RELATIONAL ROOTS

Presenter: Fong Chun Chan
Discussion Leaders: Noreen Kamal and Dibesh Shakya
Papers Covered

- A Relational Model of Data for Large Shared Data Banks. E. F. Codd (1970)
  - A seminal paper on relational databases which caused a paradigm shift in the data models community.

  - A paper about the experimental database system, System R, which implemented and demonstrated the feasibility and usability of relational models.
Before the relational model, two major data models were competing:

- Hierarchical

Figure taken from “What comes around goes around” (Stonebraker M, Hellerstein J.)
Background information

- Network/Graph

Figure taken from “What comes around goes around” (Stonebraker M, Hellerstein J.)
Background information

- Edgar Frank "Ted" Codd introduced the “Relational Model” in 1970 which sparked “The Great Debate” and eventually caused a paradigm shift.
- The relational model appeared to be superior in several aspects to the other competing models.
Motivations behind the model

- Provide a means of describing the data with its natural structure only. In other words, **data independence was a major goal**.

- Provide a foundation for high level data language that separates the application programs from the machine representation and organization of data.

- Permit a clearer evaluation of the scope and logical limitations of the present data systems.
Data Independency

- Data independency refers to making data applications immune from modifications in the definition and organization of the data it uses.

- Three principal kinds of physical data dependencies that needed to be removed:
  - Ordering Dependence
  - Indexing Dependence
  - Access Path Dependence
Existing systems require or permit the elements to be ordered in a way that is closely related to how the hardware orders them.

- Order of presentation vs. stored order
- No clear distinction between these two types of orderings

While it can be advantageous to have a stored ordering of a file, the system will likely fail to operate correctly if the ordering needs to be replaced.
Indices are performance-oriented components of the data which improve the speed of particular queries.

On a system which is consistently changing, the need to create and destroy indices at any particular time will be necessary:

“Can application programs and terminal activities remain invariant as indices come and go?”
An access path describes how to actually access the data (bits) on disk.

Existing data systems provide users with complicated tree-structured or network models of the data.

If the structure of these models were changed, the application programs would fail.
The Relational Model

- Everything can be represented as a relation
  - Relation = Set = Table
- Relations have domains (attributes)
  - Domains may have the same name
A major goal of the relational model was to ensure that users do not need to know about indices to write queries. Though users do not *need* to know about indices, changing them can have serious performance impact, leaving users puzzled. Has independence of indices really been achieved?
Benefits of the Relational Model

- In other models, the initial design of the system was very important:
  - For example, hierarchical model, the hierarchy had to be decided on ahead of time. Who is the parent of who? Who is the child of?
- With the relational model, because everything is represented as a relation it is no longer critical that all the relationships are decided at the initial design.
Benefits of the Relational Model

- With other models, if indices existed, then querying required knowing they existed and the removal of them make problems for applications using the data.

- With the relational model, indices could be created and dropped readily to enhance the system performance without having any real drastic effects.
Benefits of the Relational Model

- With other models, a structural change in the representation of the data meant that applications that used this data needed to be modified.

- With the relational model, a structural change doesn’t have such a drastic effect. Modifications of SQL queries are simpler.
Once the relational model made it to the market, people flocked to it and previous models were almost forgotten about.

Was it possible that the success of relational databases killed off any interest in making tree-structured data easier to work with?
Normal Form

- Simple domains (columns) have elements which are atomic values. A simple two-dimensional array can be used to stored this data.
- If the domain is non-simple, then a more complicated data structure is necessary.
- To eliminate these non-simple domains, Codd presents a technique called normalization.
  - This is not to be confused with the modern notion of normalization which is used to maintain the database integrity.
Operators

- Introduced to allow the ability to derive relations from other relations.
- Codd suggested four different operators:
  - Permutation (not used today)
  - Projection (used today)
  - Join (used today)
  - Composition and Restriction (not used today)
An introduction to the concept of relational databases which caused a paradigm shift.

We use many of Codd’s ideas today, but not everything “made it”:

- “…Codd was originally a mathematician...his DML proposals were rigorous and formal, but not necessarily easy for mere mortals to understand” (What Comes Around Goes around)
- Duplicate domain names
- Original concept of normalization
- Some operators
A final note on Codd’s Paper

- Paper was published in Communications of the ACM
  - A leading publisher for Computer Science and IT fields.
  - Accepted very technical papers back in Codd’s period, but not so much anymore.
Codd's paper is mathematically rigorous but doesn't have implementation or evaluation; and doesn't meet the requirement of conferences today. What does it say about the metrics today? Are we impeding the chances of paradigm change?
System R

- An experimental project to implement a relational database management system.
- One of the first relational database systems to be implemented.
Three phases of the project

- **Phase Zero: An Initial Prototype**
  - Designed to be a quick implementation of a subset of the functions. Intended to be thrown away.

- **Phase One: Construction of a Multiuser Prototype**
  - Re-design of the phase zero prototype with concurrent access and some new features.

- **Phase Two: Evaluation**
  - Review of the work done and some enhancements
Phase Zero: An Initial Prototype

- No concurrent access was implemented yet. Only single-user access was concerned.
- Supported the “subquery” SQL command, but not the “join” command.
  - A query was capable of searching through several tables to find the desired results, however the final results had to be from only one table.
Phase Zero: XRM

- XRM was used as the relational access method

  - Relations were stored as tuples with a unique TID associated with each one.
  
  - Tuple didn’t store any data itself, but contained pointers to “domains” that actually stored the data.
  
  - Inversions could be used to find TIDs of tuples that contained a given domain value.

Fig. 2. XRM Storage Structure.
Phase Zero: Optimization

- Designing an optimizer to efficiently run SQL queries on top of XRM was the most challenging part.
- Optimizer tried to minimize the number of tuples retrieved
  - Extensive usage of “inversions” was used
  - Didn’t take into account the “hidden costs” being the costs of creating and manipulating the TID lists, fetching those tuples, and then using the pointers to finally fetch the data.
- “A better measure of cost would have been measure of I/Os”
  - Storing the data values separate from the tuples led to many I/O requests to retrieve the data.
Discussion: Why Prototype???

They first implemented a Phase 0 prototype, which is currently the norm (i.e. to implement a prototype).

What benefits were truly obtained by having a prototype phase? Indeed much was learned about the limitations of XRM, but this was already identified as it was defined as single user,

So was it a "waste" of time to go through work of creating this phase if it was always meant to be abandoned?
Phase One: Construction of a Multiuser Prototype

- Scraped phase zero, but learned from evaluating it.
- The Research Storage System, RSS, replaced the XRM as the relational access method.
- Implemented concurrent access with a locking subsystem
- Implemented a recovery subsystem
- Implemented a security system with view and authorization subsystems.
Phase One: Optimization

- The RSS access method was more efficient
- Didn’t rely on manipulating TID lists, but scanned each table in the query using any available index that exists to find the desired results.
- The choice of the access path is entirely up to the optimizer and was abstracted from the user.
Phase One: Optimization of the “Join” Method.

- Nested Loop Join vs. Sort-Merge Join
- Depending on the circumstance, one is optimal.
- The optimizer will consider all paths and choose the path which has the lowest predicted cost
Discussion: Joins!!! DUH!

- It is interesting that the benefits of joins were finally recognized upon user feedback during phase 0 and then implemented in Phase 1.
- Why did they not implement it from the beginning if this concept was introduced by Codd?
Phase One: Security System

- Quite primitive with simply GRANT, REVOKE, and RESOURCE commands to users of the system to allow them manage the tables of the database.
Phase One: Recovery Subsystem

- Implemented solutions for:
  - Media failure (hard disk died)
    - Image dump and database log of “before” and “after” changes
  - System failure (information in main memory lost)
    - Log and usage of “shadow pages”
  - Transaction failure (query didn’t finish)
    - Log
Phase One: Locking Subsystem

- Originally proposed a predicate locking concept where tuples were locked based on a domain value.
  - Product = Aircraft
- Settled on a hierarchy locking concept with “intention” and “exclusion” locks.
Phase Two: Evaluation

- At this point, the response to System R had been excellent.
  - “...ease of installation, a high-level user language, and an ability to rapidly reconfigure the database.”
- SQL was demonstrated as a feasible and highly usable language for querying the data.
Phase Two: SQL

- The high level language to query the data was primarily successful:
  - Replaced Codd’s initially proposed Data Manipulation Language concept which was far too convoluted.
- Praised for the uniformity of the syntax regardless of the environment.
- Users suggested improvements:
  - EXISTS
  - LIKES
  - PREPARE AND EXECUTE
Discussion: EASY SQL... ahhh

- System R developed a usable high-level SQL language to allow easy user interaction including ad hoc queries.
- Has the success of relational database models been due to the simplicity of SQL?
Summary of System R

- The success of System R essentially proved that Codd’s idea were possible to implement.

- It showed that many of the problems that plagued old systems, could actually be solved by the relational model.

- Similar to Codd, they didn’t get everything right at first:
  - XRM access path method
  - Shadow pages for recovery
  - Predicate locking
Discussion: Have you been “relationalized”?

- Relational model has been around for such a long time with such a widespread penetration into almost all data storage market that Database experts, administrators and aspirants alike have "relationalized" their design approach.

- Can we break out of the shackles of relational model and think some different and better paradigm?

- “How have our application been molded from a relational model? What could we have done better if a graph or hierarchical model provided data independence?”
Thanks for listening! Any Questions?