Relational Database Model

Relational Model

- One single data modeling “tool”: relation, or a 2D table;
- A relational database is a collection of relations;
- High degree of data independence
- Association between information elements (constraints)

More Formally

Relation: a two-dimensional table of columns and rows.
Attribute, Field: name of a column in the relation.
- take values from predefined domains

Schema: the name of a relation plus the set of attributes of the relation (and their domains).
- E.g. Book(ISBN string, Title string, Author string, year integer).

Relation instance: a set of tuples for a given relation.
- changes with time (as stuff gets added, deleted, modified)
- schema usually does not change (although it might in some cases)

Cardinality: number of tuples in a relation
Degree: number of attributes in a relation

Constraints

Superkey  a collection of attributes in a relation that uniquely identifies each tuple in it.

Candidate key  a superkey that has no superkey subsets

Primary key  one candidate key per relation, designated to be the main way of maintaining tuple uniqueness.

Key constraint : each relation must have a primary key.

Foreign key  a primary key of one relation, included in the attributes of another relation (usually for the purpose of linking two components of the database together).

Referential integrity constraint  each collection of values of a foreign key in a relation must appear as a primary key in the referenced relation.

Null value  a “no value” value for a relational attribute. Lack of value, or value not yet available.

not null constraint  : a statement that a specific attribute is not allowed to have null values. (e.g., primary key attributes).
Examples

Example 1. A university registrar wants to organize a list of courses offered at a university. The information available about each course is its prefix (e.g., 'CSC'), its number (e.g., 365), its title (e.g., 'Introduction to Databases'), its catalog description, and the department responsible for scheduling the course. The registrar creates the following relational table to represent the courses:

Courses(Prefix String, CourseNumber Integer, Title String, Catalog String, Department String)

A portion of this table would look as follows:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>CourseNumber</th>
<th>Title</th>
<th>Catalog Description</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC</td>
<td>365</td>
<td>Introduction to Databases</td>
<td>Basic principles of database management...</td>
<td>CSSE</td>
</tr>
<tr>
<td>CSC</td>
<td>357</td>
<td>Systems Programming</td>
<td>C programming language from a system...</td>
<td>CSSE</td>
</tr>
<tr>
<td>CHEM</td>
<td>212</td>
<td>Introduction to Organic Chemistry</td>
<td>Structure, isomerism, nomenclature, fundamental...</td>
<td>Chemistry</td>
</tr>
<tr>
<td>STAT</td>
<td>331</td>
<td>Statistical Computing with R</td>
<td>Importing, managing, and cleaning data from ...</td>
<td>Statistics</td>
</tr>
<tr>
<td>STAT</td>
<td>365</td>
<td>Statistical Communication</td>
<td>Written communication of statistical ideas...</td>
<td>Statistics</td>
</tr>
</tbody>
</table>

Determining the keys:

- The pair of attributes (Prefix, CourseNumber) together is a candidate key for the Courses table. The requirement that the Prefix-CourseNumber pair is unique for each course comes from outside - this is university catalog rule established before the Courses table is introduced.
- The Prefix attribute alone is not enough to uniquely identify every row in the table (e.g., there are two that have the CSC prefix).
- The CourseNumber attribute alone is also not enough: there are two rows describing different courses whose number is 365.
- The second candidate key in the table could be Title. This means that the Registrar’s office requires all courses to have a unique title.
- Between the Title and the (Prefix, CourseNumber), the registrar selects (Prefix, CourseNumber) as the primary key.

Example 2. After examining the Courses table, the registrar arrives to the conclusion that they want another attribute in the table that merges (concatenates) the course prefix and the course number. As a result, the new Courses table looks as follows:

Courses(CourseId String, Prefix String, CourseNo Integer, Title String, Catalog String, Department String)

A portion of this table looks as follows:

<table>
<thead>
<tr>
<th>CourseId</th>
<th>Prefix</th>
<th>CourseNo</th>
<th>Title</th>
<th>Catalog Description</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC 365</td>
<td>CSC</td>
<td>365</td>
<td>Introduction to Databases</td>
<td>Basic principles of database management...</td>
<td>CSSE</td>
</tr>
<tr>
<td>CSC 357</td>
<td>CSC</td>
<td>357</td>
<td>Systems Programming</td>
<td>C programming language from a system...</td>
<td>CSSE</td>
</tr>
<tr>
<td>CHEM 212</td>
<td>CHEM</td>
<td>212</td>
<td>Introduction to Organic Chemistry</td>
<td>Structure, isomerism, nomenclature, fundamental...</td>
<td>Chemistry</td>
</tr>
<tr>
<td>STAT 331</td>
<td>STAT</td>
<td>331</td>
<td>Statistical Computing with R</td>
<td>Importing, managing, and cleaning data ...</td>
<td>Statistics</td>
</tr>
<tr>
<td>STAT 365</td>
<td>STAT</td>
<td>365</td>
<td>Statistical Communication</td>
<td>Written communication of statistical ideas...</td>
<td>Statistics</td>
</tr>
</tbody>
</table>

We now have the following candidate keys in this table:

- (Prefix, CourseNo) - our old primary key

1 This is because there is a lot of code being written that accesses the table and has to concatenate these two attributes every single time.
• **Title** - per our assumption that the Registrar’s office will not let two courses with exactly the same name to co-exist.

• **CourseId** - as this is literally concatenation of the two values that forms a candidate key.

Given these three candidate keys, the registrar now selects **CourseId** as the **primary key** for this table.

**Example 3.** In addition to the courses, the registrar wants to organize a list of the departments. The registrar needs the following information about each department: its full name, the college it is in, it’s location on campus (building number and room number). The first version of the **Departments** table looks as follows:

\[
\text{Departments} (\text{Name String, College String, Building Integer, Room String})
\]

(note: rooms may have values like ’245a’ so, they are strings.)

A portion of the table looks as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>College</th>
<th>Building</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science and Software Engineering</td>
<td>CENG</td>
<td>14</td>
<td>245</td>
</tr>
<tr>
<td>Statistics</td>
<td>COSAM</td>
<td>25</td>
<td>107D</td>
</tr>
<tr>
<td>Chemistry and Biochemistry</td>
<td>COSAM</td>
<td>180</td>
<td>206</td>
</tr>
<tr>
<td>Physics</td>
<td>COSAM</td>
<td>180</td>
<td>207</td>
</tr>
</tbody>
</table>

Keys:

• The **Name** attribute is a candidate key (no two departments can have the same name).

• Assuming no two departments share the same office - a reasonable assumption at Cal Poly, the pair (**Building**, **Room**) is also a candidate key.

• These are the only candidate keys in the table (**College** by itself is NOT a candidate key, and any other combination of attributes either contains a key, or is not unique - e.g., (**College**, **Building**) is not a candidate key, because multiple departments can have offices in the same building, and (**College**, **Room**) is not unique because multiple departments can have offices with the same room number but in different buildings).

• Between the two candidate keys, the registrar selects **Name** as the **primary key**.

**Example 4.** The registrar notes that the **Courses** table has a **Department** column. Now that there is a **Departments** table in the database, the registrar wants to make sure that it is possible to look up the department that teaches a specific course.

However, *at present*, this is not always possible.

• For example, for **STAT 331**, the **Department** is listed as **Statistics**, which matches the primary key of one record in the **Departments** table.

• But for **CSC 365**, the **Department** is listed as **CSSE**, and the **Departments** table does not have a row where the value of the **Name** attribute is **CSSE**.

The registrar has several options on how to achieve their goal of being able to match the department to each course. All of these options involve the concept of a **foreign key constraint**.

• Require that the values of the **Department** field of the **Courses** table are the same strings as the values of the **Name** field of the **Departments** table. In this case, **Courses.Dependent** is a **foreign key** referencing **Departments.Name**.

This would require changing the values in the **Courses.Dependent** column for some rows, but will not require any changes to the **Departments** table.
• Create a new column named **Abbreviation** in the **Departments** table, and make its values the abbreviated names of the departments that are used in the **Courses**.Department column. **Departments.Abbreviation** is a candidate key, and it can also be used as a *primary key* for the **Departments** table. **Courses**.Department is a foreign key referencing **Departments**.Abbreviation.

This would require (a) ensuring consistency in the use of department abbreviations in **Courses**.Department (i.e., always use CSSE, always use Chemistry, and so on), and a change to the schema of the **Departments** table.

• Create a new column named **DeptId** in the **Departments** table, make its values integers, make this column the *primary key* of the **Departments** table, and assign unique Ids to each row in the table. Then, change the **Courses**.Department column so that it now has integer values, and assign to each tuple the department Id of the department teaching the course. **Courses**.Department becomes a foreign key referencing Department.Abbreviation.

This would require alterations to both the schema of the **Departments** table and the **Courses** table.

Each approach has its own benefits and drawbacks.

• **Advantage:** the first approach requires no changes in the schema of either table, and simply aligns the values in the **Courses** table with those in the **Departments** table. **Disadvantage:** however, this approach will store potentially long strings of text (full department names) in the **Courses** tables.

• **Advantage:** the second approach does not change the contents of the **Courses** table (short of ensuring consistency of naming conventions). **Disadvantage:** this approach does not really have too many disadvantages, but it does introduce a new column to the **Departments** table with somewhat redundant content (abbreviated name of the department is often the same as the name of the department).

• **Advantage:** the third approach is the most robust - the values used to identify the departments are unique integers, that are easy to store, compare, and use. **Disadvantage:** however, this approach (a) reduces the readability of the data in the **Courses** tuples (what does the Department number 4 refer to?), and (b) requires the largest amount of change to the table schemas.

Of the three approaches, the registrar selects the third, because despite its disadvantages, use of numbers as unique ids is more robust and easy to maintain.

As a result, the registrar winds up with the following tables:

**Departments**(DeptId Integer, Name String, College String, Building Integer, Room String)

**Courses**(Prefix String, CourseNumber Integer, Title String, Catalog String, Department Integer)

and the following constraints:

• **Departments:**
  - Primary Key: DeptId
  - Additional Candidate Keys: (Name); (Building, Room)

• **Courses:**
  - Primary Key: CourseId
  - Additional Candidate Keys: (Prefix, CourseNo); (Title); (Description)
- **Foreign Key**: *Department* is a foreign key referencing *Departments.DeptId*

The revised fragments of both tables now look as follows.

**Departments:**

<table>
<thead>
<tr>
<th>DeptId</th>
<th>Name</th>
<th>College</th>
<th>Building</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Computer Science and Software Engineering</td>
<td>CENG</td>
<td>11</td>
<td>215</td>
</tr>
<tr>
<td>2</td>
<td>Statistics</td>
<td>COSAM</td>
<td>25</td>
<td>107D</td>
</tr>
<tr>
<td>3</td>
<td>Chemistry and Biochemistry</td>
<td>COSAM</td>
<td>180</td>
<td>206</td>
</tr>
<tr>
<td>4</td>
<td>Physics</td>
<td>COSAM</td>
<td>180</td>
<td>207</td>
</tr>
</tbody>
</table>

**Courses:**

<table>
<thead>
<tr>
<th>CourseId</th>
<th>Prefix</th>
<th>CourseNo</th>
<th>Title</th>
<th>Catalog</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC 365</td>
<td>CSC</td>
<td>365</td>
<td>Introduction to Databases</td>
<td>Basic principles of database management...</td>
<td>1</td>
</tr>
<tr>
<td>CSC 357</td>
<td>CSC</td>
<td>357</td>
<td>Systems Programming</td>
<td>C programming language from a system...</td>
<td>1</td>
</tr>
<tr>
<td>CHEM 212</td>
<td>CHEM</td>
<td>212</td>
<td>Introduction to Organic Chemistry</td>
<td>Structure, isomerism, nomenclature...</td>
<td>3</td>
</tr>
<tr>
<td>STAT 331</td>
<td>STAT</td>
<td>331</td>
<td>Statistical Computing with R</td>
<td>Importing, managing, and cleaning data ...</td>
<td>2</td>
</tr>
<tr>
<td>STAT 365</td>
<td>STAT</td>
<td>365</td>
<td>Statistical Communication</td>
<td>Written communication of statistical ideas...</td>
<td>2</td>
</tr>
</tbody>
</table>