**IS 312 Database (with Access) Tutorial 1 and Demo 1:**

**Table Structure; Relationships between Entities; Creating Data Table**

*Dr. Yüe “Jeff” Zhang*Version 15.2, January 22, 2018

1. **Database Hierarchy, and Data Table Structure**

Opening note: The differences between Access (and any DBMS) and Word or Excel: In Word or Excel, you open the software and begin typing letters/numbers to create a memo or a spreadsheet, then save it with a file name - you start directly with the element CONTENTS of the file or “bottom-up”. In Access (a DBMS), instead, you start with the STRUCTURE: a BLANK database (DB), then table structures, finally data – “Top-down”. Here is an analogy:

|  |  |
| --- | --- |
| **Analogy** | **In Access (a DBMS)** |
| **Task:** Dept of Acct & IS plans to use JH2214 as the Acct/IS library | **Task:** You plan to build a database (such as a Student DB) |
| ***Step 1:*** Clear the ROOM JH 2214, put a label on its door saying "Acct & IS Library" | ***Step 1:*** Create a blank DATABASE, name it "StudentDB" |
| ***Step 2:*** Move in BOOKSHELVES, label individual shelves as "Accounting" or "Info Sys", with individual shelf-rows being "Auditing", "Tax", "CPA exam prep", "Database", "Networking", "E-commerce", etc. | ***Step 2:*** Create a table inside the just created DB, build the table STRUCTURE - the names of fields, the data types of the fields, the size (length) of a field, etc, and save the table as "StudInfo". (Then build a second table and save it as "StudAcademicPerfom", etc, etc.) |
| ***Step 3:*** Move the BOOKS into the library room and put the books in their corresponding shelves/rows. | ***Step 3:*** Enter DATA RECORDS in their corresponding tables. |
| ***Step 4:*** Now the library is ready for use! Acct or IS books can easily be located from the shelves/rows with the labels. | ***Step 4:*** Now the StudentDB is ready for use! Records can easily be located using various criteria on certain attributes (fields). |

1. Create database 🡺 (2) Build/define table structure 🡺 (3) Fill table w data records

 [Save with DB file name] [Create w field name] [Save w table name]

|  |
| --- |
| Concepts introduced in Figure 1 (next page): Field Name, Data Type, and Field Properties |

Field name: identifies the data that is entered into the field. [Field names become column headers in the table]

Data type: determines the type of data that can be entered, and the operations that can be performed on that data.

Common types of data: Number field, Text field, Memo field, Currency field, Yes/No field, AutoNumber field, Hyperlink field.
Any data field needs to be defined as one of the above types.
In Figure 1, the data type of RestaurantID (restaurant ID) is “text” (*Why? Why ID numbers are not of the type “Number”?*). Same question can be asked about Zipcode. [Note sometimes ID is “AutoNumber]

**Field Properties**: Required? Size (length in characters)? If of number or currency type, decimal places? If of date type, date format? ...

**Figure 1. The creation of a data table**

|  |  |
| --- | --- |
| **Figure 1a**. Step 1 - create a blank database (DB): | **Figure 1b**. Step 2 - create a blank table with structure: |
|

|  |
| --- |
| **RestaurantsDB** |
| (A table to be created) |

 |

|  |
| --- |
|  **RestaurantsDB**= “Table\_Restaurants” |
|  **tblRestaurants** |
|

|  |  |  |
| --- | --- | --- |
| RestaurantID | Text | ... |
| Address | Text | ... |
| City | Text | ... |
| State | Text | ... |
| Zipcode | Text | … |
| AnnualSales | Currency |  |
| DateOpened | Date/Time | **…** |
| … | … | … |

Step 2.2: Defining **field properties** – Decimal placeDate format…etc |

 |

**Figure 1c. Step 3 - "populate" the data table (entering data records):**

|  |
| --- |
| **tblRestaurants** |
|                  RestaurantsTable |
|

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RestaurantID | Address | City | State | Zipcode | AnnualSales | DateOpened | ... |
| R0001 | 2345 SW 98 Street | Miami | FL | 33133 | $575,000 | 6/1/1998 | ... |
| R0002 | 3487 Main Highway | Pensacola | FL | 32504 | $350,000 | 4/15/1999 | ... |
| … | … | … | … | … | … | .. | ... |
| ... | ... | ... | ... | ... | … | ... | ... |

 |

To create another table, one only needs to repeat the above steps 2 and 3 with corresponding field names (with appropriate data types) and values for fields (i.e., data).

After being created, a DB can be used for three functions: Slide PI3-25~26

1. **Relationships between Entities**

Study the table by comparing left and right columns

1. One-to-one and one-to-many relationships

|  |  |
| --- | --- |
| A1. One-to-one relationshipExample: STUDENT and PARKING-PERMITSTUDENT entity has many instances (many students), PARKING-PERMIT entity also has many instances (many parking permits). Each individual student (each instance in the STUDENT entity) can be associated with ONE parking permit (one instance in the PARKING-PERMIT entity) – a student doesn’t have to have a parking permit (such as the 3rd & 5th students in the figure below); but if s/he does have parking permit, s/he has only ONE permit. On the other hand, **each** permit is assigned to ONLY ONE student.================So, from left to right: “each instance on LEFT can be associated with only ONE instance on the RIGHT;”from right to left: “EACH instance on the RIGHT can be associated with ONLY ONE instance on the LEFT.” | A2. One-to-many relationshipExample: FACULTY advises STUDENTSTUDENT entity has many instances (many students), FACULTY entity also has many instances (individual professors). Each individual faculty (each instance in the FACULTY entity) can be associated with MANY students (many instances in the STUDENT entity) – a professor doesn’t have to advise a student (such as the 3rd professor in the figure below), but if s/he does, s/he may advise one or many students.On the other hand, each student is advised by ONLY ONE professor.==================So, from left to right: “each instance on the left can be associated with MANY instances on the right;”Comparefrom right to left: “EACH instance on the right can be associated with ONLY ONE instance on the left.” |
| STUDENT has PARKING-PERMITPARKING-PERMIT is assigned to STUDENT1. 1

1 1 |   FACULTY Advises STUDENT STUDENT Is Advised by FACULTY1. M

1 1 |

A3. How a one-to-many relationship is maintained between tables (note: tables represent entities) in a database?

– Through **a common column** in the related tables. Between the participating tables –

|  |  |
| --- | --- |
| On the ONE-side, also 🡪 | On the MANY-side. |
| **Primary key** of the ONE-side, 🡪 | **Foreign key** of the MANY-side. |
| Example: |
| Fac-ID: **Primary key** of FACULTY table | Fac-ID: **Foreign key** of STUDENT table |

So, “the primary key on the one-side must appear in the many-side as the foreign key.”

M

1

STUDENT

FACULTY

Advises

Primary key

**ONE-side**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fac-ID** | **Fac-LN** | **Fac-FN** | **Rank** | **Office** | **Extension** |
| F01 | Ackerman | Bill | Professor | JH 3133 | 6010 |
| F02 | Cohen | Debbie | Associate Professor | JH 4133 | 8022 |
| F03 | Edwards | Frank | Assistant Professor | JH 3113 | 4567 |
| F04 | Green | Helen | Professor | JH 4213 | 3456 |
| F05 | Zhang | Jeff | Professor | JH 3218 | 6050 |

The primary key-foreign key pair (the common column)

**MANY-side**

 Foreign key

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S-ID** | **S-LN** | **S-FN** | **Gender** | **DOB** | **Major** | **Phone** | **Fac-ID** |
| S001 | Adams | Ben | M | 01/23/1987 | Acct | 993-1234 | F01 |
| S002 | Chen | David | M | 02/24/1992 | Fin | 832-6666 | F03 |
| S003 | Evans | Francesca | F | 03/31/1989 | Acct | 676-3333 | F02 |
| S004 | Gonzalez | Juan | M | 04/04/1991 | Fin | 993-3214 | F03 |
| S005 | Jung | Yun-Il | M | 05/06/1993 | IS | 993-9876 | F05 |
| … |  |  |  |  |  |  |  |
| S799 | Zhang | Min | F | 11/11/1996 | IS | 832-0123 | F05 |

[Please examine which professor advises which students (or none)]

**IS 312 Access Database Demo 1: Creating a Table; Creating Relationship**

Dr. Yüe “Jeff” Zhang

Version 13, January 31, 2016

**I. Creating Tables**

“Road Map”:

1. Create a BLANK database with a name (File – Save As – Type ***RestaurantDB*** for File Name).

|  |  |
| --- | --- |
| Blank desktop database |  |

2. Inside the DB, create a blank table, give it a name (“Restaurants”).

 2.1 At this point, click the “Home” tab to go to the table.

3. Inside the table, create fields - use the ***Design View***: 

To CIT majors: Comp Sci like to name a DB table “tblName”, such as “tblRestaurants”.

 3.1 Define data type for each field;

 3.2 Set up properties for the fields (if applicable).

4. Enter data in the ***Datasheet View***.

**Key points:**

1. Fields: fields have different data types – number, text, date, currency, etc.
2. Primary Key: unique identifier (a column that services to differentiates a record from the others)
3. Field Properties
	1. Required field
	2. Input mask: 1, facilitates data entry (phone # etc); 2, helps to avoid basic mistakes.

Input mask definition characters:

|  |  |
| --- | --- |
| 0 – required digit (0-9), no + or – signs | > - cause letters that followed to become uppercase |
| L – required letter | A – required letter or digit |

**Create the table with the following fields and properties:**



 Pay attention to:

1. Format >;

2. Input Mask L0000;

3. Validation Rule;

4. Validation Text

5. Required No – any thoughts/comments on this?

6. Don’t forget the primary key!

**II. Creating Relationship**

[Assuming the two tables – Restaurants & Franchises - already exit]

* + Click (1) “**Database Tools**” – (2) “**Relationships**” 
	+ You will see this displayed (Primary key – foreign key pair indicated by arrows):



To

From

* + **Drag** the primary key, to drop it at the foreign key, you will see the following:



* + In the above dialog box, click the box to select:
		- Enforce Referential Integrity.
	+ Click Create to create the relationship. The result will be as the figure below:



Note the “1 – ∞” link, which means “1-M” or “one-to-many”.

[Can you precisely interpret what “one-to-many” means for this scenario/example? Try it]