Refresher: ER-modeling, logical relational model, dependencies

Toon Calders
toon.calders@ulb.ac.be
Different Levels

• Conceptual level: ER-diagrams

• Logical level: Relations, attributes, schemas, primary keys, foreign key dependencies

• Physical level: Storage model, partitions, indices, triggers, ...
Conceptual Level: ER-Diagram

• Expresses entities and relations between them

```
<table>
<thead>
<tr>
<th>doctor</th>
<th>(0,n)</th>
<th>Primary doctor</th>
<th>(0,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DID</td>
<td></td>
<td>name</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>specialization</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>patient</th>
<th>(0,n)</th>
<th>prescription</th>
<th>(0,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td></td>
<td>Nr</td>
<td></td>
</tr>
<tr>
<td>fname</td>
<td></td>
<td>product</td>
<td></td>
</tr>
<tr>
<td>sname</td>
<td></td>
<td>quantity</td>
<td></td>
</tr>
<tr>
<td>address</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

by

since

for
```
Conceptual Level: ER-Diagram

- Expresses entities and relations between them

**Entity**: 
- **doctor**: 
  - DID
  - name
  - specialization
  - (0,n)
  - attributes
  - Cardinality constraint: (0,n)
  - Primary key

- **patient**: 
  - SSN
  - fname
  - sname
  - address
  - (0,1)
  - Attribute of the relation
  - (0,n)

- **prescription**: 
  - Nr
  - product
  - quantity
  - (1,1)
  - (0,1)

**Relation**: 
- Primary doctor
  - (0,1)
  - Attributes: since
  - (0,n)

- by
  - (1,1)
  - for
  - (0,1)
Conceptual Level: ER-Diagram

- Expresses entities and relations between them

**doctor**
- DID
- name
- specialization

**patient**
- SSN
- fname
- sname
- address

**prescription**
- Nr
- product
- quantity

**Primary doctor**
- (0,n)
- (0,1)

**since**
- (1,1)

**by**
- (0,n)

**Identifying relation**
- partial key
- weak entity

**Total participation**
- 1 to many relation
Different Notations

- **doctor**
  - DID
  - name
  - specialization
  - (0,n) relationship with **patient**

- **patient**
  - SSN
  - fname
  - surname
  - address
  - (0,1) relationship with **doctor**
  - since

- **prescription**
  - Nr
  - product
  - quantity
  - (1,1) relationship with **by**
  - for
  - (0,1) relationship with **by**
Different Notations

- **doctor**
  - (0,n) relationship with **patient**
  - name
  - specialization

- **prescription**
  - (1,1) relationship with **doctor**
  - (0,1) relationship with **patient**
  - nr
  - product
  - quantity

- **patient**
  - (0,n) relationship with **doctor**
  - SSN
  - fname
  - sname
  - addr
Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars, and has one or more premium payments associated with it. Each payment is for a particular period of time, and has associated due date, and the date the payment was received.
Logical Model: Relational

• A schema is a set of attributes
• Domain of attribute A: $\text{dom}(A)$
• Tuple $t$ over schema $S$: mapping from $S$ to values; for all $A \in S$, $t(A) \in \text{dom}(A)$
• Relation $R(S)$: (finite) set of tuples over $S$
• Database $D$ is a set of relations
# Relational Database: Example

## Courses

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ID45</td>
<td>Advanced Databases</td>
</tr>
<tr>
<td>2ID05</td>
<td>Databases I</td>
</tr>
</tbody>
</table>

## Offerings

<table>
<thead>
<tr>
<th>Code</th>
<th>Semester</th>
<th>Lecturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ID45</td>
<td>Spring 2011</td>
<td>Calders</td>
</tr>
<tr>
<td>2ID05</td>
<td>Spring 2011</td>
<td>Fletcher</td>
</tr>
</tbody>
</table>

## Follows

<table>
<thead>
<tr>
<th>Student</th>
<th>Code</th>
<th>Semester</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phil</td>
<td>2ID45</td>
<td>Spring 2011</td>
<td>A+</td>
</tr>
<tr>
<td>Mary</td>
<td>2ID45</td>
<td>Spring 2011</td>
<td>C</td>
</tr>
<tr>
<td>John</td>
<td>2ID45</td>
<td>Spring 2011</td>
<td>B-</td>
</tr>
<tr>
<td>Paul</td>
<td>2ID05</td>
<td>Spring 2011</td>
<td>C</td>
</tr>
</tbody>
</table>
Relational Model: Keys

• Set of attributes $K \subseteq S$ is a superkey for relation $R(S)$ if: for every legal instance $r$ of $R$, and for every $t_1, t_2 \in r$, $t_1(K) = t_2(K)$, then $t_1 = t_2$

• $K$ is a candidate key if $K$ is a superkey and $K$ is minimal (no strict subset is a key)

• In the logical model, we chose one of the candidate keys as the primary key.
Keys: Example

<table>
<thead>
<tr>
<th>Student</th>
<th>Code</th>
<th>Name</th>
<th>Semester</th>
<th>Lecturer</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phil</td>
<td>2ID45</td>
<td>Advanced Databases</td>
<td>Spring 2011</td>
<td>Calders</td>
<td>A+</td>
</tr>
<tr>
<td>Mary</td>
<td>2ID45</td>
<td>Advanced Databases</td>
<td>Spring 2011</td>
<td>Calders</td>
<td>C</td>
</tr>
<tr>
<td>John</td>
<td>2ID45</td>
<td>Advanced Databases</td>
<td>Spring 2011</td>
<td>Calders</td>
<td>B-</td>
</tr>
<tr>
<td>Paul</td>
<td>2ID05</td>
<td>Databases I</td>
<td>Spring 2011</td>
<td>Fletcher</td>
<td>C</td>
</tr>
</tbody>
</table>

- Some superkeys:
  - {Student, Code, Name, Semester}
  - {Student, Code, Semester, Lecturer}
- Only candidate Key: {Student, Code, Semester}
Functional Dependencies

- X, Y ⊆ S Functional dependency X→Y holds in relation R(S) if: for every legal instance r of R, and for every t₁, t₂ ∈ r, if t₁(X)=t₂(X), then t₁(Y)=t₂(Y)

- “X→Y holds” is equivalent to “if we project the relation on XY, X is a superkey.”
Example: Functional Dependencies

<table>
<thead>
<tr>
<th>Student</th>
<th>Code</th>
<th>Name</th>
<th>Semester</th>
<th>Lecturer</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phil</td>
<td>2ID45</td>
<td>Advanced Databases</td>
<td>Spring 2011</td>
<td>Calders</td>
<td>A+</td>
</tr>
<tr>
<td>Mary</td>
<td>2ID45</td>
<td>Advanced Databases</td>
<td>Spring 2011</td>
<td>Calders</td>
<td>C</td>
</tr>
<tr>
<td>John</td>
<td>2ID45</td>
<td>Advanced Databases</td>
<td>Spring 2011</td>
<td>Calders</td>
<td>B-</td>
</tr>
<tr>
<td>Paul</td>
<td>2ID05</td>
<td>Databases I</td>
<td>Spring 2011</td>
<td>Fletcher</td>
<td>C</td>
</tr>
</tbody>
</table>

- Code $\rightarrow$ Name
- Code, Semester $\rightarrow$ Lecturer
- Student, Code, Semester $\rightarrow$ Grade
Foreign Key Dependencies

(Also called “inclusion dependencies”)

• Two relations R(S) and T(U);
  T has primary key PK.

• F ⊆ S is a foreign key into T if:
  for every legal instance r of R and t of T,
  \[ \Pi_F r \subseteq \Pi_{PK} t \]

  (Alternative: for every \( u \in r \), there exists a \( v \in t \) such that \( u(F) = v(PK) \))
### Foreign Keys: Example

#### Courses

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ID45</td>
<td>Advanced Databases</td>
</tr>
<tr>
<td>2ID05</td>
<td>Databases I</td>
</tr>
</tbody>
</table>

#### Offerings

<table>
<thead>
<tr>
<th>Code</th>
<th>Semester</th>
<th>Lecturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ID45</td>
<td>Spring 2011</td>
<td>Calders</td>
</tr>
<tr>
<td>2ID05</td>
<td>Spring 2011</td>
<td>Fletcher</td>
</tr>
</tbody>
</table>

#### Follows

<table>
<thead>
<tr>
<th>Student</th>
<th>Code</th>
<th>Semester</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phil</td>
<td>2ID45</td>
<td>Spring 2011</td>
<td>A+</td>
</tr>
<tr>
<td>Mary</td>
<td>2ID45</td>
<td>Spring 2011</td>
<td>C</td>
</tr>
<tr>
<td>John</td>
<td>2ID45</td>
<td>Spring 2011</td>
<td>B-</td>
</tr>
<tr>
<td>Paul</td>
<td>2ID05</td>
<td>Spring 2011</td>
<td>C</td>
</tr>
</tbody>
</table>

- $\text{Offerings}(\text{Code}) \subseteq \text{Courses}(\text{Code})$
- $\text{Follows}(\text{Code},\text{Sem}) \subseteq \text{Offerings}(\text{Code},\text{Sem})$
Translate to the relational model

- **doctor**
  - DID
  - name
  - specialization
  - (0,n)

- **patient**
  - SSN
  - fname
  - sname
  - address
  - (0,n)

- **prescription**
  - Nr
  - product
  - quantity
  - for
  - (0,1)

- **Primary doctor**
  - (0,1)

- **by**
  - (1,1)
Database Normalization

• Relation R is in Boyce-Codd Normal Form if: for every functional dependency $X \rightarrow Y$ that holds in R, either
  – $Y \subseteq X$, or
  – X is a superkey

• Idea: avoid redundancies, which may lead to inconsistencies
**Example: Non-BCNF**

<table>
<thead>
<tr>
<th>Student</th>
<th>Code</th>
<th>Name</th>
<th>Semester</th>
<th>Lecturer</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phil</td>
<td>2ID45</td>
<td>Advanced Databases</td>
<td>Spring 2011</td>
<td>Calders</td>
<td>A+</td>
</tr>
<tr>
<td>Mary</td>
<td>2ID45</td>
<td>Advanced Databases</td>
<td>Spring 2011</td>
<td>Calders</td>
<td>C</td>
</tr>
<tr>
<td>John</td>
<td>2ID45</td>
<td>Advanced Databases</td>
<td>Spring 2011</td>
<td>Calders</td>
<td>B-</td>
</tr>
<tr>
<td>Paul</td>
<td>2ID05</td>
<td>Databases I</td>
<td>Spring 2011</td>
<td>Fletcher</td>
<td>C</td>
</tr>
</tbody>
</table>

- Some violating FDs:
  - Code $\rightarrow$ Name
  - Code, Semester $\rightarrow$ Lecturer
### Example: BCNF

#### Courses

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ID45</td>
<td>Advanced Databases</td>
</tr>
<tr>
<td>2ID05</td>
<td>Databases I</td>
</tr>
</tbody>
</table>

#### Offerings

<table>
<thead>
<tr>
<th>Code</th>
<th>Semester</th>
<th>Lecturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ID45</td>
<td>Spring 2011</td>
<td>Calders</td>
</tr>
<tr>
<td>2ID05</td>
<td>Spring 2011</td>
<td>Fletcher</td>
</tr>
</tbody>
</table>

#### Follows

<table>
<thead>
<tr>
<th>Student</th>
<th>Code</th>
<th>Semester</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phil</td>
<td>2ID45</td>
<td>Spring 2011</td>
<td>A+</td>
</tr>
<tr>
<td>Mary</td>
<td>2ID45</td>
<td>Spring 2011</td>
<td>C</td>
</tr>
<tr>
<td>John</td>
<td>2ID45</td>
<td>Spring 2011</td>
<td>B-</td>
</tr>
<tr>
<td>Paul</td>
<td>2ID05</td>
<td>Spring 2011</td>
<td>C</td>
</tr>
</tbody>
</table>
OLTP Systems

• Relations in principle normalized
  – Avoid update inconsistencies

• Primary keys and foreign key dependencies are automatically checked

• Transaction management ensures ACID
Exercise: ER Modeling

Design an ER schema for keeping track of information about vote taken in the U.S. House of Representatives during the current two-year congressional session. The database needs to keep track of each U.S. STATEs Name (e.g., Texas, New York, Connecticut) and include the Region of the state (whose domain is Northeast, Midwest, Southeast, Southwest, West). Each congress person in the House of Representatives is described by his or her Name, plus the District represented, the start date when the congress person was first elected, and the political Party to which he or she belongs (whose domain is Republican, Democrat, Independent, Other)). The database keeps track of each BILL (i.e., proposed law), including the name of the bill, the date of the vote on the bill, whether the bill passed or failed (whose domain is Yes, No, and the Sponsor (the congress person(s) who sponsored-that is, proposed-the bill). The database keeps track of how each congress person voted on each bill (domain of Vote attribute is Yes, No, Abstain, Absent).