Teaching Case

Slushie World
An In-Class Access Database Tutorial

Donald E. Wynn, Jr.
University of Dayton
dwynn1@udayton.edu
Dayton, OH 45469

Renée M. E. Pratt
Washington & Lee University
prattr@wlu.edu
Lexington, VA 24450

Abstract

The Slushie World case study is designed to teach the basics of Microsoft Access and database management over a series of three 75-minute class sessions. Students are asked to build a basic database to track sales and inventory for a small business. Skills to be learned include table creation, data entry and importing, form and report design, switchboards, and numerous single and multiple table query designs. The resulting application is sufficient for students to identify a number of basic functions available to the small business, including order entry, sales reports, customer information, and more. The case is designed to be used in a course where the students have little or no prior relational database experience, such as an Introduction to MIS course.

Keywords: Tutorial, Database Basics, Microsoft Access

1. BACKGROUND

Twins Ed and Elaine Milano grew up in Tampa Florida. Every year, they grew more and more tired of having to bring their wallet to the local beach to buy drinks, such as slushies and sodas. Invariably, they'd forget to bring their wallet or forget that they didn't have any money in it. To solve this issue, Elaine (a business student at a local university) came up with the bright idea of opening a stand to sell drinks, but only to residents and tourists that registered as members of the stand. Thus, Slushie World was born.

Slushie World sells 10 flavors of slushies, 4 sodas, lemonade, and water. Each member would be allowed to purchase drinks using their membership id, up to a preset limit. At the end of each month, each member would be billed for their purchases via email. Initially, each member would be given a $50 credit limit initially, which could increase over time based on their prompt payment history.

Ed and Elaine started out trying to track inventory, sales, and members on a spreadsheet, but soon realized that there were significant limits to how they were able to manage their business this way. After an introductory MIS course, Ed decided to have a Microsoft Access database designed and built to manage the sales process and generate a number of reports to help run Slushie World. Since you have taken this course, he has asked

©2015 EDSIG (Education Special Interest Group of the AITP)

you to help develop the database, including the desired tables, queries, forms, and reports.

2. FUNCTIONAL REQUIREMENTS

The owners need to track the following things:

- Membership information, including member standing
- Beverage sales information
- Beverage inventory

3. TUTORIAL STEPS – PART ONE

Prior to completing this lab, please download the **slushie.xlsx, member.xlsx**, and **OldSalesDB.accdb** files and save in a new subdirectory called **SlushieWorld** (or whatever else you wish to name it).

1. Create a new database called ***SlushieDB***.
2. Create a new table within the ***SlushieDB*** database called **Member** consisting of four fields as shown in Figure 1.

   [See Figure 1, Appendix]

3. Enter the data in Figure 2, located in the Appendix, into the **Member** Table.

   [See Figure 2, Appendix]

4. Import the **Inventory** Table from the **Slushie.xlsx** spreadsheet.
5. Import members from the old member spreadsheet (**member.xlsx**) to the **Member** table.
6. Import the **SalesOrder** and **OrderLine** tables from the **OldSalesDB** database.
7. Build the relationships between the tables as show in Figure 3.

   [See Figure 3, Appendix]

8. Build a form to see information in the **Inventory** table.
9. Use the form to change the quantity of Raspberry Slushie on hand to zero.
10. Build a report to show all the inventory on hand (name the report "Product Inventory").
11. Build a report to show a current menu (name the report "Menu").
12. Build a form to see the order information.
13. Use the form to enter an order for Member #5, for 2 Lemon Slushies (prod #105) and 2 Orange Slushies (prod #107).

*** End of Part 1. See SlushieDB-Part 1 for the completed database to this point.

4. TUTORIAL STEPS – PART TWO

After seeing the progress made so far, Ed and Elaine have asked you to make a few changes to the member table in order to capture more information. They have added additional members and information, and also want to see more examples of the types of information that can be generated from the database. The names of the products can be found in parenthesis at the end of each step.

1. Add the following fields to the **Member** table: MemberCreditLimit, City, State.

   [See Figure 4, Appendix]

2. Modify the information in the member table to include the following information in the new fields.

   [See Figure 5, Appendix]

3. Generate a list of all members' first and last name, city, state, credit limit, and standing, sorted by last name. (*MemberList*)
4. Generate the same information, but sorted by state, then within state by credit limit (higher values first). (*MemberListByStateAndLimit*)
5. Generate a list of all members' first & last name, and credit standing, with a bad credit standing. (*BadCreditQuery*)
6. Generate to show all members' names and credit limit where their credit limit is over $50.00. (*HighLimitQuery*)
7. Generate a list of all members from Georgia or Florida. (*GAFLMemberQuery*)
8. Generate a list of all members NOT from Georgia or Florida. (*NonGAFLMemberQuery*)
9. Generate a list of all members with either a bad credit standing OR who are from Georgia or Florida. (*BadOrLocalMemberQuery*)
10. Generate a list of all members with a bad credit standing AND from either Ohio or Kentucky. (*BadAndLocalMemberQuery*)
11. Generate a list of members whose first name starts with "D". (*DNameMemberQuery*)
12. Generate a list of the order numbers and order dates, along with the id, first and

last name of the member who place the order, sorted by order number. (*OrderQuery*)
13. Generate the order and member information for order #5. (*OrderNum5Query*)
14. Generate the order and member information for a single order number, which Access will prompt us for. (*OrderPromptingQuery*)
15. Generate the same information, but with Access prompting us for the member ID. (*OrdersByMembersQuery*)
16. Show the product information (flavor, price, sales quantity) along with the previous order information from #11 above. (Note: Not the "product quantity") (*OrderAndProductQuery*)
17. Show the same product and order information, with the system prompting us for the order number. (*ProductsByOrderQuery*)
18. Generate a list showing the order number, quantity, and extended price (quantity * price) for each item on each order, sorted by order. (*ExtendedPriceQuery*)
19. Generate a list to show the total number of items and the total extended price by order, sorted in descending order (highest values first) by extended price. (*OrderTotalQuery*)
20. Modify the list to show only the five highest orders. (*FiveHighestOrderQuery*)
21. Build a report (not just a query), based on the information in problem #18. (*OrderTotalReport*)

*** End of Part 2. See SlushieDB-Part 2 for the completed database to this point.

5. TUTORIAL STEPS – PART THREE

Ed and Elaine really like the database application so far. But they would like a few more reports and queries to help with the management of the business, including getting information about best sellers and profit reports. They also want to make the application more user friendly.

1. Build a list to show how much each type of product (Slushie, Soda, and Other) have sold, both in terms of quantity and dollar sales. (*SalesByType*)
2. Show all the members that have ordered a "Lemon Slushie". We want to see their first and last name, member ID, the date they ordered this flavor, and how many they ordered. Sort this list by last name, first name order. (*LemonSlushieMembers*)
3. List all of the orders with more than 3 total cups sold, with orders having the highest quantity sales listed first. (*OrdersOver3Cups*)
4. Show the most sold beverages by quantity. (*Top3Beverages*)
5. Show the top 3 slushie flavors sold by quantity. (*Top3Slushies*)
6. Show the total sales revenue for each flavor, with the highest revenues listed first. (*SalesByFlavor*)
7. Show the total sales revenue by date. (*SalesByDate*)
8. Add a field to the **Inventory** table to show ProductCost, representing the costs for each beverage sold by SlushieWorld.

[See Figure 6, Appendix]

Enter data in the new table to show the following
   a. Slushies cost $3.00
   b. Sodas cost $2.00
   c. Lemonade costs $2.75
   d. Water costs $2.50
9. Now develop a query to see how much profit we've made. (*ProfitByOrder*)
10. Build a report around the ProfitQuery. (*ProfitReport*)
11. In order to better understand their customers, generate a list of all customers who have placed orders (first name, last name, city, then state), along with the total sales they have made, and the total number of orders they have made. Sort by state, then city. (*CustomerInfo*)
12. Build a form associated with the Member table. (*MemberForm*)
13. Make a switchboard with the items in Figure 6.

[See Figure 7, Appendix]

*** End of Part 3. See SlushieDB-Part 3 for the completed database.

6. CONCLUSION

Clearly, the existing process of recording information on paper and spreadsheets would not be enough to truly run Slushie World as a real business. Eventually, Ed and Elaine would like to add even more forms and reports to this database in order to add more functionality and outputs. For instance, they are considering

expanding to add more slushie stands on different beaches, which would change parts of the database to track each stand's sales and profits. But that is for later. For now, you will have developed a sufficient database for Ed and Elain's current needs by completing each of the above tasks. In addition, you should now have a working knowledge of developing Microsoft Access-based database applications for other uses.

Note: Teaching Notes and Case Supplements are available by contacting the authors

Appendix

Field Name	Data Type	Description
MemberID	Autonumber ; Primary Key	Member ID Number
MemberFirstName	Text; 25 characters	Member First Name
MemberLastName	Text; 25 characters	Member Last Name
MemberStanding	Yes/No Field	Does member have good credit?

Figure 1. Member Table Data Description

MemberID	MemberFirstName	MemberLastName	MemberStanding
(auto entered)	(your name)	(your name)	Yes
"	Darryl	Martin	Yes
"	Jerry	Campbell	No
"	Tony	Roland	No
"	Casey	Palmer	Yes

Figure 2. Member Table Data

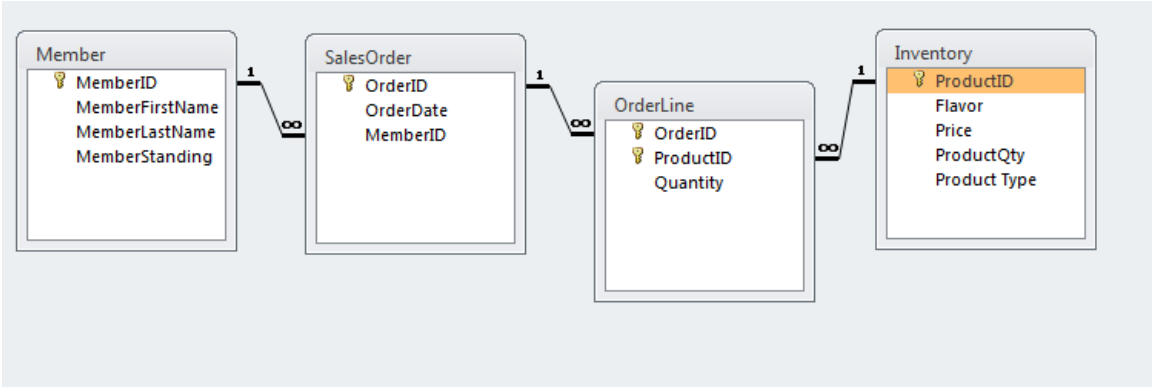

Figure 3. Relationship Table

Field Name	Data Type	Description
MemberCreditLimit	Currency	Member's Maximum Credit Limit
MemberCity	Text; 30 characters	City where Member lives
MemberState	Text; 2 characters	State where Member lives

Figure 4. Additional Member Table Attributes

MemberID	Member FirstName	Member LastName	Member Standing	Member City	Member State	Member CreditLimit
1	(Your)	(Name)	TRUE	Dayton	OH	$200.00
2	Darryl	Martin	TRUE	Houston	TX	$100.00
3	Jerry	Campbell	FALSE	West Palm Beach	FL	$100.00
4	Tony	Roland	FALSE	Orlando	FL	$100.00
5	Casey	Palmer	TRUE	Tampa	FL	$50.00
6	Dee	O'Durant	TRUE	Tampa	FL	$50.00
7	Terry	Dactle	FALSE	Savannah	GA	$50.00
8	Hal E.	Lewya	TRUE	Port St. Lucie	FL	$50.00
9	Nick	O'Teen	FALSE	Columbia	SC	$50.00
10	Izzy	Able	TRUE	Atlanta	GA	$50.00
11	Harry	Kerry	TRUE	Columbia	SC	$50.00
12	Moe	Turoil	FALSE	San Diego	CA	$50.00

Figure 5: Additional Member Data

Field Name	Data Type	Description
ProductCost	Currency	Cost of Goods Sold per unit

Figure 6. Additional Inventory Table Attribute

Main Menu		
*	Add New Member (*MemberForm*)	
*	Review Current Members (*MemberForm*)	
*	Review Product Inventory (*InventoryForm*)	
*	Show Existing Orders (*SalesOrderForm*)	
*	Add New Order (*SalesOrderForm*)	
*	Reports Menu	
	+	Product Inventory Report
	+	Menu
	+	Order Total Report
	+	Profit Report

Figure 7. Switchboard Menu Items