Databases

dr. Jörg Verstraete

Polish Academy of Sciences
Systems Research Institute

jorg.verstraete@ibspan.waw.pl
http://www.mini.pw.edu.pl/~verstraetej/
Today

• Lectures
  – Data Integrity
  – Entity Relationship modelling

• Labs
  – Exercises on ER modelling
Data integrity

• Several aspects
  – Entity integrity
  – Referential integrity
  – Additional constraints

• Modern DBMS, when configured properly, reject data modification statements that violate data integrity

• Terminology
  – entity = table (in relational model)
  – reference = foreign key
Entity integrity

• Objective
  – ensures that the entities in the model are well defined
    • records should be uniquely identifiable: primary key has to be unique

• Mechanism
  – primary key is defined for a table (+ unique index)
  – DBMS monitors CRUD (CRReate / Update / Delete) operations to maintain uniqueness
    • e.g. insert of a duplicate key is rejected
Example

- Entity integrity
  - Primary key
    - Students: { Student id }
  - Operations:
    - Student with id 2
      - exists
    - Student with id NULL
      - not allowed for primary key attributes: not unique
    - Student with id 100
- Note: DBMS will define a clustered index on the primary key

<table>
<thead>
<tr>
<th>Student id</th>
<th>Last Name</th>
<th>First Name</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Marsh</td>
<td>Stan</td>
<td>F1</td>
</tr>
<tr>
<td>2</td>
<td>Broflovski</td>
<td>Kyle</td>
<td>F1</td>
</tr>
<tr>
<td>3</td>
<td>Cartman</td>
<td>Erik</td>
<td>F2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Referential integrity

• Objective
  – guarantee that the references (foreign keys) between entities refer to existing records, identified by a matching primary key

• Mechanism
  – primary key is defined for a relation; foreign key is defined for a table that references it
  – DBMS monitors CRUD (CReate / Update / Delete) operations to maintain uniqueness
    • e.g. upon insert, existence of reference is checked
Example

- Referential integrity
  - Primary key
    - Students: { Student id }
    - Faculty: { Faculty id }
  - Foreign key
    - Students: { Faculty id }
    - references table Faculties
  - Operations:
    - insert student with faculty F3
      - no such faculty
    - delete faculty F1
      - reject or cascade!

<table>
<thead>
<tr>
<th>Student</th>
<th>Last Name</th>
<th>First Name</th>
<th>Faculty</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Marsh</td>
<td>Stan</td>
<td>F1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Broflovski</td>
<td>Kyle</td>
<td>F1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Cartman</td>
<td>Erik</td>
<td>F2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Faculties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faculty id</td>
</tr>
<tr>
<td>F1</td>
</tr>
<tr>
<td>F2</td>
</tr>
</tbody>
</table>
Example

- Referential integrity
  - Primary key
    - Rooms: \{ Building, Room \}
    - Assets: \{ AssetId \}
  - Foreign key
    - Assets: \{ Building, Room \}
    - references table Rooms
  - Operations:
    - Every asset must refer to an existing building AND room
Additional constraints

• It is possible to define logical constraints that must be fulfilled by all the records
  – mathematical: e.g. salary > tax
  – allowing / disallowing null values

• These conditions are checked whenever CRUD operations are attempted

• Conditions are the same for all clients and DBMS enforces them. Such constraints can also be placed in the client applications to reflect local conditions / configuration settings.

• Constraints can be lifted temporarily
Integrity / constraints

- All conditions and constraints must be valid for all possible content of the database
  - if an attribute at this time is not null for all values, it does not imply that null should not be allowed
  - model of reality determines which constraints are needed, not the current content of the database
ER Modelling

• What?
  – Relational database: many tables
    • how many? which tables?
  – Issues with representation of reality, normal forms, integrity

• ER modelling
  – Creation of a diagram that helps to translate a real world problem to a relational database
Terminology

• Entity
  – real world object: student, course, faculty, …
• Entity class (entity type / entity set)
  – set of entities with the same characteristics
    • e.g. students of a faculty
• Organisations make decisions based on entities;
  – they should be identified
  – their attributes should be described
  – relations have to be identified
    • e.g. one student has many courses
Entity relationship diagram

- Graphical diagram that depicts entities, their relations and attributes
- De facto standard in database programming:
  - detailed description of a database structure
  - there are tools that allow to generate table definitions (sql CREATE TABLE statements)
  - there are tools to generate an E-R diagram based on a known DB structure (e.g. MS SQL Server)
- Used at all stages of a database system life cycle, from project initiation, through planning, execution to maintenance
Entity relationship diagram

- Notations
  - several variants exist
    - defined by company policy
    - case tools used
  - example
- How to create?
  - Identify entities (objects, people and relations) that need to be described in the system
Design approaches

- **Top down**
  - identify organisation's entities
  - attach attributes and determine relationships

- **Bottom up**
  - start from requirements of different end users to determine possible entities
  - merge entities appropriately

- **Reality**
  - mixture of both
  - top down allows to capture overall requirements and select architecture
  - bottom-up helps to answer the needs of different users
Guidelines

• Top down
  • Organisation's management staff should provide business objectives
  • Often, economic justification for the system is also expressed through ROI (return on investment) and NPV (net profit value).
  – Start from identifying supported departments and entities: they need to improve decision making and business processes

• Bottom-up
  – Interview with key stakeholders should proved their perspective on the system and its functionality.
  – Investigate processes to be supported by the system, not to be left with the users' imagination of the system.
  – Combining the opinions of different users into a consistent design requires experience

• Reality
  – Management defines the scope: you start top-down
  – Then, you verify the initial view of the system using both bottom-up and top-down
Guidelines

• Reality
  - The role of the system to be created is often not clear
  - The system is often expected to solve organizational deficiencies: software analysts have to both design the system and participate in redefining existing business processes and procedures

• Thus
  - Try to determine real reasons for creating the database system and identify the problems to be solved
Entity relationship diagram

- **Multiplicities**
  - **Crow's Foot notation**
  - property of a relation
    - **minimal**
      - $0 \circ$
    - $1 \mid$
    - **maximal**
      - $1 \mid$
  - more than $1 \prec$
  - both directions

- Faculty has 0 or more courses
- Course is given in **at least one** and **at most one** faculty
- Student has 1 or more courses
- Course has 0 or more students
Multiplicities: examples

- Consider 1 faculty
  - Minimal
    - Q: can it have 0 courses?
      - yes
      - no
  - Maximal
    - Q: can it have more than 1 course?
      - no
      - yes

- Consider 1 course: same situation
Multiplicities: examples

• Consider entities A, B
  • Possible multiplicities
    – With 1 A, how many B?
      • 0 or 1
      • 1, not more
      • 0 or many
      • 1 or many
  • One-to-one / one-to-many / many-to-one / many-to-many
  • Multiplicities stem from the real world
Entities

- Entity represents real world object
  - has attributes, different notations exist
- Entities will translate to tables
  - attributes are columns
  - relations are foreign keys
- multiplicities help with
  - direction of foreign keys
  - allowed values
Entities: subtypes

- Entities can have sub-types
  - Subtypes can
    - share common attributes
    - have own attributes
    - have relations with other entities
- Different options for translating entities to tables

<table>
<thead>
<tr>
<th>Vehicle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
</tr>
<tr>
<td>Van</td>
</tr>
<tr>
<td>Lorry</td>
</tr>
</tbody>
</table>
Overview of relations

- **One-to-many**
  - foreign key

<table>
<thead>
<tr>
<th>Students</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Student</strong></td>
<td><strong>Faculty</strong></td>
</tr>
<tr>
<td>Student ID</td>
<td>Student ID</td>
</tr>
<tr>
<td>Last name</td>
<td>Faculty ID</td>
</tr>
<tr>
<td>First name</td>
<td>Name</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>Marsh</td>
</tr>
<tr>
<td>Stan</td>
<td>Mathematics</td>
</tr>
<tr>
<td>2</td>
<td>Broflovski</td>
</tr>
<tr>
<td>Kyle</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>Cartman</td>
</tr>
<tr>
<td>Erik</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Faculties

<table>
<thead>
<tr>
<th>Faculties</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Faculty</strong></td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>Mathematics</td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>Chemistry</td>
<td></td>
</tr>
</tbody>
</table>

**Would** allow students without faculty, nulls would be allowed in **Faculty**

**Would** require faculties to have at least one student

Not very realistic and not easy to enforce
Overview of relations

- Many-to-many

**Orders**

<table>
<thead>
<tr>
<th>Order id</th>
<th>Date</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01/02/16</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>01/02/16</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>02/02/16</td>
<td>...</td>
</tr>
</tbody>
</table>

**Order Details**

<table>
<thead>
<tr>
<th>Order id</th>
<th>Product Id</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>P2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>P1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>P2</td>
<td></td>
</tr>
</tbody>
</table>

**Products**

<table>
<thead>
<tr>
<th>Product Id</th>
<th>Name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

*can contain attributes, e.g. amount*
Overview of relations

- Recurrent relations
  - entity references itself
  - e.g. hierarchy

<table>
<thead>
<tr>
<th>Employee</th>
<th>Last Name</th>
<th>First Name</th>
<th>Managed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Marsh</td>
<td>Stan</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Broflovski</td>
<td>Kyle</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Cartman</td>
<td>Erik</td>
<td>NULL</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Employee has a manager, who also is an employee that can have a manager
Overview of relations

- Mutually exclusive relations
  - Entity has relation with just one of multiple
  - e.g. for each finished order there is either a receipt or an invoice

Line with dots placed on them denotes an XOR relation: either there is an invoice, or there is a receipt
Additional properties

• Some tools allow additional properties of the relation
  – cardinality greater than one, but limited
  – settings for cascaded / restricted delete
  – definition of candidate keys
Entity relationship modelling

• Vital step towards fully functional system
  – ER diagrams help (and force team members) to capture all relevant data needs and detailed relations in the data
  – The diagrams help to involve stakeholders in the analysis process and they ensure that entities and relations are properly captured and documented
  – ER models are created before tables come into being. This promotes revisions based on a better understanding of the system through the diagrams.
Additional properties

• Example
  - Nouns
    • entities
    • attributes
  - Key words
    • has
    • can have
    • is

Consider the database for a car rental company.
A person can make a reservation for a car. The reservation
is numbered per person and per day and holds information
about the type of car, pick-up data and location and drop-off date and location.
The person can provide a list of people that will drive the
car. Each car has a license plate, manufacturer and type
(compact car, ...) specified. The car also has a damage
report that describes damages on the car. A history of
these reports is kept.