Lecture 6: A Primer on Relational Database Management Systems

Introduction

The last two lectures provided the conceptual foundation necessary for the construction and management of relational databases. Today’s lecture begins the transition from conceptual to logical issues of database design and management. We will examine definitions for relation schemes, database schemes, and relationships, along with examples of how these schemes become tables, databases, relationships, and relational databases. Examples are provided by a simple database for restaurants.

Relation Schemes and Relations

A relation scheme can be defined as a set of attribute names, each of which is tied to a domain. Relation schemes are written using the standard format:

RELATION NAME (Attribute_1, Attribute_2, … Attribute_n)

This format begins with the name of the relation scheme, followed by a set of attribute names. One or more of these attribute names are underlined which signifies that these are key attributes. Key attributes allow for unique identification of individual tuples. For example a relation scheme for restaurants could look like this:

RESTAURANT (Rest_ID, Name, X-coord, Y-coord, Address, Telephone)

Relation is a formal name for what we commonly call a table.

Using this scheme, it is a simple matter to create a relation from the relation scheme.

Relations are defined as a finite set of tuples associated with a relation scheme, such that

- Each tuple is an ordered list containing as many data items as there are attribute names in the relation scheme.
- Each data item is drawn from the domain with which its attribute name is associated.

Following this definition, we can use the RESTAURANT relation scheme to create the RESTAURANT relation (below). In this relation, note that the attributes in the relation match those of the relation scheme. Also note that because the values used in the relation are drawn from domains associated with the attributes, the tuples form an ordered list. In other words, you won’t find a telephone number in the Name attribute because telephone
numbers are not in the domain for Name. Finally, the key field in this relation is Rest_ID. Because the values in a key field provide a means to uniquely identify each tuple, none of the values in this attribute will repeat.

### RESTAURANT (Rest_ID, Name, X-coord, Y-coord, Address, Telephone)

<table>
<thead>
<tr>
<th>Rest_ID</th>
<th>Name</th>
<th>X-coord</th>
<th>Y-coord</th>
<th>Address</th>
<th>Phone_no</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bob’s</td>
<td>507100</td>
<td>356700</td>
<td>1800 W. Peace St.</td>
<td>445-5055</td>
</tr>
<tr>
<td>2</td>
<td>Arthur’s</td>
<td>505100</td>
<td>357520</td>
<td>6440 N Warhol Rd.</td>
<td>445-1771</td>
</tr>
<tr>
<td>3</td>
<td>Lucy’s</td>
<td>515000</td>
<td>356700</td>
<td>6530 E 2nd St.</td>
<td>445-4250</td>
</tr>
<tr>
<td>4</td>
<td>Bob’s</td>
<td>507500</td>
<td>357550</td>
<td>3301 E. Lincoln</td>
<td>445-1436</td>
</tr>
</tbody>
</table>

### Database Schemes

A database scheme can be defined as a set of relation schemes. For example, the following relation schemes combine to form the Restaurant Database scheme.

**RESTAURANT** (Rest_ID, Name, X-coord, Y-coord, Address, Telephone)

**ROOM** (Rest_ID, Room_no, Capacity, Use, Location)

**CUISINE** (Rest_ID, Room_no, Cuisine)

**CHEF** (Rest_ID, Name, Cntry, School)

**AMBIANCE** (Rest_ID, Avg_cost, Dress, Parking)

In this database, each of these relation schemes would become a table with the listed name, attributes, and key field/s. Particularly important in database schemes are issues of key fields and domains.

<table>
<thead>
<tr>
<th>Rest_id</th>
<th>Room_#</th>
<th>Cap</th>
<th>Use</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>45</td>
<td>Dining</td>
<td>Indoors</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>200</td>
<td>Dining</td>
<td>Indoors</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>50</td>
<td>Bar</td>
<td>Indoors</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>35</td>
<td>Dining</td>
<td>Patio</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>15</td>
<td>Dining</td>
<td>Indoors</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>15</td>
<td>Dining</td>
<td>Patio</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>35</td>
<td>Dining</td>
<td>Indoors</td>
</tr>
</tbody>
</table>

It is important to note that many of these relations have compound keys. For example, the ROOM relation requires a combination of two attributes to uniquely identify each tuple. (Can you think of another way to uniquely identify each record without resorting to compound keys?)
It is also important to maintain logical consistency within values in a domain. For example, values in the Cntr attribute in the CHEF relation are not logically consistent. Sometimes USA is used and sometimes America is used to refer to the same country. Although we understand that these values are referencing the same entity, a computer is not as smart as we are and will not recognize them as being equivalent. The result of this lack of consistency is that retrieval of information becomes problematic.

Finally, domains work best when they are defined at the atomic level. That is, for each cell in the table, there is a single value, not a group of values. For example, the CUISINE relation indicates a restaurant with multiple cuisines by using a group of values (Steak, Seafood, and Vietnamese). This non-atomic approach would make information retrieval more difficult.

A better way to model these same data would be to define the domain at the atomic level, using 3 tuples to describe the cuisine at this restaurant. (Can you think of another way to model these data that would not involve so many duplicate cuisine entries?)

### Relationships and Relational Databases

A set of disconnected relations is not particularly useful, but if the tables can be related to each other they become a powerful vehicle for data management and analysis. A relational database can be defined as a set of relations connected by relationships. These relationships

- connect relations.
- Describe the connection between tables semantically (e.g. owns, works on, offers, etc.)
• may or may not be relations themselves

Associating the RESTAURANT and ROOMS relations can be done with the semantic relationship *has*; i.e., Restaurant *has* Rooms. This relationship between relations can also be diagrammed using conventional shapes and lines. The relations are represented as rectangles, the relationship as a diamond, and the connection between the relations and the relationship by lines.

![Diagram of Restaurant and Rooms relationship](Image)

Commonly called ER (for Entity-Relationship) diagrams, these graphic representations are important for designing relational databases. Using these graphic conventions, entire databases can be diagrammed to examine the different tables, attributes, and relationships before the creation and population of the tables. The simple example below shows five tables and the relationships between them for the Restaurant database. ER diagrams can be much more complicated that this, indicating with graphic conventions the types of relationships between tables, attributes, attribute definitions etc. If you want to see more examples, you can use an internet search engine to look for *ER diagram*.

![Diagram of Restaurant, Cuisine, Chef, Ambiance, and Rooms relationships](Image)

This description of relational database design has been intentionally simplified. If you are interested in learning more about relational databases and how they are designed and implemented, there are a number of books and articles that you can examine. The standard textbook is C. J. Date’s book, *An Introduction to Database Systems*. An academic and largely theoretical approach to database design, this book is the standard
work on relational databases but is less practical than many readers like. Other works
that are domain specific or written for particular software applications are also available,
including some specifically for GIS applications. For example, Arctur and Zeiler’s,
*Designing Geodatabases: case studies in GIS data modeling.*

**References Cited**

Arctur, D. and M. Zeiler

2004  *Designing geodatabases: case studies in GIS data modeling.* ESRI Press,
Redlands, Calif.

Date, C. J.

1995  *An introduction to database systems.* 6th ed. Addison-Wesley systems