CSC 355 Database Systems
Lecture 2

Eric J. Schwabe

School of Computing, DePaul University

Spring 2020

Topics:

- The Relational Model

Data Models

- A *data model* describes three things about stored data:
  - Structure of the data: How is the data organized?
  - Operations on the data: What can be done with the data?
  - Constraints on the data: How are the data restricted?

Relational vs. Semi-structured

- From Ullman/Widom:

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

Figure 2.1: An example relation

```
<Movies>
    <Movie title="Gone With the Wind">
        <Year>1939</Year>
        <Length>231</Length>
        <Genre>drama</Genre>
    </Movie>
    <Movie title="Star Wars">
        <Year>1977</Year>
        <Length>124</Length>
        <Genre>sciFi</Genre>
    </Movie>
    <Movie title="Wayne's World">
        <Year>1992</Year>
        <Length>95</Length>
        <Genre>comedy</Genre>
    </Movie>
</Movies>
```

Figure 2.2: Movie data as XML

Relational vs. Semi-structured

- Relational Model:
  - Structure: Two-dimensional tables with links
  - Operations: Relational algebra / SQL
  - Constraints: Data domains, uniqueness,…
- Semi-structured Model:
  - Structure: Nested XML elements
  - Operations: Element traversal, searching
  - Constraints: Data domains, nesting restrictions,…

The Relational Model

- Introduced by E.F.Codd in 1970
- First model that separated the logical organization of the data from its physical implementation
- Model is based on formal logic and the relational algebra

The Relational Model

- Data is stored in two-dimensional tables called *relations*, each one having a name
- Each row is a *tuple* representing one instance of the entity the table represents
- Each column is an *attribute*, representing a property for which each instance has a value
- Every *component* in a tuple must have a value taken from its attribute's associated *domain*

Relation Example

- Name: EMPLOYEE
- Tuples: { (100, Margaret Simpson, Marketing, 48,000) ,
  (140, Allen Beeton, Accounting, 52,000.00) ,
  (110, Chris Lucero, Info Systems, 43,000.00) ,
  …and three more… }
- Attributes: ID, Name, Dept, Salary
- Domains: (Three-digit) integer, string (of length at most 20), string (of length at most 12), decimal number (with at most five digits to the left of the decimal point and two to the right)
  - Note that domains can be a little complicated to describe...

Properties of Relations

1. Each relation has a unique name (in database)
2. Each attribute has a unique name (in relation)
3. Each entry of a relation contains a single value from its attribute's domain (or NULL)
4. The order of the records does not matter
5. The order of the attributes does not matter
6. No two records in a relation are identical

Relation Schema vs. Instance

- The *schema* of a relation consists of the name of the relation followed by a list of its attributes (domains may be included also…)
  - EMPLOYEE (ID , Name , Dept , Salary) …
  - EMPLOYEE (ID:number , Name:string , Dept:string , Salary:number) … ?
  - EMPLOYEE (ID:integer(3) , Name:string(20) , Dept:string(12) , Salary:number(5.2)) … ?

Relation Schema vs. Instance

- An *instance* of a relation is a set of tuples, where each tuple contains a value for each attribute (from the associated domain), or perhaps NULL (indicating no value)
  - DBMS enforces these "domain constraints"
- Instances are often presented as tables rather than as sets, but the order of the rows in these tables is not significant

Candidate Keys

- A *candidate key* is a set of attributes for which each tuple in the relation must have a unique set of values ("key constraints"), and for which no subset of the set has this property
  - The book calls these just *keys*, but I will use the more specific term *candidate keys* since there are different types of keys…
- This property must hold for <u>all possible relation instances</u> for it to be a candidate key

Primary Keys

- One of the candidate keys can be chosen as the *primary key* for the relation
- A relation may have many candidate keys, but can have only one primary key
- The primary key will be underlined in the schema
  - A primary key must have a unique set of values in each tuple, and may not contain any NULL values in any tuple ("entity integrity")

Foreign Keys

- We link two relations using a shared key that is the primary key in one of the relations
- In the other relation, this key is called a *foreign key* (dotted underline in schema, with arrow to corresponding primary key)
  - Every value of the foreign key must be the value of the corresponding primary key in some tuple
  - Thus the foreign key associates each tuple with exactly one tuple in the relation that it references

Referential Integrity

- Every foreign key value must appear as the value of the primary key in some row of the table it references
- This restricts the changes that can be made:
  - We can add a row containing a foreign key only if the value of the foreign key appears among the values of the referenced primary key
  - We can remove a row containing a primary key only if the value of the primary key does not appear among the values of any referencing foreign key

Constraints

- All of these will be maintained by the DBMS:
  - Domain constraints: In every tuple, the value of each attribute must come from its specified domain
  - Key constraints: Each tuple must have a unique set of values in each of its candidate keys
  - Entity integrity: Each tuple must have a unique set of values in its primary key, and not any NULLs
  - Referential integrity: Every foreign key value must appear as the value of the primary key in some tuple of the relation it references

Database Schema vs. Instance

- The *schema* of a relational database consists of:
  - The schema of each relation in the database
    - Name, list of attributes (and maybe domains)
    - Primary keys and foreign keys underlined
  - An arrow from each foreign key to the primary key it references

Database Schema vs. Instance

- An *instance* of a relational database consists of:
  - An instance of each relation in the database, where each relation instance satisfies all the required constraints:
    - domain constraints, key constraints, entity integrity, referential integrity (and any user-defined constraints)
  - The only changes allowed by the DBMS are those that result in another valid instance…

Next:

- SQL Data Definition Language