The Relational Model

Database Systems Lecture 3
Natasha Alechina

In This Lecture

- Relational data integrity
- For more information
  - Connolly and Begg chapter 3
  - E.F. Codd’s paper
    ‘A Relational Model of Data for Large Shared Data Banks’ – a link from the module web page, ~nza/G51DBS.

The Relational Model

- Introduced in E.F. Codd’s 1970 paper
  “A Relational Model of Data for Large Shared Databanks”
- The foundation for most (but not all) current database systems
- Concerned with 3 main things
  - Data structure (how data is represented)
  - Data integrity (what data is allowed)
  - Data manipulation (what you can do with the data)

The Relational Data Structure

- Data is stored in relations (tables)
- Each relation has a scheme (heading)
- The scheme defines the relation’s attributes (columns)
- Data takes the form of tuples (rows)

New thing: scheme (and attributes)

Before... After

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>Just numbers of columns</th>
<th>Attributes</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>23</td>
<td>Mary</td>
<td>20</td>
<td>Mark</td>
<td>18</td>
</tr>
<tr>
<td>Jane</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>Just numbers of columns</th>
<th>Attributes</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>23</td>
<td>Mary</td>
<td>20</td>
<td>Mark</td>
<td>18</td>
</tr>
<tr>
<td>Jane</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unnamed and named tuples

A tuple is <John, 23>

A tuple is {((Name,John), (Age,23))}
Not a big difference!

- There is no fundamental difference between named and unnamed perspectives on relations.
- We could have written tuples \( <a,b,c> \) as sets of pairs \( \{ (1,a), (2,b), (3,c) \} \), only we know anyway in which order 1,2,3 go, so we can skip the numbers.
- Written as sets of pairs (partial functions), tuples can be written in any order, e.g. \( \{ (3,c), (2,b), (1,a) \} \).

More formally -

- A scheme is a set of attributes.
- A tuple assigns a value to each attribute in its scheme.
- A relation is a set of tuples with the same scheme.

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>23</td>
</tr>
<tr>
<td>Mary</td>
<td>20</td>
</tr>
<tr>
<td>Mark</td>
<td>18</td>
</tr>
<tr>
<td>Jane</td>
<td>21</td>
</tr>
</tbody>
</table>

(\( \{ \text{Name}, \text{John} \}, \{\text{Age}, 23\} \), \( \{ \text{Name}, \text{Mary} \}, \{\text{Age}, 20\} \), \( \{ \text{Name}, \text{Mark} \}, \{\text{Age}, 18\} \), \( \{ \text{Name}, \text{Jane} \}, \{\text{Age}, 21\} \))

The Relational Model

Relations

- Scheme is (ID, Name, Salary, Department)
- Attributes are ID, Name, Salary, and Department
- Degree is 4

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Salary</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>M139</td>
<td>John Smith</td>
<td>18,000</td>
<td>Marketing</td>
</tr>
<tr>
<td>M140</td>
<td>Mary Jones</td>
<td>22,000</td>
<td>Marketing</td>
</tr>
<tr>
<td>A368</td>
<td>Jane Brown</td>
<td>24,000</td>
<td>Personnel</td>
</tr>
<tr>
<td>P222</td>
<td>Mark Brown</td>
<td>20,000</td>
<td>Accounts</td>
</tr>
<tr>
<td>A367</td>
<td>David Jones</td>
<td>20,000</td>
<td>Accounts</td>
</tr>
</tbody>
</table>

Tuples, e.g.

- \( \{ \text{ID}, \text{A368} \} \)
- \( \{ \text{Name}, \text{Jane Brown} \} \)
- \( \{ \text{Salary}, 22,000 \} \)
- \( \{ \text{Department}, \text{Accounts} \} \)

Cardinality is 5

The Relational Model

Relational Data Integrity

- Data integrity controls what data can be in a relation.
- Domains restrict the possible values a tuple can assign to each attribute.
- Candidate and Primary Keys identify tuples within a relation.
- Foreign Keys link relations to each other.

Attributes and Domains

- A domain is given for each attribute.
- The domain lists the possible values for that attribute.
- Each tuple assigns a value to each attribute from its domain.

Examples:

- An ‘age’ might have to come from the set of integers between 0 and 150.
- A ‘department’ might come from a given list of strings.
- A ‘notes’ field might allow any string at all.

Candidate Keys

- A set of attributes in a relation is called a candidate key if, and only if,
  - Every tuple has a unique value for the set of attributes (uniqueness).
  - No proper subset of the set has the uniqueness property (minimality).

ID       | First   | Last  |
---------|---------|-------|
S139     | John    | Smith |
S140     | Mary    | Jones |
S141     | John    | Brown |
S142     | Jane    | Smith |

Candidate key: (ID, (First,Last)) looks plausible but we may get people with the same name (ID, First), ( mass, Last) and (ID, First, Last) satisfy uniqueness, but are not minimal (First and Last) do not give a unique identifier for each row.
Choosing Candidate Keys

- Important: don’t look just on the data in the table to determine what is a candidate key.
- The table may contain just one tuple, so anything would do!
- Use knowledge of the real world – what is going to stay unique!

Primary Keys

- One Candidate Key is usually chosen to be used to identify tuples in a relation.
- This is called the Primary Key.
- Often a special ID attribute is used as the Primary Key.

NULLs and Primary Keys

- Missing information can be represented using NULLs.
- A NULL indicates a missing or unknown value.
- More on this later...

Entity Integrity: Primary Keys cannot contain NULL values.

Foreign Keys

- Foreign Keys are used to link data in two relations. A set of attributes in the first (referencing) relation is a Foreign Key if its value always either
  - Matches a Candidate Key value in the second (referenced) relation, or
  - Is wholly NULL
- This is called Referential Integrity.

Foreign Keys - Example

- Employee
  - ID: E1496, E1497, E1498, E1499
  - Name: John Smith, Mary Brown, Mark Jones, Jane Smith
  - Manager: ID

- Department
  - DID: 13, 14, 15
  - DName: Marketing, Accounts, Personnel

Foreign Keys - Example

- Employee
  - ID: E1496, E1497, E1498, E1499
  - Name: John Smith, Mary Brown, Mark Jones, Jane Smith
  - Manager: ID

- Department
  - DID: 13, 14, 15
  - DName: Marketing, Accounts, Personnel

(ID) is a Candidate Key for Employee, and (Manager) is a Foreign Key, which refers to the same relation - every tuple’s Manager value is either NULL or matches an ID value.
Referential Integrity

- When relations are updated, referential integrity can be violated
- This usually occurs when a referenced tuple is updated or deleted
- There are a number of options:
  - **RESTRICT** - stop the user from doing it
  - **CASCADE** - let the changes flow on
  - **NULLIFY** - make values NULL

Referential Integrity - Example

- What happens if Marketing’s DID is changed to 16 in Department?
- The entry for Accounts is deleted from Department?

<table>
<thead>
<tr>
<th>Department</th>
<th>DName</th>
<th>DID</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Marketing</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>Accounts</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>Personnel</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employee</th>
<th>EName</th>
<th>EID</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>John Smith</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>Mary Brown</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>Mark Jones</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>Jane Smith</td>
<td>18</td>
</tr>
</tbody>
</table>

RESTRICT

- RESTRICT stops any action that violates integrity
  - You cannot update or delete Marketing or Accounts
  - You can change Personnel as it is not referenced

<table>
<thead>
<tr>
<th>Department</th>
<th>DName</th>
<th>DID</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Marketing</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>Accounts</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>Personnel</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employee</th>
<th>EName</th>
<th>EID</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>John Smith</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>Mary Brown</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>Mark Jones</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>Jane Smith</td>
<td>18</td>
</tr>
</tbody>
</table>

CASCADE

- CASCADE allows the changes made to flow through
  - If Marketing’s DID is changed to 16 in Department, then the DIDs for John Smith and Mark Jones also change
  - If Accounts is deleted then so is Mary Brown

<table>
<thead>
<tr>
<th>Department</th>
<th>DName</th>
<th>DID</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Marketing</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>Accounts</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>Personnel</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employee</th>
<th>EName</th>
<th>EID</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>John Smith</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>Mary Brown</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>Mark Jones</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>Jane Smith</td>
<td>18</td>
</tr>
</tbody>
</table>

NULLIFY

- NULLIFY sets problem values to NULL
  - If Marketing’s DID changes then John Smith’s and Mark Jones’ DIDs are set to NULL
  - If Accounts is deleted, Mary Brown’s DID becomes NULL

<table>
<thead>
<tr>
<th>Department</th>
<th>DName</th>
<th>DID</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Marketing</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>Personnel</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employee</th>
<th>EName</th>
<th>EID</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>John Smith</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>Mary Brown</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>Mark Jones</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>Jane Smith</td>
<td>18</td>
</tr>
</tbody>
</table>

Naming Conventions

- Naming conventions
  - A consistent naming convention can help to remind you of the structure
  - Assign each table a unique prefix, so a student name may be stuName, and a module name modName

- Naming keys
  - Having a unique number as the primary key can be useful
  - If the table prefix is abc, call this abcID
  - A foreign key to this table is then also called abcID
The Relational Model

Example

- Student
  - stuID
  - stuName

- Module
  - modID
  - modName

- Enrolment
  - stuID
  - modID

These entries are clearly related to the Student table.

These entries are clearly related to the Module table.

These entries are clearly related to tables other than Enrolment.

Next Lecture

- Entity/Relationship models
- Entities and Attributes
- Relationships and Cardinality Ratios
- E/R Diagrams
- For more information
  - Connolly and Begg chapter 11.
  - Ullman and Widom chapter 2.