due to this, the term has over time been slowly but surely began to describe a broader class of database systems which must hold **at least two minimums**.

* **Firstly**, it must show all the data to the user in a relation/as relation. By that, it means to show the data in a tabular form i.e., as a collection of tables which each one of them consisting of both rows and columns *(example on previous page)*.
* **The second minimum**, is that the user should be given relational operators that can manipulate the data in its tabular form. Some of these operators are: *union, intersection, difference, cartesian product.* Codd’s operators are: *selection/restriction, inner join, relational division* and *projection operation.*

## Release dates of relational database programs

Back in 1974 **IBM** began to develop **System R**, which was a research project to develop a prototype **Relational Database Management System** (***RDBMS***). It was first sold as an ***RDBMS*** in June 1976, which was called **Multics Relational Data Store**.

Three years later, **Oracle** was released (*now known as* ***Oracle Corporation***).

**Ingres**, **IBM BS12**, **DB2**, **SAP Sybase ASE** and **Informix** were all released soon one after the other following **Oracles** release.

In 1984, the development for the first Macintosh based relational database program began under the code name: **Silver-Surfer**. It was released in 1987 as **4th Dimension**, also known as **4D**.

Some of the first systems that followed more faithfully the relational model were: - *University of Michigan (****Micro DMBS*** *1969); - Massachusetts Institute of Technology (1971); - IBM UK Scientific Centre of Peterlee (first* ***IS1*** *1970-72, replaced by* ***PRTV*** *1973-79).*

Due to some discourse, some **RDBMS** are referred to as ***Truly-relational (TRDBMS)*** or ***Pseudo-relational (PRDBMS)*** depending on if they follow all of “*Codd’s 12*” rules.

# Final remarks, thoughts and how do I see the future of relation databases progressing

While this is only the surface of relational database systems, there is a decent amount to unpack.

Firstly, I think that the possibility of a newer technology being created is always there, as is the way with modern technology. Programs are adapted or entirely recreated if presented a never-before seen problem and the likelihood of such a problem showing up soon is possible, I think we are a few years or maybe even decades away.

What I find more likely maybe being created anew or based off of already existing systems/programs is a more barebones system that can be altered by the consumer when the system is being installed, and later on be updated with custom commands and premade commands alike. Think of a *Linux* style **RDBMS**, where everyone is given a shell and it’s up to them and their knowledge/ability to fully customize their system.

This is only one of the probably many possibilities that this technologies path might be, personally I find it the simultaneously the most difficult to create but also most likely to be created or wanted. While roughly ~80/90% of the world will continue to use premade services, the few that do use this new technology would possibly be more advanced and maybe even more secure, as each system would be unique in some way.

Alas, this is all speculation and only time will tell what the world will want or need.