THE MANY-TO-MANY RELATIONSHIP

Reading: Chapter 5

Objectives

- Learn to model a many-to-many relationship between two entities
- Create a database with a many-to-many relationship
- Write queries for a database with a many-to-many relationship
A Sales Form

Form represents a single sale to a customer, and includes one or more line items.

The Expeditioneer

Sale of Goods

<table>
<thead>
<tr>
<th>Sale# 123456</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item#</td>
<td>Description</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Grand Total

A sale might include many items, but a given item might be included in many sales.

Example

- A sale includes multiple items
- Items with the same item number might be sold to many customers
- Many-to-many relationship between Items and Sales

<table>
<thead>
<tr>
<th>ItemNo</th>
<th>ItemName</th>
<th>ItemType</th>
<th>ItemColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pocket knife - Nile</td>
<td>E</td>
<td>Brown</td>
</tr>
<tr>
<td>2</td>
<td>Pocket knife - Avon</td>
<td>E</td>
<td>Brown</td>
</tr>
<tr>
<td>3</td>
<td>Compass</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Compassing setup</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>
Items

<table>
<thead>
<tr>
<th>ID</th>
<th>ItemNo</th>
<th>ItemName</th>
<th>ItemType</th>
<th>ItemColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Pocket knife</td>
<td>E</td>
<td>Brown</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Compass</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Geopositioning system</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Map measure</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>Hat - Polar Explorer</td>
<td>C</td>
<td>Red</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Hat - Polar Explorer</td>
<td>C</td>
<td>White</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Boots - snake proof</td>
<td>C</td>
<td>Green</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>Boots - snake proof</td>
<td>C</td>
<td>Black</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>Safari Chair</td>
<td>F</td>
<td>Khaki</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>Hammock</td>
<td>F</td>
<td>Khaki</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>Tent - 8 person</td>
<td>F</td>
<td>Khaki</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>Tent - 2 person</td>
<td>F</td>
<td>Khaki</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>Safari cooking kit</td>
<td>F</td>
<td>Khaki</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>Pith helmet</td>
<td>C</td>
<td>Khaki</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>Pith helmet</td>
<td>C</td>
<td>White</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td>Map case</td>
<td>N</td>
<td>Brown</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>Sextant</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>Station</td>
<td>C</td>
<td>Black</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>Station</td>
<td>C</td>
<td>Brown</td>
</tr>
</tbody>
</table>

Sales

- LineItems table refers to both the Sales table and the Items table

<table>
<thead>
<tr>
<th>ID</th>
<th>saleNo</th>
<th>saleDate</th>
<th>saleText</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1/15/2003</td>
<td>Scuffy Australian - called himself Bruce</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1/15/2003</td>
<td>Man, Rather fond of hats.</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1/15/2003</td>
<td>Woman, Planning to row Atlantic - lengthwise</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1/15/2003</td>
<td>Man. Trip to New York - Thinks NY is a jungle!</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1/16/2003</td>
<td>Expedition leader for African safari</td>
</tr>
</tbody>
</table>
Many-to-Many Relationships

- Many-to-many relationships cannot be represented directly in a relational DB
- You refine the relationship to include an associative (or “in-between”) entity with only one-to-many relationships

Workbench will do this automatically when you drag a m:n relationship between 2 entities

Why a Third Entity?

- Store data about the relationship
- Usually represents a real-world entity (e.g., line items)
- Can have its own attributes
- Think of an n:m as two 1:n relationships

E.g., A line item would include the quantity (of a given item) and the sale price of that item.
Creating Relationships in Access

- To edit relationships in your DB
  - Click on Database Tools / Relationships
  - Click on Edit Relationships

You can also drag the primary key to the foreign key in the second table.

Will create the 1:n relationship.
Associative Table

- Has two foreign keys - one for each of the entities in the n:m relationship
- Contains other attributes consistent with the table

Textbook uses natural keys and a composite primary key for LineItems – we use non-natural keys

Queries of the many-to-many relationship usually require a join of more than 2 tables

A Three Table Join

- List the names of the three tables after FROM
- Specify two matching conditions with the associative table in both join conditions

```
SELECT * FROM Sales, LineItems, Items
WHERE Sales.ID = LineItems.saleID
    AND Items.ID = LineItems.itemID;
```
Example - Three Table Join

```
SELECT saleNo, saleDate, itemName, itemType, quantity, price
FROM Sales INNER JOIN (Items INNER JOIN LineItems ON Items.ID = LineItems.itemID) ON Sales.ID = LineItems.saleID
ORDER BY Sales.saleNo;
```

Example

- List the dates, names of items, quantity, and value of items

```
SELECT saleDate, itemName, quantity, price, quantity*LineItems.price AS total
FROM Sales, LineItems, Items
WHERE LineItems.saleID = Sales.ID
AND LineItems.itemID = Items.ID;
```
Example 2

- List the names of items, quantity, and value of items sold on January 16, 2003.

```sql
SELECT saleDate, itemName, quantity, price, quantity*LineItems.price AS total
FROM Sales, LineItems, Items
WHERE LineItems.saleID = Sales.ID
AND LineItems.itemID = Items.ID
AND saleDate = '1/16/2003';
```

Are We on Track Setup

- Set up a new database using the data from Chapter 5 in the Watson text.
- Import from ISE305 Web site (http://www.cs.stonybrook.edu/~ise305/xxxxxx.xlsx)
  - Items.xlsx
  - Sales.xlsx
  - ExportLineItems.xlsx (store as LineItems table)
- Add foreign keys to LineItems table and establish relationships

You can set the foreign keys in LineItems using values real values of the primary keys.
Are We on Track?

- Using Design View in Access, write a query to:
  - Compute the total value of sales of each item by date
  - Primary sort by date
  - Include a count of the items sold on each date

```
SELECT Items.itemName, Sales.saleDate, 
    Count(LineItems.lineNo) AS [Number Sold], 
    Sum(LineItems.price) AS [Total Sales]
FROM Sales INNER JOIN (Items INNER JOIN LineItems ON Items.ID = LineItems.itemID) ON Sales.ID = LineItems.saleID 
GROUP BY Items.itemName, Sales.saleDate 
ORDER BY Sales.saleDate;
```

Were We on Track?

```
SELECT Items.itemName, Sales.saleDate, 
    Count(LineItems.lineNo) AS [Number Sold], 
    Sum(LineItems.price) AS [Total Sales]
FROM Sales INNER JOIN (Items INNER JOIN LineItems ON Items.ID = LineItems.itemID) ON Sales.ID = LineItems.saleID 
GROUP BY Items.itemName, Sales.saleDate 
ORDER BY Sales.saleDate;
```

Notice the alternate syntax in the generated SQL.
Homework

- Using the Awards DB (multi-table), develop a query to:
  - Show number of awards, total budget by sponsor
  - Sorted by total budget

<table>
<thead>
<tr>
<th>sponsorName</th>
<th>totalBudget</th>
<th>numberAwards</th>
</tr>
</thead>
<tbody>
<tr>
<td>National Science Foundation</td>
<td>$101,197,276.00</td>
<td>251</td>
</tr>
<tr>
<td>US Department of Energy</td>
<td>$71,826,927.25</td>
<td>33</td>
</tr>
<tr>
<td>Empire State Development Corp</td>
<td>$60,914,498.00</td>
<td>4</td>
</tr>
<tr>
<td>National Institute of Allergy &amp; Infectious Disease</td>
<td>$55,867,955.47</td>
<td>29</td>
</tr>
<tr>
<td>National Institute of General Medical Sciences</td>
<td>$41,477,116.33</td>
<td>54</td>
</tr>
<tr>
<td>Dormitory Authority of the State of New York</td>
<td>$36,542,000.00</td>
<td>2</td>
</tr>
<tr>
<td>National Cancer Institute</td>
<td>$30,104,236.81</td>
<td>34</td>
</tr>
<tr>
<td>National Inst of Neurological Disorders &amp; Stroke</td>
<td>$14,882,561.67</td>
<td>33</td>
</tr>
</tbody>
</table>

Homework

- Assume that awards are allowed to have multiple PIs.
- Update the SBU_Awards DB accordingly
- Add additional PIs to some of the awards
- Hint:
  - Create a new PIs table with a handful of principle investigators
  - When you create the “in-between” table, add an additional attribute for Percentage (the percentage of the contract for the given PI)
EXISTS

- Existential qualifier
- Returns true or false
- Returns true if the table contains at least one row satisfying the specified condition
- Report all clothing items (type “C”) for which a sale is recorded

```sql
SELECT itemName, itemColor FROM Items
WHERE itemtype = 'C'
AND EXISTS
  (SELECT * FROM LineItems
   WHERE LineItems.itemID = Items.ID);
```

The sub-query is evaluated for each row of Items. Resulting row is only added to result table if EXISTS evaluates to TRUE

NOT EXISTS

- Returns true if the table contains no rows satisfying the specified condition
- Example: report all clothing items (type “C”) that have not been sold

```sql
SELECT itemName, itemcolor FROM Items
WHERE itemtype = 'C'
AND NOT EXISTS
  (SELECT * FROM lineitem
   WHERE item.itemno = lineitem.itemno);
```
Divide

- The universal quantifier
  - ForAll (uses the symbol ∀)
  - Meaning – something is true for everything or every relevant thing
- Not directly mapped into SQL
- Implement using NOT EXISTS
  - Find all items that have appeared in all sales becomes
  - Find items such that there does not exist a sale in which this item does not appear
    Not a very natural expression of the condition

Divide Example

- Find the items that have appeared in all sales
  
  ```sql
  SELECT itemNo, itemName FROM Items
  WHERE NOT EXISTS
  (SELECT * FROM Sales
    WHERE NOT EXISTS
    (SELECT * FROM LineItems
      WHERE LineItems.itemID = Items.ID
      AND LineItems.saleID = Sales.ID));
  ```

Pocket knife - Avon
Divide Reformulation

- Find the items that have appeared in all sales can be rephrased as
  - Find all the items for which the number of sales that include this item is equal to the total number of sales.

```sql
SELECT Items.ID, Items.itemName
FROM Items, LineItems
WHERE Items.ID = LineItems.itemID
GROUP BY Items.ID, Items.itemName
HAVING
  COUNT(LineItems.saleID) = (SELECT COUNT(Sales.ID) FROM Sales);
```

Set Operations

- Useful in combining values derived from two or more queries
  - UNION
    - Equivalent to OR
  - INTERSECT
    - Equivalent to AND
UNION Example

* List all items that were sold on January 16, 2003, or are brown

```sql
SELECT itemName FROM Items, LineItems, Sales
WHERE Items.ID = LineItems.itemID
    AND LineItems.saleID = Sales.ID
    AND Sales.saleDate = #2003-01-16#
UNION
SELECT itemName FROM Items
WHERE itemColor = 'Brown';
```

INTERSECT

* List all items that were sold on January 16, 2003, and are brown.

```sql
SELECT itemName FROM Items, LineItems, Sales
WHERE Items.ID = LineItems.itemID
    AND LineItems.saleID = Sales.ID
    AND Sales.saleDate = #2003-01-16#
INTERSECT
SELECT itemName FROM Items
WHERE itemColor = 'Brown';
```

**INTERSECT not supported by some DBMSs (e.g., Access)**