The Relational Data Model

Why Relational Model?

• Most of the current DBMS are based on the relational data model:
  – simplicity
  – mathematically based:
    • expressions (queries) can be analyzed by DBMS
    • transformed to equivalent expressions automatically (query optimization)

Basics

• The relational data model is a particular way of structuring data (relations)
• Each relation is a two-dimensional table:

<table>
<thead>
<tr>
<th>color</th>
<th>title</th>
<th>year</th>
<th>length</th>
<th>filmType</th>
</tr>
</thead>
<tbody>
<tr>
<td>95</td>
<td>Wayne's World</td>
<td>1992</td>
<td>95</td>
<td>color</td>
</tr>
<tr>
<td>104</td>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>color</td>
</tr>
<tr>
<td>124</td>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>color</td>
</tr>
</tbody>
</table>

The relation Movies

Attributes, Schemas, and Tuples

Attributes

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>length</th>
<th>filmType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>color</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>color</td>
</tr>
<tr>
<td>Wayne's World</td>
<td>1992</td>
<td>95</td>
<td>color</td>
</tr>
</tbody>
</table>

Tuple t.

Schema: Movies(title, year, length, filmType)

Domains

Each attribute is associated to a domain.

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>length</th>
<th>filmType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>color</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>color</td>
</tr>
<tr>
<td>Wayne's World</td>
<td>1992</td>
<td>95</td>
<td>color</td>
</tr>
</tbody>
</table>

The relation Movies

Schema: Movies(title, year, length, filmType)

Requirements

• Each component of a tuple is atomic (similar to atomic requirement in E/R diagram)
• Each attribute is associated with a domain
Equivalent representation of a Relation

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>length</th>
<th>filmType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>color</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>color</td>
</tr>
<tr>
<td>Wayne's World</td>
<td>1992</td>
<td>95</td>
<td>color</td>
</tr>
</tbody>
</table>

IS THE SAME AS

<table>
<thead>
<tr>
<th>year</th>
<th>length</th>
<th>title</th>
<th>filmType</th>
</tr>
</thead>
<tbody>
<tr>
<td>1977</td>
<td>124</td>
<td>Star Wars</td>
<td>color</td>
</tr>
<tr>
<td>1991</td>
<td>104</td>
<td>Mighty Ducks</td>
<td>color</td>
</tr>
<tr>
<td>1992</td>
<td>95</td>
<td>Wayne's World</td>
<td>color</td>
</tr>
</tbody>
</table>

Movies(title,year,length,filmType) is equivalent to Movies(year,length,title,filmType)

Relation Instance

- A set of tuples for a relation is an instance. **NOTE**: An instance is **not** a schema.
- Instance changes when
  - a new tuple is added
  - a tuple is deleted
  - some attributes of a tuple are modified
- Relation schema does not often change but could be changed; when it changes lot of things need to be done – what?

Some Exercises

<table>
<thead>
<tr>
<th>acctNo</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>savings</td>
<td>1200</td>
</tr>
<tr>
<td>23456</td>
<td>savings</td>
<td>100</td>
</tr>
<tr>
<td>12347</td>
<td>checking</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>firstName</th>
<th>lastName</th>
<th>idNo</th>
<th>account</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robie</td>
<td>Banks</td>
<td>000-111</td>
<td>12345</td>
</tr>
<tr>
<td>Lena</td>
<td>Hood</td>
<td>111-222</td>
<td>23456</td>
</tr>
<tr>
<td>Lennon</td>
<td>John 3</td>
<td>222-333</td>
<td>12347</td>
</tr>
</tbody>
</table>

Steps in designing a database (Remember?)

- Analysis:
  - What information needs to be stored?
  - What are the relationships between different components of the stored information?
  - What is the suitable database structure (or schema)?
- Design the database structure (using a database design language or notation suitable for expressing design)
- Implementation in DBMS once committed to the design

From E/R Diagrams to Relational Designs

- Why?
  - We often start out with a E/R diagram and then convert to a relational model
- How?
  - Relation with the same set of attributes
  - Relation are the keys of the connecting entity sets

What about weak entity set and isa relationships? Can it be simplified?
**Entity sets to relations**
- Entity set $E$ with attributes $a_1, \ldots, a_n$, which is not a weak entity set is converted to a schema $E(a_1, \ldots, a_n)$
- Examples:
  - Movies(title, year, length, filmType)
  - Stars(name, address)
  - Studios(name, address)

**Relationships to relations**
- $R$ is a relationship is converted to a relation with
  - key attributes of the involved entity sets (those connecting to $R$)
  - attributes of $R$
- Examples:
  - Stars_in(title, year, name)
  - Owns(title, year, name)
- What happens if an entity set $E$ is involved in $R$ more than one time?
  - the key attributes of $E$ have to appeared as many times as $E$ is involved in $R$; rename the attributes of $E$ each time they are added

**Example**

```
Movies

Contracts

Stars

Producing studio

Studios

Contracts(title, year, name, name_producing_studio, name_studio_of_star)
```

**Combining Relations**
- So far: rules for converting E/R diagrams to relations
- Sometimes: not optimal

```
F(A, B)
R(A, C)
E(C, D)

Because C determines A  
ER(C,D,A)
```

**Combining Relations**
- $E$ is an entity set, $R$ is a many-one relationship from $E$ to $F$
- $E$ and $R$ can be combined into one relation
- Attributes of the relation that combines $E$ and $R$ consists of
  - all attributes of $E$
  - the key attributes of $F$
  - all attributes of $R$
- This cannot be done if $R$ is a many-many relationship

**Weak Entity Sets to Relations**
- $W$ – weak entity set: three components
  - entity set $W$
  - relationship from $W$ to another entity set (single border)
  - supporting relationship (double border)
- the relation corresponds to $W$ have the following attributes:
  - attributes of $W$
  - attributes of other entity sets that help form the key of $W$
- if $R$ is a relationship connecting $W$ then attributes of $R$ must contain the key attributes of $W$
- if $R$ is a supporting relationship from $W$ to another entity set then no relation corresponds to $R$ is needed
Example

Crews(name, number) & Studios(name, address)

Unit_of(number, name, nameStudio)

Redundant

Subclasses to Relations

- Assumptions about isa-hierarchy:
  - there is a root entity set for the hierarchy
  - the root’s key identifies every entity in the hierarchy
  - an entity set in the hierarchy might have attributes belonging to the different entity sets in the hierarchy

- Several choices:
  - E/R viewpoint: each entity set in the hierarchy corresponds to a relation
  - Object-oriented viewpoint: each subtree corresponds to a relation
  - Use null values: the whole tree corresponds to a relation

E/R:

Relations correspond to Movies, Cartoons, Murder_Mysteries, Children, Story_Books

Object-oriented:

Relations correspond to the subtrees with the root: e.g. Cartoons

Null values:

One relation: Movies

E/R style conversion

Movies(title, year, length, filmType)
MurderMysteries(title, year, weapon)
Cartoons(title, year, weapon)

No relations for isa

OO conversion

Movies(title, year, length, filmType)
MoviesMurderMysteries(title, year, length, filmType, weapon)
MoviesCartoons(title, year, length, filmType)
MoviesAll(title, year, length, filmType, weapon)

No relations for isa

Null values

Movies(title, year, length, filmType, weapon)
No relations for isa
Comparison

• It depends, again, because each approach has its own advantages and disadvantages!
• Query answering:
  – Null values: good for queries related to several entity sets in the hierarchy
  – E/R: good for queries related to one single entity
  – OO: good for queries related to a subhierarchy
• Number of relations: null – only one, E/R – number of entities, OO – exponential
• Space: OO is worse, depending on specific situation: null more or less E/R

Relational Design

• So far:
  – how to get relational designs from E/R diagrams?
  – different approaches produce different relational designs
  – each design has advantages and disadvantages
• Question:
  – Can we do relational designs directly from applications?
  – Can we improve the relational designs? If yes, how?

Improving Relational Designs

• Using functional dependencies: redesign of relations, remove redundancy
• Multivalued dependencies and integrity constraints: create good database schemas

Functional Dependencies (FD)

• Definition:
  A functional dependency on a relation R is a statement of the form “if two tuples of R agree on attributes \( A_1, \ldots, A_n \), then they must also agree on other attributes \( B_1, \ldots, B_m \).”

• Notation:
  \( A_1, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m \)

• Shorthand:
  \( A_1, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m \)

  If the A's of t equal the A's of u, then the B's of t must equal the B's of u

Example

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>length</th>
<th>studioName</th>
<th>starName</th>
<th>filmType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Fox</td>
<td>Harrison Ford</td>
<td>color</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Fox</td>
<td>Carrie Fisher</td>
<td>color</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>Disney</td>
<td>Emilio Estevez</td>
<td>color</td>
</tr>
<tr>
<td>Wayne's World</td>
<td>1992</td>
<td>95</td>
<td>Paramount</td>
<td>Dana Carvey</td>
<td>color</td>
</tr>
</tbody>
</table>

Some of the FDs:

- title year \( \rightarrow \) length studioName starName filmType

Shorthand: title year \( \rightarrow \) length filmType studioName starName

Note: shorthand is used to combine FDs with the same left side.

Keys of Relations

• Definition: \( \{A_1, \ldots, A_n\} \) is a key for a relation R if
  – \( \{A_1, \ldots, A_n\} \) functionally determines all other attributes of R, and
  – no subset of \( \{A_1, \ldots, A_n\} \) functionally determines all other attributes of R.

• What is the key for

Movies(title, year, length, filmType, studioName, starName)?
**Superkeys**

- A set of attributes that contains a key is called a superkey.
  - every key is itself a superkey
  - a superkey needs not be a key (not minimal)

**Example**

Movies(title, year, length, filmType, studioName, starName)

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>length</th>
<th>studioName</th>
<th>starName</th>
<th>filmType</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Disney</td>
<td>Harrison Ford</td>
<td>color</td>
<td></td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Fox</td>
<td>Carrie Fisher</td>
<td>color</td>
<td></td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>Disney</td>
<td>Emilio Estevez</td>
<td>color</td>
<td></td>
</tr>
<tr>
<td>Wayne's World</td>
<td>1992</td>
<td>95</td>
<td>Paramount</td>
<td>Dana Carney</td>
<td>color</td>
<td></td>
</tr>
</tbody>
</table>

FDs: title, year, length, filmType, studioName

Keys?
- (title, year) NO, cannot functionally determine ‘starName’
- (title, year, starName) YES


**Rules about Functional Dependencies**

- FDs are good for database schema design
- knowledge of FDs needed
- FDs determine the set of legal instances of a database schema
- Rules about FDs: allow us to reason about FDs
  - checking whether a FD holds
  - construct new FDs

**Finding Keys for Relations**

- Relations are obtained from E/R diagrams
- Key for a relation R can be determined as follows:
  - R is obtained from an entity set E: a key of E is a key for R
  - from a binary relationship:
    - many-many from E to F: a key for R is the union of a key of E and a key of F
    - many-one from E to F: a key of E is a key for R
    - one-one from E to F: a key of E or a key of F can be used as a key for R
  - from a multiway relationship: situation dependent and need to be considered carefully

**Example**

Movies(title, year, length, filmType, studioName, starName)

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>length</th>
<th>studioName</th>
<th>starName</th>
<th>filmType</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Disney</td>
<td>Harrison Ford</td>
<td>color</td>
<td></td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Fox</td>
<td>Carrie Fisher</td>
<td>color</td>
<td></td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>Disney</td>
<td>Emilio Estevez</td>
<td>color</td>
<td></td>
</tr>
<tr>
<td>Wayne's World</td>
<td>1992</td>
<td>95</td>
<td>Paramount</td>
<td>Dana Carney</td>
<td>color</td>
<td></td>
</tr>
</tbody>
</table>

FDs: title, year, length, filmType, studioName

Keys?
- (title, year) NO, cannot functionally determine ‘starName’
- (title, year, starName) YES

Notation: underlining attributes to specify the primary key in relation schema.
Rules about FDs

- Two sets of FDs $S$ and $T$ are *equivalent* if the set of relation instances satisfying $S$ is exactly the set of relation instances satisfying $T$.
- A set of FDs $S$ follows from a set of FDs $T$ if every relation instance that satisfies all the FDs in $T$ also satisfies all the FDs in $S$.

Trivial FDs

- Given $A_1 \ldots A_n \rightarrow B_1 \ldots B_m$:
  - *trivial*: if $B_j \in \{A_1 \ldots A_n\}$ for $j=1,\ldots,m$
  - *nontrivial*: some $B_j$ does not belong to $\{A_1 \ldots A_n\}$
  - *completely nontrivial*: none of the $B_j$ belongs to $\{A_1 \ldots A_n\}$

Closure of Attributes

- Given:
  - a relation $R$ with the set of attributes $\{A_1 \ldots A_n\}$ (also written as: $R(A_1 \ldots A_n)$)
  - a set $S$ of FDs
- Question: does $B_1 \ldots B_n \rightarrow B$ hold?
- How can we answer this question? Answer: Math!
- Compute the closure of $B_1 \ldots B_n$ and then check if $B$ is in the closure then yes; otherwise, no.