TU2983: Advanced Databases

Lab Notes 1:

Database Refresher:
Crash Course in Microsoft Access

Written by:

Hafiz Mohd Sarim

Centre of Artificial Intelligence Technology
Fakulti Teknologi dan Sains Maklumat
Universiti Kebangsaan Malaysia

Table of Contents

PART 1: Introduction

A. Lab Objectives

1. The "TU2983: Advanced Databases" course expands and continues from the pre-requisite "TT1964: Databases" course, which previously covered database development and data manipulation using Structured Query Language (SQL) within enterprise-class database management systems, such as Oracle, IBM DB2, Microsoft SQL Server, or MySQL.

2. The lab modules for TU2983 will focus on database application programming, and students of TU2983 are required to already be familiar with SQL, as it will not be repeated for this course.

3. The objective of the TU2983 lab modules is to teach the development of fully functional two-tier software programs, also known as "**database applications**", which utilizes a database as its core data management component.

4. For practical development of database applications, the lab modules for TU2983 will alternate between the following when necessary:
    a. A file-based database management system (i.e. portable databases), when it is necessary to easily transport the database, or include it as part of a software package.
    b. An enterprise-class databases (i.e. server databases), when it is necessary to centrally locate the database and provide data access over the network.

5. The portable database used for the TU2983 lab modules is **Microsoft Access**, and the minimum version required is Microsoft Access 2007, which will produce databases that have the '***.accdb**' file extension. These lab notes were written using Microsoft Access 2013 as the portable database. However, the steps demonstrated in these lab notes can also be used for almost all versions of Microsoft Access starting from Microsoft Access 2007.

6. The server database used for the TU2983 lab modules is IBM DB2 Enterprise Server Edition, and the minimum version required is IBM DB2 v9.7. These lab notes were written using IBM DB2 v9.7 as the server database. However, the steps demonstrated in these lab notes can also be used for almost all versions of IBM DB2.

7. FOR SUBMITTING ASSIGNMENTS: All students will be instructed of the type of database (portable or server) you must use for submitting your assignment.

B. Database Object Naming Conventions

1. Standardized prefixes and suffixes will be appended to the names of database objects, according to the type of object created, for two purposes:

   - To ease the identification of a database object through its name alone, when referenced in SQL-code or VB-code.

   - To prove originality of a student's work and ownership of the database through the student's matric number.

2. The database object naming conventions are as follows:

   i. All object names should be in **UPPERCASE** (no lowercase letters) and must not contain any 'spaces', punctuation, or special characters. All 'spaces' must be replaced by the 'underscore' (_) character.

   ii. All object names must begin with a **prefix** to indicate the object type, and end with a **suffix** to indicate originality and ownership, as follows:

Object Type	Name Prefix	Name Suffix	Name Format	Example
Database file	**DB_**	**_<matric>**	**DB_<name>_<matric>**	DB_UNIVERSITY_A123456
Table	**TBL_**	**_<matric>**	**TBL_<name>_<matric>**	TBL_STUDENTS_A123456
Attribute	**FLD_**	*none*	**FLD_<name>**	FLD_COURSE
Query	**QRY_**	*none*	**QRY_<name>**	QRY_EXAMRESULTS

IMPORTANT MANDATORY RULE :

All database file names and table names MUST end with the student's matric number as the suffix to indicate the creator of the database object, as shown in the example above.

C. Student Requirements for this Lab

1. All students must bring the following equipment for the lab:

   - **MANDATORY REQUIREMENT:** One USB hard drive / thumb drive / pen drive (to save your work).

2. All students must have the following requisite knowledge:

   - **MANDATORY REQUIREMENT:** Using the Microsoft Windows operating system

   - **MANDATORY REQUIREMENT:** Writing Structured Query Language code, or SQL.

PART 2: Creating a Database in Microsoft Access

A. Creating a new Database File

1. Create your working folder. In the desktop, click **'Start' > 'Computer' > 'C:'**. Create a folder or directory with the name ***'<matric>_<your project name>'***. For this example, type **'c:\a123456_facultyrecords'**. Remember this folder location as it will be used throughout these lab notes.

2. Start Microsoft Access. In the Windows desktop, click **'Start' > 'All Programs' > 'Microsoft Office' > 'Microsoft Office Access [version]'**.

3. In the Microsoft Access start page, click **'Blank database'** or **'Blank desktop database'**.

4. In the file name field, type in your database file name, with the format **'db_<name>_<matric>.accdb'**. For this example, type **'DB_FACULTYRECORDS_A123456.accdb'**.

5. Click on the folder icon next to the file name field, and browse to your working folder. For this example, browse to **'c:\a123456_facultyrecords'**, and click 'OK'.

6. Click '**Create**'. When the main Microsoft Access interface appears, you have successfully created a blank database.

B. Creating and Designing a Table

1. When the Microsoft Access interface appears for the first time, you will see a tab called '**Table1**'. This is not an actual data table, but a 'new table placeholder'.

> ***OPTIONAL***: *Alternatively, you can create a new table manually. Click on the '**X**' on the right side of the '**Table1**' tab to close the '**Table1**' new table placeholder. On the main menu, click the '**CREATE**' tab. In the **'CREATE'** toolbar, click '**Table**'. You will once again receive the **'Table1'** new table placeholder.*

2. You can create a new table using this new table placeholder, by clicking the **'View'** or **'Design View'** icon (which has the ruler and pencil) on the **'HOME'** toolbar.

3. When the **'Save as'** window appears, type your table name with the format **'TBL_<name>_<matric>'**. For this example, type **'TBL_STUDENTS_A123456'**. Note, the **'Table1'** tab name has also changed to **'TBL_STUDENTS_A123456'**, and you will now be in the **'Design View'** mode.

4. The **'Design View'** shows you the design of each attribute (or field) in your table. Each new table will be given one default field 'ID':

   - This **'ID'** field has the '**AutoNumber**' data type. Attributes with the **'AutoNumber'** data type actually hold an integer value that automatically increments as new rows of records are added. Attributes with the **'AutoNumber'** data type also cannot be manually edited.

- On the left side of the **'ID'** field name, you may notice a key icon. This indicates that the **'ID'** field is a **Primary Key** attribute, which cannot be null and cannot have duplicate values.

5. To change the name of an attribute, overwrite the **'ID'** field name, and type **'FLD_MATRIC'**.

6. To add a new attribute, in the blank row below **'FLD_MATRIC'**, type **'FLD_NAME'**. This new **'FLD_NAME'** attribute will automatically be given the **'Short text'** or **'Text'** data type, which holds text string data. The length of characters for this attribute can be seen in the 'Field properties' list below the attribute list. We are now ready to add data into the 'TBL_STUDENTS_ A123456' table.

C. Adding and Changing Table Data

1. Switch to the **'Datasheet view'** by clicking the **'View'** or **'Datasheet view'** icon (which has a table grid). When a message appears to save the table, click **'Yes'**. You will be brought back to the **'TBL_STUDENTS_A123456'** table.

> **TIP**: You can switch back-and-forth between the 'Design view' and 'Datasheet view' by clicking the same 'View' icon. You will only need to save the table when any changes are made in the 'Design view'.

2. The first row of your table will have an asterisk (*****) to its left. This row is not an actual record, but is the **'new record placeholder'**. You can add a new record by typing into any of the fields in this 'new record placeholder' row.

3. In the new record placeholder row, click in the empty cell under **'FLD_NAME'** and type in **'Ahmad'**. As you type you will notice the following:

   - The icon on the left of the row changes to a pencil. This **'edit'** icon indicates that the data row is currently being edited.

   - The cell under the **'FLD_MATRIC'** attribute automatically changes to '1'. Attributes with the AutoNumber data type will automatically be filled with the next incremented number, and cannot be manually altered.

EXERCISE 1: **Go to the 'new record placeholder' in next row, and keep typing in data until your table looks like this:**

FLD_MATRIC	FLD_NAME
1	Ahmad
2	Ali
3	Siti
4	Lim
5	Raja
6	Anna

INFO: In Microsoft Access, typing data into a new row is equivalent to executing an **SQL INSERT** command. Changing data in an existing row is equivalent to executing an **SQL UPDATE** command. Furthermore, you do not need to manually **COMMIT** any changes to the table. All **INSERT** and **UPDATE** changes are automatically **COMMIT** when you leave the **'edit'** mode of a data row, and all data is automatically saved. You do not need to save the table manually after adding or changing data in the **'Datasheet view'**. You only need to save the table after you alter the table's design or attribute's design in the **'Design View'**

D. Changing Attributes Data Types and Creating Primary Keys

1. In the **'Datasheet view'** of table **'TBL_STUDENTS_A123456'** try to add a new record as shown below:

FLD_MATRIC	FLD_NAME
A107	Abu

2. You will notice that you cannot add this row above because you cannot edit the **'FLD_MATRIC'** field. You will need to change the data type for this field.

3. To change an attribute, switch to **'Design view'**. For **'FLD_MATRIC'** attribute, click its **'Data Type'**. Click the arrow next to 'AutoNumber' and select 'Short Text' or 'Text'.

4. In the **'Field Properties'** list, change the **'Field Size'** from 255 to 10. This reduces the space used by this attribute in the database and restricts the length of the field.

5. Switch back to **'Datasheet view'** and now add the row above.

EXERCISE 2: After adding the row above, change the earlier rows so that the table now looks like this:

FLD_MATRIC	FLD_NAME
A101	Ahmad
A102	Ali
A103	Siti
A104	Lim
A105	Raja
A106	Anna
A107	Abu

6. Close the **'TBL_STUDENTS_A123456'** table. To close a table, click the '**X**' button on the far-right of the currently opened table tab.

INFO: **Primary Key fields** which act as **identifiers** should preferably have a string or text data type and not numeric data types like AutoNumber, Integer, or Double. Numeric data types have a limited length restriction (e.g. cannot exceed 2147483647 for an Integer), cannot have leading zeros (cannot type 01933334444 as a phone number), and cannot have leading or training characters (cannot type A123456 as a matric number). Numeric data types should only be used in attributes which will later be used for calculation, for example to calculate the total credits from the **'fld_credits'** field.

7. Create another table called **'TBL_DEPARTMENT_A123456'** and go to its **'Design View'**

8. In **'Design View'** delete the default **'ID'** attribute. To delete an attribute, first highlight the attribute by clicking on the box on the left of the attribute name. When the **'ID'** attribute is highlighted, press the **'Delete'** key on your keyboard. When you receive the warning that you are about to delete a primary key. Click **'Yes'**.

9. Add an attribute called **'FLD_DEPT_ID'** and give it a **'Short Text'** or **'Text'** data type.

10. Add another attribute called **'FLD_DEPT_NAME'** and give it a **'Short Text'** or **'Text'** data type.

11. Make **'FLD_DEPT_ID'** a primary key. To make an attribute become a primary key, highlight the attribute **'FLD_DEPT_ID'** and click the **'Primary Key'** icon on the **'HOME'** toolbar. Alternatively, you can also right-click on the box on the left of the attribute's field name and click **'Primary Key'.**

EXERCISE 3: Populate the **'TBL_DEPARTMENT_A123456'** table by adding data until it looks like the following:

FLD_DEPT_ID	FLD_DEPT_NAME
TS	Information Systems
TK	Computer Science
TR	Industrial Computing

EXERCISE 4: Add a new table **'TBL_COURSES_A123456'** and add the following data rows. Make **'FLD_COURSE_CODE'** the primary key for the table. The attribute 'FLD_CREDITS' should have a numeric data type:

FLD_COURSE_CODE	FLD_COURSE_NAME	FLD_CREDITS
TT1964	Databases	4
TU2983	Advanced Databases	3
TK1914	C Programming	4
TK2013	Software Engineering	3
TR1313	Mathematics	3
TR1713	Statistics	3

PART 3: Managing Table Relationships

A. Creating Relationships Automatically with the Lookup Wizard

1. In Microsoft Access, relationships between tables can be created automatically when adding a foreign key attribute, if this attribute is added using the **Lookup Wizard**.

> **INFO:** Generally, a **foreign key** attribute is an attribute in a table that references an attribute value that is a primary key in a different table.

2. Open the table **'TBL_STUDENTS_A123456'** in **'Design View'**.

3. Add a new field **'FLD_DEPT'**. Click on its **Data Type**, and click on the arrow to change it to **'Lookup Wizard...'**

4. When the **'Lookup Wizard'** window opens, select the first option that says "**I want to lookup field to get the values from another table or query**", and click **'Next'**.

5. In the table list, highlight **'TBL_DEPARTMENT_A123456'**, and click '**Next**'.

6. In the next screen, highlight **'FLD_DEPT_ID'** in the **'Available Fields'** list, click the **'>'** arrow, and click **'Next'**.

7. In the next screen, click the drop-down list next to '**1**' and select **'FLD_DEPT_ID'**, and click **'Next'**.

8. In the next screen, click **'Next'**.

9. In the next screen, leave the label for the lookup field as **'FLD_DEPT'**, and click **'Finish'**. Click **'Yes'** when you receive the **'Save'** message.

10. Switch to the **'Datasheet view'** of the table **'TBL_STUDENTS_A123456'**.

11. Click the empty cell under **'FLD_DEPT'** for student **'Ahmad'**. Click the arrow and select **'TS'** from the drop-down list.

EXERCISE 5: Change the department for all students as follows:

FLD_MATRIC	FLD_NAME	FLD_DEPT
A101	Ahmad	TS
A102	Ali	TK
A103	Siti	TR
A104	Lim	TS
A105	Raja	TK
A106	Anna	TR
A107	Abu	TS

12. Close the table **'TBL_STUDENTS_A123456'**.

> **TIP**: In step 3.A.6, you can select more than one attribute by repeatedly clicking the '**>**' arrow, or select all attributes by clicking the '**>>**' arrow. If you do this, in step 3.A.8, untick the **'Hide key column'** box to show all selected attributes. When you click **'Next'** and go to the next screen, you will be asked which value to store in the lookup field. Click on the primary key field **'FLD_MATRIC'** and click **'Next'** > **'Finish'**. When you switch to **'Datasheet View'** and use the drop-down list in the **'FLD_MATRIC'** field, it will also show you the name of the student. This is useful when you cannot remember what record a foreign key value is actually referring to.

B. The Relationship View and Creating Manual Relationships

1. To view the relationships between tables, on the main menu click the **'DATABASE TOOLS'** tab, and click the **'Relationships'** icon.

2. In this **'Relationship'** view, you will see that the tables **'TBL_STUDENT_A123456'** and **'TBL_DEPARTMENT_A123456'** has already been added.

3. A link connects **'FLD_DEPT_ID'** in **'TBL_DEPARTMENT_A123456'** with **'FLD_DEPT'** in **'TBL_STUDENT_A123456'**. This link represents the relationship between these two tables.

4. Close the **'Relationship'** view, by clicking the **'X'** button on the far-right of the **'Relationship'** tab.

5. Open the table **'TBL_COURSES_A123456'** in **'Design View'**, and add a new attribute **'FLD_MANAGING_DEPT'** that has a **'Short text'** or **'Text'** data type. This time, DO NOT use the **'Lookup Wizard...'** to automatically create a relationship.

EXERCISE 6: Switch to the **'Datasheet view'** of the table **'TBL_COURSES_A123456'**, and type in the data for **'FLD_MANAGING_DEPT'** as follows, and close the **'TBL_COURSES_A123456'** table.

FLD_COURSE_CODE	FLD_COURSE_NAME	FLD_CREDITS	FLD_MANAGING_DEPT
TT1964	Databases	4	TS
TU2983	Advanced Databases	3	TS
TK1914	C Programming	4	TK
TK2013	Software Engineering	3	TK
TR1313	Mathematics	3	TR
TR1713	Statistics	3	TR

6. Close the **'TBL_COURSES_A123456'** table and open the **'Relationship'** view.

> **TIP**: Make sure you close all opened tables before opening the Relationship view. Relationships cannot be edited if there is a table opened in **'Design view'** or if a table is currently in **'Edit'** mode in **'Datasheet view'**.

7. In the **'DATABASE TOOLS'** toolbar, click **'Show Table'**.

8. In the **'Show Table'** list, highlight **'TBL_COURSES_A123456'**. Click **'Add'** and click **'Close'**. The structure for **'TBL_COURSES_A123456'** will appear in the **'Relationship'** view.

9. To manually create a relationship, click-and-drag the attribute **'FLD_DEPT_ID'** from **'TBL_DEPARTMENT_A123456'** and release-click over **'FLD_MANAGING_DEPARTMENT'** in **'TBL_COURSES_A123456'**.

10. When the **'Edit Relationships'** window appears, click **'Create'**. A link will now appear between the two tables to show that the relationship has been established.

> **TIP:** Use the **'Lookup Wizard'** to create relationships between as many tables as possible, before adding any data into these tables. Manually created relationships made in the **'Relationship'** view do not have the data lookup function using drop-down list that are automatically created with the **'Lookup Wizard'**. Data entry into these manually created foreign key fields need to be typed in manually.

> **INFO**: The first time a relationship is created, whether manually or automatically with the **'Lookup Wizard'**, the generated relationship is only useful for referencing data. In this initial state, non-existent data references can still be entered in the referencing table, even though the data reference, or primary key value, does not exist in the referenced table. This may cause errors later on when referencing data. To prevent reference errors, you should enforce referential integrity and data integrity in all relationships.

C. Enforcing Referential Integrity

1. You can force a database user to enter only valid foreign key references in a table by enforcing the referential integrity of the relationship.

2. Open the 'Relationship' view.

3. Click on the relationship linking **'TBL_STUDENT_A123456'** and **'TBL_DEPT_A123456'** to highlight it.

4. With the relationship highlighted, click the **'Edit Relationship'** icon on the **'DATABASE TOOLS'** toolbar. Alternatively, you can also right-click on the highlighted relationship, and click **'Edit Relationship'**.

5. In the **'Edit Relationship'** window, tick the **'Enforce Referential Integrity'** box, and click **'OK'**.

6. A '**1**' and '**∞**' icon will appear on each edge of the highlighted relationship. This shows that a **One-to-Many (1:M)** relationship has been created between **'TBL_STUDENT_A123456'** and **'TBL_DEPT_ A123456'**.

> **INFO**: The cardinality of a relationship (**1:M**, **1:1**, or **M:N**) are automatically determined by type of attributes are involved in a relationship. A relationship between a primary key and another primary key, is automatically **1:1**. A relationship between a primary key and a non-primary key OR a subset of a composite primary key, is automatically **1:M**. A relationship between a non-primary key with another non-primary key OR with a subset of a composite primary key, is automatically **M:N**.

7. Close the 'Relationship' view, and open the table **'TBL_STUDENT_A123456'** in **'Datasheet view'**.

8. Try to type in the following record:

FLD_MATRIC	FLD_NAME	FLD_DEPT
A108	Sarah	TU

9. Microsoft Access will prevent you from entering the data above because the value **'TU'** does not exist as a primary key value in the table **'TBL_DEPARTMENT_A123456'**.

EXERCISE 7: Enforce the referential integrity in the relationship linking **'TBL_DEPARTMENT_A123456'** and **'TBL_COURSE_A123456'.**

TIP: You can only enforce referential integrity if the referencing table only contains valid values in the foreign key field, or a null value. Therefore, you should ideally enforce referential integrity immediately after the foreign key field is added to the referencing table, which is when values for this foreign key attribute are still blank (null).

D. Enforcing Data Integrity with Cascade Update

1. One impact of enforcing referential integrity is that any primary key value that is involved in a relationship cannot be changed if it has a related record in a referencing table.

2. Open the table **'TBL_DEPARTMENT_A123456'** in **'Datasheet View'**, and try to change the **'FLD_DEPT_ID'** value for the 'Information Systems' department from 'TS' to 'TU'.

3. Microsoft Access will prevent you from changing this data, because doing so would cause referencing records in the **'TBL_STUDENT_A123456'** and **'TBL_COURSE_A123456'** tables to violate referential integrity rules.

4. Close all opened tables, and open the **'Relationship'** View.

5. To allow changes to a referenced primary key, we will let Microsoft Access automatically change all foreign key references to an old primary key value, and replace it with the new primary key value. This is called **'Cascade updates'**.

6. Click on the relationship between **'TBL_DEPARTMENT_A123456'** and **'TBL_STUDENT_A123456'**, and click **'Edit Relationship'**.

7. In the **'Edit Relationships'** window, tick **'Enforce Referential Integrity'** and tick **'Cascade Update Related Fields'**. Click **'OK'**.

8. Click on the relationship between **'TBL_DEPARTMENT_A123456'** and **'TBL_COURSE_A123456'**, and click **'Edit Relationship'**.

9. In the **'Edit Relationships'** window, tick **'Enforce Referential Integrity'** and tick **'Cascade Update Related Fields'**. Click 'OK.

10. Close the **'Relationship'** view, and open the table **'TBL_DEPARTMENT_A123456'** in **'Datasheet view'**.

11. Now change the **'FLD_DEPT_ID'** value for the **'Information Systems'** department from '**TS**' to '**TU**', and close the table.

12. Open both **'TBL_STUDENT_A123456'** and **'TBL_COURSE_A123456'** tables in 'Datasheet View'. You will find that all foreign key references to 'TS' have been automatically updated to 'TU'.

> **TIP**: Cascade Updates are generally safe to use in the database itself, since reference data is uniformly changed and propagated throughout related tables. This also eliminates the need to manually change all references in all tables, as long as these references are in foreign key attributes that are linked using relationships. Attributes that reference the changed data, but which are not linked using relationships are not changed.

E. Using the Lookup Wizard to Create Limited Lists

1. The Lookup Wizard does not necessarily have to reference actual tables to get valid data values. The Lookup Wizard can also be used to create short lists of valid data values, which are static and are almost never updated.

Exercise 8: Create a new table called **'TBL_REGISTATION_A123456'** that has the following attributes and data types:

Field Name	Data type
FLD_SESSION	"Short text" or "Text"
FLD_SEMESTER	"Short text" or "Text"

2. In the 'Design View' of table **'TBL_REGISTRATION_A123456'**, click the 'Data Type' for the attribute **'fld_session'**.

3. When the **'Lookup Wizard'** window opens, select the second option that says "**I will type in the values that I want**", and click **'Next'**.

4. In the Column list, under **'Col1'** type in the values "**20132014**" and "**20142015**" underneath it, and click **'Next' > 'Finish'**.

5. In the 'Design View' of table **'TBL_REGISTRATION_A123456'**, click the 'Data Type' for the attribute **'FLD_SEMESTER'**.

6. When the **'Lookup Wizard'** window opens, select the second option that says "**I will type in the values that I want**", and click **'Next'**.

7. In the Column list, under **'Col1'** type in the values "**1**" and "**2**" underneath it, and click **'Next' > 'Finish'**.

8. Switch to the **'Datasheet View'**, and use the drop-down list to select **'20142015'** for **'FLD_SESSION'** and '**1**' for **'FLD_SEMESTER'**. Limited lists are useful for these two attributes since the values list is almost never updated.

TIP: If you ever need to add to a limited list, you do not need to rerun "Lookup Wizard". For the attribute whose values you want to add to, click on the attribute name in 'Design View' and click the 'Lookup' tab in the 'Field Properties'. Values can be added under the 'Row Source' field. Each new value that you add needs to be separated by a semi-colon (;).

F. Creating a Composite Key

1. A composite key is a primary key field that is made up of more than one attribute. Composite keys are useful when you want to ensure that a combination of attribute values can only occur once in a table. For example, if we want to ensure that a student can only register for a particular course code only once during a semester and session.

Exercise 9: Edit the table **'TBL_REGISTATION_A123456', and** add the following attributes and data types:

Field Name	Data type
FLD_MATRIC	"Lookup Wizard" referencing **'TBL_STUDENT_A123456.FLD_MATRIC'**
FLD_COURSE_CODE	"Lookup Wizard" referencing **'TBL_COURSES_A123456.FLD_COURSE_CODE'**
FLD_REG_DATE	"Date/Time" or "Date", Format = 'Short Date'

2. In the **'Design View'** of table **'TBL_REGISTRATION_A123456'**, click and highlight the attributes **'FLD_SESSION'**, **'FLD_SEMESTER'**, **'FLD_MATRIC'**, and **'FLD_COURSE_CODE'** and click the **'Primary Key'** icon on the **'HOME'** toolbar. A key icon will appear on all four attributes.

EXERCISE 10: Switch to the **'Datasheet view'** of the table **'TBL_REGISTRATION_A123456'**, and type in the data as follows, and close the **'TBL_ REGISTRATION _A123456'** table.

FLD_SESSION	FLD_SEMESTER	FLD_MATRIC	FLD_COURSE_CODE	FLD_REG_DATE
20132014	1	A101	TU2983	18/09/2013
20142015	1	A101	TU2983	13/09/2014
20142015	1	A102	TU2983	13/09/2014
20142015	1	A105	TU2983	15/09/2014
20132014	2	A102	TK1914	14/03/2013
20132014	2	A107	TT1964	19/03/2013

3. Using the composite key, the student **'Ahmad'** (**A101**) can register for **TU2983** in two different semester sessions, but he cannot register for the same course twice in the same semester session.

G. Further Enforcing Data Integrity with Cascade Delete

1. Sometimes, we would like to ensure that if record is deleted in a referenced table (a.k.a. source table), then all records that reference it, which are determined to be useless, are also removed. This is called **'Cascade Delete'**.

2. Close all open tables an open the **'Relationship'** view.

3. In the **'DATABASE TOOLS'** toolbar, click **'Show Table'**.

4. In the **'Show Table'** list, highlight **'TBL_REGISTRATION_A123456'**. Click **'Add'** and click **'Close'**. The structure for **'TBL_ REGISTRATION _A123456'** will appear in the **'Relationship'** view.

5. Click on the relationship between **'TBL_REGISTRATION_A123456'** and **'TBL_STUDENT_A123456'**, and click **'Edit Relationship'**.

6. In the **'Edit Relationships'** window, tick **'Enforce Referential Integrity'**, tick **'Cascade Update Related Fields'**, and tick **'Cascade Delete Related Records'**. Click 'OK.

7. Click on the relationship between **'TBL_REGISTRATION_A123456'** and **'TBL_COURSE_A123456'**, and click **'Edit Relationship'**.

8. In the **'Edit Relationships'** window, tick **'Enforce Referential Integrity'**, and tick **'Cascade Update Related Fields'** (DO NOT TICK **'Cascade Delete Related Records'**) . Click 'OK.

9. Close the **'Relationship'** view.

10. To demonstrate the effect of a cascade delete, let's say that student **'Ali'** (**A102**) has withdrawn from the Faculty. Under the PDPA 2010 act, the faculty is obliged to delete all of his student records and course registration records. In this case, it is useless to keep his records any further in the database.

11. Open the table **'TBL_STUDENT_A123456'** in **'Datasheet view'**.

12. Highlight the record that says **'A102'**, and press the **'Delete'** key on your keyboard. A warning will appear that says all related records will also be deleted, click **'Yes'**.

13. Open the table **'TBL_REGISTRATION_A123456'** in **'Datasheet view'**. You will see that all rows that reference **'A102'** are also deleted.

14. Close all open tables.

TIP: Cascade Delete must be used with great care, because it can potentially propagate unintended data deletion in an instant if used incorrectly. Therefore, it should only be used on the specific relationships that we are sure we would like to delete all useless related records, such as the example above. If it is used in all relationships, the effects could be far reaching and devastating. Furthermore, undoing and recovering from cascade deletion is difficult if not impossible.

For example, if Cascade Delete were used in all relationships, deleting a department in **'TBL_DEPARTMENT_A123456'** will also delete courses formally offered by **'TS'** and will also delete:

- all courses that have ever been managed by 'TS'.
- all registration records of students who have ever taken any of the courses managed by 'TS'.
- all students in the 'TS' department.

When in doubt, do not enable (un-tick) Cascade Delete from the relationship properties of all relationship.

PART 4: Creating Views

1. **Views** or **queries** are essentially prepared SQL SELECT statements, which are stored in a database, that can operated upon just like an actual table. Views are generally used for the following purposes:

   - To provide a database user with a custom table display of data that may be spread across multiple tables.
   - To filter data from tables using user specified conditions.
   - To allow the user to view aggregated data that are calculated using some of the database's aggregation functions.

2. To a limited degree, some views can be used for data entry (**SQL INSERT**), changing data (**SQL UPDATE**), and removing data (**SQL DELETE**) if and only if the view is created from only one table source. However, it is not advisable to use views for these purposes, since referential integrity violations from existing table relationships may occur.

3. Before creating views, all relationships between tables and the referential integrity settings for all relationships must be created first.

A. Creating a View to Combine Multi-Table Data with the Query Designer

1. Let us create a view that shows all student's matric number, each student's names, and each student's full department name.

2. Create a query using the **'Query Designer'**. In the main menu, click the **'CREATE'** tab, and click the **'Query Design'** icon in the **'CREATE'** toolbar. The **'Query Design'** view will open, and a **'Show Table'** window will appear.

3. In the **'Show Table'** window, click **'TBL_STUDENT_A123456'** > **'Add'** > **'TBL_DEPARTMENT_A123456'** > **'Add'** > **'Close'**.

4. Each time you click **'Add'**, the structure of the table you selected will appear in the **'Query Setup'** view. If you add more than one table, like we just did, the relationship between the tables will also appear. A view can be created successfully if and only if all selected tables have relationships that link each other.

5. From the **'TBL_STUDENT_A123456'** table structure, click-and-drag **'FLD_MATRIC'** to the empty **'Field:'** below the **'Query Setup'** window.

6. Next, from the **'TBL_STUDENT_A123456'** table structure, click-and-drag **'FLD_NAME'** to the empty **'Field:'** to the right of the previous one.

7. Next, from the **'TBL_DEPARTMENT_A123456'** table structure, click-and-drag **'FLD_DEPT_NAME'** to the empty **'Field:'** to the right of the previous one.

8. Switch to the **'Datasheet View',** by clicking the . The results of the view you have just created is displayed in a table.

9. Click 'Save', and name this view **'QRY_STUDENT_DEPARTMENT'**.

B. Using Views as an SQL Generation Tool

1. When you are using the **'Query Designer'** to create a view, Microsoft Access is actually translating your attribute and table selections as **SQL SELECT** code in the background. You can view the SQL code by switching to the SQL view.

2. Open the SQL View. To do this, click the small down-arrow under the **'View'** icon, and click **'SQL View'**. The SQL code for creating this view will appear.

3. The SQL code may appear much longer than you are used to for the following reasons:

   - Microsoft Access automatically fully **'qualifies'** all attribute names. In other words, all attribute names are preceded by their table names. For example, instead of just **'FLD_MATRIC'**, the fully qualified attribute name is **'TBL_STUDENT_A123456.FLD_MATRIC'**.

   - Microsoft Access uses **JOIN** operators (i.e. **JOIN...ON**) to connect related tables instead of using WHERE conditions.

4. This SQL code will be automatically re-written as you make changes to the view in the **'Query Designer'**.

5. In most cases, you can immediately copy the SQL code generated in the SQL view, and paste it directly in another database program, or in your Visual Basic programming code.

6. However, you need to check if the other database program or programming language can understand the SQL written by Microsoft Access, since some of the SQL operations used by Microsoft Access are not standard **SQL-92** operators, and some SQL operations are exclusive to Microsoft Access. For example:

   - A few database programs do not recognise **JOIN** operators or any of its variations.

   - Some database programs do not have the aggregation functions used by Microsoft Access, or have different aggregation function names.

   - Almost all database programs do not recognize the **PIVOT** operator.

C. Renaming Displayed Attributes using Aliases

1. Switch to the 'Design View' for the view **'QRY_STUDENT_DEPARTMENT'**.

2. In the first **'Field:'** column, place your cursor in beginning of **'FLD_MATRIC'**, and append by typing **'MATRIC:'** (without the quotes).

3. Switch to **'Datasheet View'**. You will see that the attribute **'FLD_MATRIC'** has be renamed **'MATRIC'** which is more readable by end-users. This renaming is not permanent and the original attribute name is not deleted, since it actually creates an 'Alias'.

4. Switch to the **'SQL View'**. You will see that Microsoft Access has added the **'AS'** operator to the SQL code. This feature is useful when you want to reduce lengthy field names, or to put a new name for aggregated fields. Any subsequent SQL operations can be done using these aliases instead of the actual attribute names.

EXERCISE 11: Place aliases so that the output of **'QRY_STUDENT_DEPARTMENT'** has the following structure:

MATRIC	STUDENT	DEPARTMENT

D. Filtering Data in a View with the Query Designer

1. Let's say we want the view to only show students from the **'Information Systems'** department

2. Switch to the 'Design View' for the view **'QRY_STUDENT_DEPARTMENT'**.

3. In the **'Criteria:**' row under the **'FLD_DEPT_NAME'** column, type "**Information Systems**" (Microsoft Access will automatically add the quote marks).

4. Switch to **'Datasheet View'**. You will see that the output of the view has been filtered to only show Information Systems' students.

5. Switch to **'SQL view'**. You will see that Microsoft Access has added a **WHERE** condition to filter the displayed data.

6. Additional conditions can be added to the view. An **OR** condition is made when conditions are added to the **same column** of an attribute, but in a **different row** under **'Criteria:'**. In the Query Designer, all rows under the **'Criteria:'** row are **OR** conditions.

7. Switch to **'Design view'**. In the empty cell under the same column as the **'Information System'** condition, type "**Computer Science**".

8. Switching to **'Datasheet view'** shows the **OR** condition output. Switching to **'SQL view'** shows the **OR** condition added to the SQL code.

9. An **AND** condition is made when conditions are added to the **same row** or an existing condition, but in a **different attribute column**. This way, you can make a complex filter by just adding conditions to different rows and columns. By default, Microsoft Access will process conditions row-by-row, i.e. it will process **AND** conditions first, before processing **OR** conditions.

10. Switch to **'Design view'**. In the empty **'Criteria:**' cell under **'FLD_NAME'**, which is the same row as the "**Information Systems**" condition, type **Like "A*"**.

11. Switching to **'Datasheet view'** shows the output of the combination of the **AND/OR** conditions. Switching to **'SQL view'** shows the precedence of **AND/OR** conditions in the WHERE operator.

12. Click **'Save'** and close **'qry_student_department'**.

E. Creating a Query-on-a-Query with the Query Designer

1. Views can also be made against other views that have been stored in database, and not just tables.

2. Create a new view with the **'Query Designer'**. When the **'Show Table'** window appears, click the **'Queries'** tab.

3. Click **'QRY_STUDENT_DEPARTMENT'** > '**Add**' > **'Close'**. The view will appear in the **'Query Setup'** window just like a table, and you can work on it as if it were an actual table.

> TIP: If a view like **'QRY_STUDENT_DEPARTMENT'** uses aliases, the query will be made against the alias directly, and not against the real attribute name of the source tables. Because of this, the relationships between this view an all other tables or views might not show up in the **'Query Setup'** window, because the related attribute names are now different. When this occurs, you will need to manually add a **WHERE** condition in the SQL code to indicate how views and tables are related to each other.

4. Click-and-drag **'DEPARTMENT'** and release click in the first empty **'Field:**' column.

5. Click-and-drag 'STUDENT' and release click in the next empty **'Field:'** column.

6. Switching to **'Design view'** shows the view's output. Switching to **'SQL view'** shows a more simplified SQL code that no longer needs to reference the original tables. In actual fact, this **'query-on-a-query'** view first process the view **'QRY_STUDENT_DEPARTMENT'** before processing its own SQL and conditions.

7. Click **'Save'** and name this view **'QRY_STUDENT_COUNT'**.

F. Aggregating Data in a View with the Query Designer

1. Switch to the **'Design View'** of the view **'QRY_STUDENT_COUNT'**.

2. Click the **'Totals'** icon, on the **'DESIGN'** toolbar, a new row called '**Total:**' will appear in the **'Query Setup'** window. This 'Total:' row provides the users with aggregation functions such as **Sum**, **Max**, **Count**, and others.

3. In the **'Total:**' row under the **'STUDENT'** column, click the drop-down arrow and change the aggregation function from **'Group By'** to '**Count**'.

4. In the **'Field:**' row under the '**STUDENT**' column, place your cursor in front of **'STUDENT'** and type **'NUMBER_OF_STUDENTS:'** (without the quotes).

5. Switching to **'Datasheet view'** shows the result of the aggregation function **'Count'**. Switching to **'SQL view'** shows that the necessary SQL-code or group non-aggregated attributes are automatically generated.

6. Click **'Save'** and close the view **'QRY_STUDENT_COUNT'**.

G. Using the SQL View of the Query Designer to type in SQL Code

1. You do not need to use Query Designer to generate views. You can use the **'SQL View'** of the Query Designer to manually create views using SQL SELECT.

2. The **'SQL View'** can be used to run any SQL operation just like the command line SQL processor in enterprise level databases such as Oracle, IBM DB2, or SQL Server. The **'SQL View'** is a great tool for you to practice writing your SQL code and testing the output of your SQL code, before using and exporting the SQL code in your programming language code.

3. Opening the SQL View directly. In the main menu, click the **'CREATE'** tab > **'Query Design'**. When the **'Show Table'** window opens, click **'Close'**. Click **'SQL view'** in the toolbar and the **'SQL View'** display will open.

EXERCISE 12: Type in the following SQL code and save this view as **'QRY_ABU_COURSES'**.

```
SELECT FLD_COURSE_CODE
FROM TBL_REGISTRATION_A123456
WHERE FLD_MATRIC='A107';
```

EXERCISE 13: Type in the following SQL code and save this view as **'QRY_INSERT_STUDENT'**. Click the 'Run' icon to execute the SQL Code. Click 'Yes' at the warning.

```
INSERT INTO TBL_STUDENTS_A123456
VALUES ("A109", "Fatimah", "TR");
```

EXERCISE 14: Type in the following SQL code and save this view as **'QRY_UPDATE_DEPT'**. Click the 'Run' icon to execute the SQL Code. Click 'Yes' at the warning.

```
UPDATE TBL_STUDENTS_A123456
SET FLD_DEPT="TK"
WHERE FLD_NAME="Fatimah";
```

EXERCISE 15: Type in the following SQL code and save this view as **'QRY_DELETE_COURSE'**. Click the 'Run' icon to execute the SQL Code. Click 'Yes' at the warning.

```
DELETE *
FROM TBL_COURSES_A123456
WHERE FLD_COURSE_NAME="Mathematics";
```

Part 5: Before Leaving the Lab

1. Copy your entire working folder, e.g **'c:\a123456_facultyrecords'**, to your USB drive. This folder will be used again for the next lab session.

2. Delete the existing **'c:\a123456_facultyrecords'** folder in your lab computer's **'C:\'** drive.

Additional Lab Exercises

1. Use your own interpretation on how to do the following tasks. Create your own data and attributes as needed and select a suitable data type for the task at hand.

2. Create a new table for LECTURERS that contains a list of 3 lecturers. Each lecturer works in one of the existing DEPARTMENTS.

3. Create a new table for COLLEGES that contains a list of 2 colleges and its street address.

4. Create a new table for ROOMS. Each room is located one of the existing COLLEGES.

5. Create an intermediate table that lists the students assigned to each room. Each room can house one or more existing STUDENTS. A student can reside in only one room.

6. Modify your existing STUDENT table to include an 'Advisor' attribute. The student's advisor is one of your existing LECTURERS.

7. In your Relationship View, show the relationships between all new tables, along with the cardinalities of each new relationship.

8. For each new relationship, determine if you need the new relationships to have a CASCADE UPDATE or CASCADE DELETE with related tables.

9. Create a query to show how many students are being advised by each lecturer.

10. Create a query that lists the Matric Number and Name of each student, grouped by the names of their advisor and sorted alphabetically.

11. Create a query that shows the student's name and residence details (college name, room and street address).

12. Add a 'Monthly_Rent' attribute to your ROOMS table. If the monthly rent for every room is the same (e.g. RM200 or $200) and the monthly rent is divided equally among the residents of each room, create a query that lists the name of every student and the amount of money (in RM or $) that each student has to pay for their portion of the rent.