Devere Anthony Weaver
COP4710 - Database Systems
Chapter 2 ER Textbook Notes

<u>**Ch.2 - Data and Data Models**</u>

<u>Coverage:</u>

- explore database models
- introduce relational database model

<u>**2.2 - File, Records, and Data Items**</u>

A database is mostly simply a repository of data about some specific entity

Two aspects to filing:

1. storage
2. retrieval

Each component of a record is called a *data item* or *field*

key - the field on which the file is organized

- the *key* should be unique for each record

format, file design - format of the file dictates the order of the fields in any record

- in the format or file design, the key is typically underlined

<u>Checkpoint 2.1</u>
** see Devere_ER2 document for questions 1, 2, and 3

4. What does *null* mean?

<u>Ans:</u> Null simply means a value does not exist for some field.

5. Given the customer file, what is wrong?

<u>Ans:</u> The fields for the records are not in the same order.

2.3 - Moving from 3x5 Cards to Computers

The natural progression from 3x5 cards was to use these as punch cards that could be sorted and read by a computer to input data and stored on magnetic media.

Early data files were simply “sequential files” where the file had to be sorted and redisplayed if an analyst wanted to see a listing of data by various fields. These were typically written in programming languages such as COBOL, FORTRAN, ALGO, etc.

Indexed systems -based on the maintenance of indexes for records rather than a strict sequential ordering of the records

relative record indexed system - system where one needed to know only which cylinders a file occupied and then the position of the record could be calculated relative to the starting position of the file

Database systems

- software packages that emerged in the late 60s
- purchasable programs that stored and retrieved data as well as performed maintenance (add, delete, modifying fields and records), no more need to write programs directly that handled these actions
- allows for sharing of data across departments

Sharing data reduces *redundancy* - storing the same information in different places.

Early database software evolved into two main models: *hierarchical* and *network.*

- Note: the relational model exists by this time but was treated as a good theoretical technique since computers weren’t fast enough to implement them at the time
- the only practical logical models were implemented on, at the time, supercomputers at research centers, they weren’t commercially viable yet

logical models - ways of logically perceiving the arrangement of data in a file structure

- analyst perceived how data was to be logically stored and then the database physically implements the logical model

Checkpoint 2.2.

1. What is a sequential file?

Ans: A file that contains records organized by the order in which they were entered. This order is fixed.

2. What is COBOL?

Ans: COBOL is an early programming language. It was often used to write programs that manipulated data in computer systems.

3. Why is ordering important in a sequential filing system? **

Ans: Ordering is important since these files must be read sequentially, unless a program was written to find a specific record.

4. What is a database program?

Ans: A database program is a software package that stores and retrieves data without the need to write complicated programs that manipulate the data.

5. In the early days, how was data put into a file?

Ans: Data was put into a file using punch cards that were read by computers. More advanced systems allowed for data to be stored programmatically by a programmer/analyst.

2.4 Database Models

The Hierarchical Model

Data are arranged in a top-down fashion in which some records have one or more “dependent” or “child” records and each child record is tied to one and only one “parent”.

If the same child occurs for a different parent, then the child will be recorded redundantly.

one-to-many relationship

header-trailer format

primary key and foreign key (FK)

- the primary key of a file indicates a specific record
- the foreign key refers to the primary key of a record in a different file
- this relationship only makes sense if there are two (or more) files that have some sort of connection

multiple-child pointer (MCP) / *parent pointer* - depicts the relationship between parent and child records in both directions

- allows questions to be asked of the system that are easier to answer than with just parent points
- allows file system to be audited with a computer program

Both of the above pointers can be implemented, but one might be better than the other in certain systems

- FK only keeps a parent reference
- MCP keeps a potentially much larger number of references

The Hierarchical Model with a Linked List:

This type of system uses a linked list to implement the relationship between parent and child.

The actual systems that existed using linked lists included forward/backward pointers, jumping of the chain, etc. not just pointers to the next node.

Relationship Terminology:

structural constraints

- *cardinality* - how many of one record type relate to the other and vice versa
    - *one to many (1:M)* - parent relates to multiple child nodes and child nodes can only have one parent node

    - *many to many (M:N)* - many employees relate to many dependents
- *optionality* - refers to whether one record may or must have a corresponding record in the other file
    - *optional or partial* - parent not required to have dependent in another file
    - *mandatory or full* - parent must have dependent on file

** This material will be returned to in a later chapter in more detail, but the point is that if the description of the database design is clear then the outcome is more predictable. **

<u>Drawbacks of the Hierarchical Model:</u>

Relationships between records in this model have cardinality of 1:M or 1:1 but never M:N.

When implementing the linked list version, the system is fragile in that if one dependent record is destroyed, then the entire chain of links is broken for a given record.

3 Major Drawbacks:

1. not all situations can be modeled by the 1:M format
2. linking of parent and child nodes have impacts on performance of the system
3. linking of parents and child records is done physically, if the *dependent* file were reorganized, all pointers would have to be reset

2.5 - The Network Model

The network model inherently supports the M:N cardinality.

repeating group - part of the file description that can have multiple values

A network model has added complexity when compared to a hierarchical model since there are forward and backward pointers.

Again, since pointers are implemented, a hardware-implemented connection will have an impact on performance (positive or negative). Also, network models are more complex which means they can grow more complicated to maintain and manage.

<u>2.6 - The Relational Model</u>

In 1970 Codd introduced the relational model that didn't suffer the drawbacks of physical linkage and hardware-bound restrictions.

The idea here is that one should ignore the way in which data files are connected and arrange data into a simple two dimensional, unordered table. Doing so would allow for a calculus for queries and one can focus on the data as data.

normal forms and functional dependencies

- these concepts are explored in the next chapter