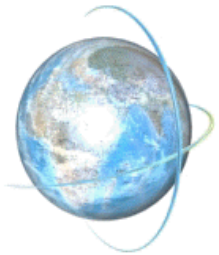
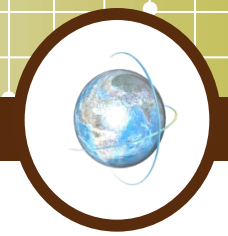


Session 2

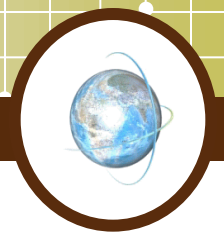
Chapter 2 - Relational Model of Data - Part 1

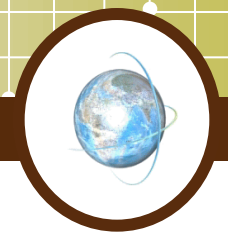
Objectives



1. Understand what is a Data Model
2. Understand what belong to a Data Model
3. Understand what are Relations, Attributes, Tuples, Domains
4. Understand what are Relation instances, Schema, DB schema

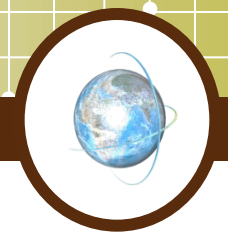
Contents





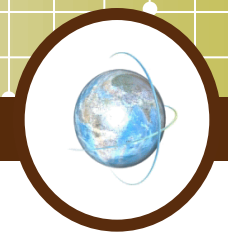
2.1. An Overview of Data Models

2.1.1. What is a Data Model?



- A Data Model is a notation for describing data or information. The description generally consists of 3 parts:
  - Structure of the data
  - Operations on the data.
  - Constraints on the data.

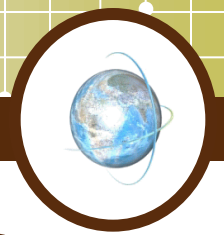
2.1.2.Important Data Models



Today, two important data models are:

- The relational model, including object-relational extensions
- The semi-structured data model, including XML and related standards

2.1.3. The Relational Model in Brief

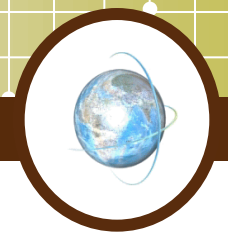


- The relational model is based on tables where:
  - Column headers are field names
  - Each row represents the values of one record

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

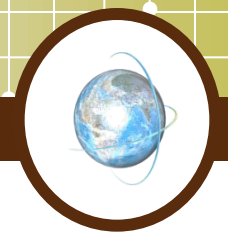
A relation of "Movies" data

2.1.4. The Semistructured Model in Brief



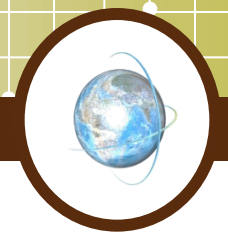
- Semistructured data is based on trees or graphs, rather than tables or arrays.
- The principal manifestation of this kind is XML: a way to represent data by hierarchically nested tagged elements where the tags are similar to column headers in the relational model.

Movie data as XML

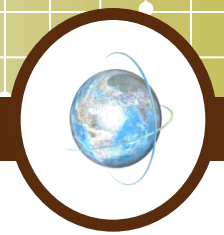


```
<Movies>
    <Movie title="Gone With the Wind">
        <Year>1939</Year>
        <Length>231</Length>
        <Genre>drama</Genre>
    </Movie>
    <Movie title="Star Wars">
        <Year>1977</Year>
        <Length>124</Length>
        <Genre>sciFi</Genre>
    </Movie>
    <Movie title="Wayne's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

2.1.5. Other Data Models

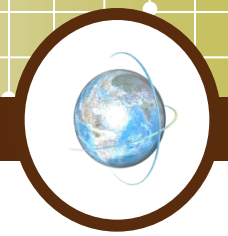


- Object-relational model:
  - Values can have structure, rather than being elementary type such as string or integer.
  - Relations can have associated methods.



2.2. Basics of Relational Model

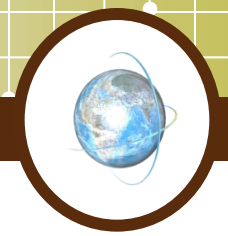
2.2.1. Attributes



- The relational model represents data as a 2-dimensional table (called a relation)
- In the relation “Movies”:
  - Each row represents a MOVIE
  - Each column represent a property of MOVIES and also called a “**attribute**”

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

2.2.2. Schemas

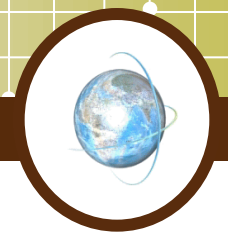


- The name of a relation and the set of attributes for that relation is called the *schema* for that relation, *e.g.*

  MOVIES (title, year, length, genre)

- The attributes in a relation schema are a set, not a list.
- The set of schemas for the relations of a database is called a *relational database schema*, or just a *database schema*.

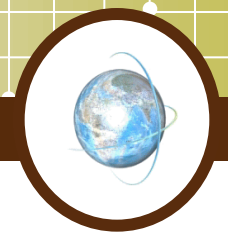
2.2.3. Tuples



- A row of a relation is called a *tuple* (or *record*)
- A tuple has *one component* for each attribute of the relation
- When we want to write a tuple in isolation, not as part of a relation, we normally use commas to separate components.
- *E.g.*: (Gone with the wind,1939, 231, drama)

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

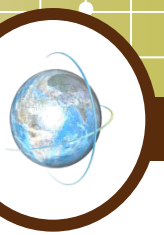
2.2.4. Domains



- The relational model requires that each component of each tuple must be atomic, that is, it must be of some elementary type such as INTEGER or STRING
- It is not permitted for a value to be a record structure, set, list, array or any type that can have its values broken into smaller components
- A Domain is a particular elementary type of a attribute, we could include the domain for each attribute in a relation schema as follows:

```
Movies(title:string, year:integer, length:integer, genre:string)
```

2.2.5. Equivalent representations of a relation

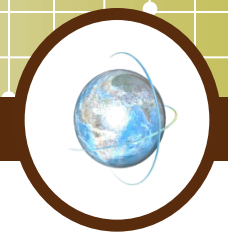


- Relations are **sets** of tuples, **not lists** of tuples
- So, the order in which the tuples of a relation are presented is not important
- We can reorder the attributes of a relation without changing the relation

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

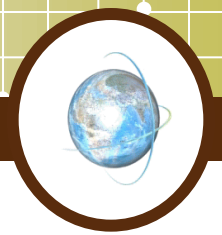
Another representation of relation "Movies"

2.2.6. Relation instances



- A relation about **Movies** is not static but changing over time. We may want to:
  - Insert tuples for new Movies, as these appear.
  - Edit existing tuples if we get corrected information about a Movie.
  - Delete a tuple from the database for some reason
- A set of tuples for a given relation is called an *instance* of that relation
  - The set of tuples that are in the relation “now” is called the *current instance*.

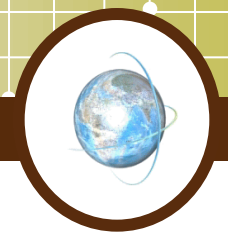
2.2.7. Keys of relations



- A set of attributes forms a *key* for a relation if we don't allow 2 tuples in a relation instance to have the same values in all the attributes of the key
  - the relation Movies has a key consisting of the two attributes title and year.
- We indicate attributes forming a key for a relation by underlining them

  Movies (<u>title</u>, <u>year</u>, length, genre)

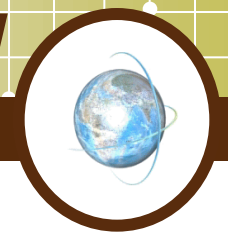
2.2.8. Example of database schema about movies



```
Movies(
    title:string,
    year:integer,
    length:integer,
    genre:string,
    studioName:string,
    producerC#:integer
)
MovieStar(
    name:string,
    address:string,
    gender:char,
    birthdate:date
)
StarsIn(
    movieTitle:string,
    movieYear:integer,
    starName:string
)
```

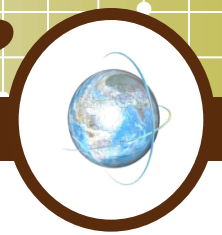
```
MovieExec(
    name:string,
    address:string,
    cert#:integer,
    netWorth:integer
)
Studio(
    name:string,
    address:string,
    presC#:integer
)
```

Summary 1: Relational Model



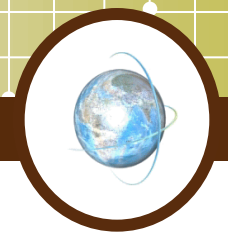
- Table = *relation*.
- Column headers = *attributes*.
- Row = *tuple*
- *Relation schema* = name(attributes) + other structure info., e.g., keys, other constraints. Example: Beers(name, manf)
  - Order of attributes is arbitrary, but in practice we need to assume the order given in the relation schema.
- *Relation instance* is current set of rows for a relation schema.
- *Database schema* = collection of relation schemas.

Summary 2: Why Relations?



- Very simple model.
- *Often* a good match for the way we think about our data.
- Abstract model that underlies SQL, the most important language in DBMS's today.
  - But SQL uses "bags" while the abstract relational model is set-oriented.

Exercises



acctNo	*type*	*balance*
12345	savings	12000
23456	checking	1000
34567	savings	25

The relation **Accounts**

firstName	*lastName*	*idNo*	*account*
Robbie	Banks	901-222	12345
Lena	Hand	805-333	12345
Lena	Hand	805-333	23456

The relation **Customers**

Figure 2.6: Two relations of a banking database

a) The attributes of each relation.

b) The tuples of each relation.

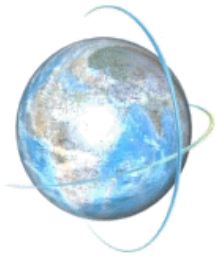
c) The components of one tuple from each relation.

d) The relation schema for each relation.

e) The database schema.

f) A suitable domain for each attribute.

g) Another equivalent way to present each relation.



Thank You !