CS561: Database Management Systems Notes

Steven DeFalco

Fall 2023

Contents

1 Introduction

Database management systems (DBMS) consist of **data**, **software** (programs such as data interfaces), and **environments** (operating systems). DBMS contains information about a particular enterprise. They are collections of interrelated data, a set of programs to access the data, and an environment that is both *convenient* and *efficient* to use. The *user* should only have to define what it is that they want from the database, whereas the *database* is responsible for defining how this query can be fulfilled; relational databases are good at this.

The three primary data models are ***entity-relationship model*** (diagrams), ***relational model*** (relational algebra), ***relational database*** (SQL).

Drawbacks to using **file systems** to store data include the following:

- data *redundancy* and *inconsistency* (multiple formats, duplication of information, etc.)
- difficulties in *accessing* data
- data isolation (multiple files and formats)
- concurrency issues (among multiple users)
- *integrity* problems
- atomicity of updates
- security problems (hard to provide varied levels of user access)

There are varying ***levels of abstraction*** in a database. The ***physical level*** defines how a record is stored. The ***logical level*** describes data stored in the database and the relationships among data. The ***view level*** is a way to hide details of data types and information for security purposes.

The ***schema*** is the logical structure of the database; this is analagous to type information of a variable in a program. ***Physical schema*** refer to database design at the physical level. ***Logical schema*** refer to database design at the logical level. An ***instance*** is the actual content of the database at a particular time; this is analagous to the value of a variable. ***Physical data independence*** is the ability to modify the physical schema without changing the logical schema.

Data manipulation languages (DML) are languages for accesing and manipulating the data organized in a DBMS. ***Procedural languages*** are ones in which the user specifies what data is required and how to get that data. ***Declarative (nonprocedural) languages*** are ones in which the user specifies what data is required without specifying how to get such data. **SQL** is the most

widely used query language.

A ***data definition language (DDL)*** is the specific notation for defining the database schema. The ***DDL compiler*** generates a set of tables stored in a data dictionary. Data dictionary contains metadata.

A ***relational database*** is based on the relational data model. Data and relationships among the data are represented by a collection of tables. These include both a **DML** and **DDL**. The most common relational database systems employ the **SQL** query language.

2 Entity-Relationship Model

A *database* can be modeled as a collection of entities or a relationship among entities. An ***entity*** is an object that exists and is distinguishable from other objects (e.g. specific person, company, even, plant). These *entities* have ***attributes*** (e.g. people hvae names and addresses). An ***entity set*** is a set of entities of the same type that share the same properties (e.g. set of all persons, companies). In the ER-model we refer to specific objects as *entities* which have *attributes* and are all a part of the entire *entity set*.

A ***relationship*** is an association among several entties. A ***relationship set*** is a mathematical relation among $n \geq 2$ entties, each taken from entity sets.

$$\{(e_1, e_2, \ldots e_n) | e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

where $(e_1, e_2, \ldots e_n)$ is a relationship.

2.1 Attributes

An ***attribute*** can also be property of a relationship set. For instance, the *depositer* relationship set between entity sets *customer* and *account* may have the attribute *access-date*. Relationshup sets that involve, two entity sets are ***binary*** (degree two). Relationship sets may involve more than two entity sets. Relationships between more than two entity sets are rare (i.e. most are binary).

An ***entity*** is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set. ***Domain*** is the set of permitted values for each attribute. The **types of attributes** include the following:

- *simple* (atomic) and *composite* attributes
- *single-valued* and *multi-valued* attributes
- *derived* attributes (can be computed from other attributes)

When attributes are *simple* and *single-valued*, then we say that the data is in ***First Normal Form***.

2.2 Keys

A ***super key*** of an entity set is a set of one or more attributes whose values uniquely determine each entity. Once you have defined a *super key*, you can add attributes and it is still considered a *super key*. A ***candidate key*** of an entity set is a minimal super key (e.g. *cusomer_id* is a candidate key of *customer*). Candidate keys *only* contain the necessary attributes to make something unique. Although several candidate keys may exist, one of the candidate keys is selected to be the ***primary key***.

The combination of primary keys of the participating entity sets forms a super key of relationship set. This means a pair of entity sets can have at most one relationship for each access.

2.3 E-R Diagrams

Rectangles represent entity sets. Diamonds represent relationship sets. Lines link attributes to entity sets and entity sets to relationship sets. Ellipses represent attributes: double ellipses represent multivalued attributes while dashed ellipses denote derived attributes. Underline indicates primary key attributes.

3 Relational Model

Formally, given sets $D_1, D_2, \ldots D_n$ a ***relation*** r is a subset of

$$D_1 \times D_2 \times \ldots \times D_n$$

Thus, a relation is a set of n-uples $(a_1, a_2, \ldots a_n)$ where each $a_i \in D_i$.

Each attribute of a relation has a name. The set of allowed valued for each attribtue is called the ***domain*** of the attribute. Attribute values are (normally) required to be ***atomic***; that is, indivisible. Domain is said to be atomic if all its members are atomic. The special value *null* is a member of every domain. The null value causes complications in the definition of many operations.
$A_1, A_2, \ldots A_n$ are attributes. $R = (A_1, A_2, \ldots, A_n)$ is a ***relation schema***. $r(R)$ denotes a relation r on the relation schema R.

The current values (***relation instance***) of a relation is specified by a table. An element t of r is a *tuple*, represented by a row in a table. Relations are unordered and thus the roder of tuples is irrelevant (tuples may be stored in an arbitrary order).

A database consists of multiple relations. Information about an enterprise is broken up into parts, with each relation storing one part of the information.