Relational Model

Dreycey Albin

September 2019

The key need for a relational model was that it allowed for data independence. Pointers and network-like representations were being used to structure data at the time the paper was published. The following dependencies were mentioned: (1) ordering dependence, (2) indexing dependence, and (3) access path dependence. In summary, it allows these dependencies to be relaxed, and allows for logic to describe operations between relations.

There are multiple short comings that current approaches at the time contained. For one, the relationships between data had to be physically stored to memory. This results in potential issues with consistency, because changing one relationship may cause a broken path between data. In addition, the existing approaches at that time were heavily reliant on the hierarchical structures of the access paths. This created problems as many systems use different structures for the hierarchies, so application programs had to be specific for a specific structure, since they are dependent on the hierarchy of the data.

The relational model of data is the key object. This model allows for data independence, a feature not represented in non-relational data systems. The relational model allows for shared data between relations/tables without having explicit pointers between data. This "functional dependence" solves many of the problems with data dependence. It also allowed for a predicate calculus to describe operations on the data, giving the data model a mathematical and logical framework. Because relational models have relationships computed based on matching keys, it addresses most of the challenges other systems/models have of data dependence [2].

One short coming of the relational approach would be with the computation needed to compute relationships. This can be an expensive approach if the operations needed to compute a particular relationship are complex. In contrast, if pointers are stored, as in other systems, then the network-like structure would already have the relationships stored. In addition, there all of the data has to be stored to the drive, which could be a problem for systems without modern physical memory [2].

Something I found interesting about this paper was how thorough most of the concepts were, even thorough it was the inception of the relational model. For example, concepts such as "triggers", "normalization", and key types were all defined in the original paper. This is surprising, as these are still major concepts that are considered for relational models almost 50 years later. Likewise, the

ability to define a predicate calculus for the relational model from the original inception is quite a feat. The IBM researchers must have really studied the potential use cases for the relational system, as well as many of the useful features to be focused on moving forward.

The Chen 1995 paper relates to the relational model paper by creating a data modeling scheme, called the entity-relationship model. This paper has many foundation concepts in connection with with the Codd 1970 paper because it attempts to relate the network model, entity set model and relational model. It builds on concepts arising from all three of these models, then extending the underlying relationships between the data into a visual technique for database design. The entity-relationship model introduced allows for database designers to visually draw the relationships between data, model the how to database should be built, all before implementing the database. New information I gained from reading this paper is both a better understanding of Chen notation for schema design, and in addition, I was introduced to the formality used to justify the data modeling technique. [1].

References

[1] Peter P.S. Chen. The entity-relationship model — towards a unified view of data. *ACM Transactions on Database Systems*, vol. 1(1):pages 377–387, 1995.

[2] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, vol. 1(1):pages 9–36, 1970.