Introduction to LibreOffice Base (LOB)-6

Queries using multiple tables

Robin Beaumont

Thursday, 21 July 2011 e-mail: robin@organplayers.co.uk

Contents

1.	LEA	RNING OUTCOMES	2
2.	INT	RODUCTION	2
3.	THE	E DATABASE	3
4.	CRE	EATING A COPY OF YOUR DATABASE	4
5.	VIE	WING AND USING SQL	5
6.	WH	IY QUERY ON MORE THAN ONE TABLE?	6
7.	TW	O TABLE QUERIES	6
8.	THE	REE TABLE QUERIES	7
	8.1	Joins	.10
	8.2	Cross products - to be avoided	.11
9.	EXT	TENDED EXERCISE - LOGBOOK	. 11
10	. MIC	CROSOFTS ACTION QUERIES AND EQUIVALENTS IN BASE/SQL	. 13
	10.1	EQUIVALENT IN BASE	.14
	10.2	USING A SELECT QUERY TO TEST AN UPDATE QUERY	.14
	10.3	UPDATING SPECIFIC RECORDS	.14
	10.4	Append Queries	.15
	10.5	DELETE QUERIES	.17
11	. sur	MMARY	. 17
12	. ДРГ	PENDIX A USES OF THE CROSS JOIN	. 18

This chapter should be studied alongside the database theory chapter:

Introduction to the theory behind QBE & SQL.

available from www.robin-beaumont.co.uk/virtualclassroom/contents.html

1. Learning outcomes

This practical chapter aims to provide you with the following skills and knowledge to achieve the learning outcomes listed below. After you have completed this practical chapter you should come back to these points ticking off those with which you feel happy

Learning outcome	Tick box
Rationale for carrying out a query on more than one table	
Be able to design multi-table queries	
Be able to edit links in the QBE window	
Be aware of the different types of join	
Discuss a possible draft model for a log book	
Understand the concept of Action queries	
Be aware of the dangers, and the necessary actions to take to minimise them, when executing Action queries	
Know that it is possible to preview, within limitations, the results of Action queries	
Be able to make backups of databases	
Be aware of the SQL view	
Be aware of the ability to be able to print query results and definitions	

2. Introduction

So far in the practical chapters concerning queries we have extracted data from a single table. In contrast in this chapter, we will now develop queries extracting data from more than one table.

We will also take a look at other types of queries (called **Action** queries in Access), those that change the table itself rather than extracting data from it. To investigate this aspect we need to look at SQL directly rather than the QBE grid.

But before we start investigating these topics we need to consider three preliminaries to set the scene:

- The database we plan to use
- How to make backups of the database
- Viewing and using SQL

3. The database

The database used in this chapter is called cons4.obd and contains the following data amongst which there are additional patient records as shown below.

	doctor													
Doc id	doc first name	doc Surname	Gender	Date reg	Addrs title	Addrs st name	Addres st no	city	postcodeA	postcodeB	Phone No			
1	john	SMITH	2	01/04/1967	The old surgery	12 station rd	Walkergate	Newcastle	Ne23	4rr	0191 009282			
2	steve	Jarvis	2	01/05/1970	Morpeth HC	High st	Morpeth		Mo2	1ts	0127 485739			
3	Mary	Goodall	1	27/11/1967	Seeham HC	The harbour	Seaham		Se1	4jp	0191 345867			
4	Fiona	black	1	30/01/1976	Cragside HC	The East Wing	Cragside castle		Cr3	1ws	0146 389431			
5	Anna	Scriabin	1	25/03/1961	Uniiverisity MC	Newcastle Univeristy		Newcastle	Ne1	1aa	0191 445776			
23	mark	Goodall	2	27/02/1955	Seaham HC	The harbour	Seaham		Se1	4jp	0191 345867			

	patient													
patient id	Title	first name	Surname	Doc id	DOB	Gender	Date on list	No children	Addrs title	Addrs st name	Addres st no	city	postcodeA	postcodeB
1	mr	john	smith	23	19/02/45	2	01/02/85	3	Cherry bud	Orchard Lane	45	newcastle	NE5	2pn
2	miss	shella	jones	23	02/01/55	1	01/02/70		Cathedral view	Cathedral Row	11	durham	Du2	1b
3	prof	richard	farmer	23	06/12/55	2	01/02/89	2		Tosson tce	134	newcastle	Ne2	3no
4	mr	John	Hewitt	1	23/01/71	2	01/02/96	1		Dovetail Gardens	9	Stanley	St1	5th
5	miss	shirley	anderson	1	15/01/60	1	01/02/93	3		Newton Rd	23	newcastle	NE2	3jl
6	mr	chris	bull	5	01/04/78	2	01/04/73	2	the ashes	broad ave.	12	newcastle	ne5	1al
7	mrs	ann	cookson	5	01/06/35	1	01/11/70	4	beautitude	the grove	20	newcastle	ne3	8yk
8	mr	Gordon	brown	4	27/10/50	2	23/10/80	2		the Larches	44	newcastle	NE10	1pp
9	mr	alan	bull	3	03/05/80	2	10/10/96	1		Valium court	128	newcastle	ne5	1pr
10	ms	christina	verdi	2	25/01/65	1	23/10/73	3		Roseworth crescent	78	newcastle	ne1	3ww
11	miss	karen	wedge	3	12/12/50	1	29/11/77	1		Rectory rd	12	newcastle	ne2	4ta
12	miss	sarah	Williams	4	05/08/65	1	10/11/92	2	comfort	Windsor tce.	57	newcastle	ne2	5уу

			episode			
episode id	Patient ID	Doc id	Date seen	urgency	systolic	diastolic
1	5	1	01/01/1995	Yes	340	130
2	5	1	05/01/1995	No	235	135
3	5	1	11/01/1995	No	180	100
4	5	1	17/01/1995	No	170	95
5	5	1	28/01/1995	No	175	95
6	5	1	10/02/1995	No	170	100
7	5	1	27/02/1995	No	170	95
8	5	1	20/03/1995	No	180	90
9	4	1	03/02/1996	No	145	80
10	4	1	23/02/1996	No	150	85
11	1	23	23/04/1994	Yes	130	70
12	1	1	06/05/1994	Yes		
13	2	2	13/04/1973	Yes		
14	3	3	02/06/1990	Yes	190	90
15	4	2	03/03/1996	No	165	85

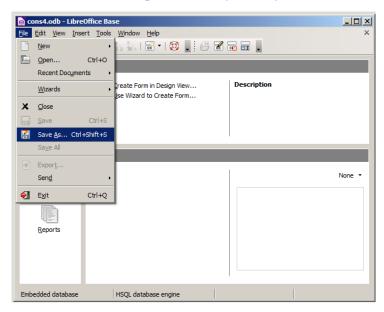
There are also two other tables containing past Doctor records:

 $\mbox{\bf doctor_old}\,$ - where the doc ID field has the same values as in the doctor table

and

doctor_old_good_doc_ids - where the doc ID field has different values to those in the doctor table

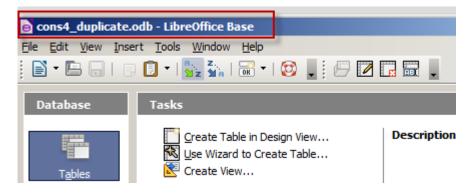
4. Creating a copy of your database



For several of the exercises in this chapter you need to create a temporary copy of your database either from within Libreoffice Base (LOB), or by just using the file copy process when in Windows explorer. From within Libreoffice Base (LOB), select the **File** menu option then the **save as** option (shown opposite).

In the Save As dialog box select the folder then type in the name you want to give the duplicate database, I have called it cons4_duplicate.

Libreoffice Base (LOB), than automatically opens this new copy. To check to see which database you have open look at the top of the database window, obviously you can just close the database you have open at any time and re open the original one to continue to work on it if you wish.



Exercise 1. creating a copy of the database

Please create **two** copies of the cons4.odb database. You are free to give then whatever names you think are appropriate.

5. Viewing and using SQL

This chapter should be studied alongside the database theory chapter *Introduction to the theory behind QBE* & *SQL*.

Although so far we have concentrated on the QBE grid as a method of specifying our queries, there are unfortunately many types of query that you can't develop using it, instead you need to use SQL (the grown up version of QBE) to specify the query.

There are basically two type of query, for reasons that will become clear, the first type of query is where you only extract data from one or more tables rather than change them and is called a SELECT query.

The other type of query (called action queries in Access) allows you to change the tables in some way such as; create, update, append or delete.

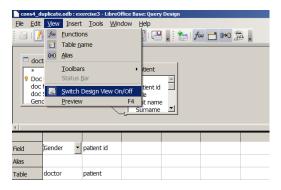
In both Access and Libreoffice Base you have the opportunity to move between the QBE grid and the SQL equivalent when developing SELECT queries. However In Access you have the additional possibility of being able to develop queries that actually change the tables in the database in the QBE window along with the opportunity to view the equivalent SQL, a luxury not available in Libreoffice Base. In Base you develop these so called action queries directly in SQL using a special SQL window described latter.

Important note - difference between Access and Base:

Base only allows you to view the SQL equivalent of SELECT gueries in the sql view from the qbe grid.

For developing other types of query in Base you must use the Tools ->SQL menu option to bring up the Execute SQL statement window

I have given you hints of what SQL looks like as we have worked through the material by occasionally showing the SQL equivalent to the QBE. If you intend to develop databases of any complexity you must learn SQL and fortunately Access and Base eases the process by providing a SQL translation of any QBE you develop (but not always in the other direction). When beginning to learn SQL it is a good idea to start by going through all your QBE queries and looking at the SQL equivalents. You will soon see how succinct the SQL is compared to the QBE.



You can always view the SQL equivalent to the QBE query in the query design window by choosing the menu option view -> Switch Design View On/Off (shown opposite).

In this chapter I have presented the SQL to the various exercises in a series of boxes I do not expect you to understand them fully but some readers might find them interesting and useful and those of you who prefer English to the graphical approach might prefer the SQL?

There are many flavours of SQL and although there is a standard every software developer offers their own extensions/limitations therefore I show the Access and Base equivalents in this chapter.

SQL some sql

For reference only: You can also make both Access and Base interact with specific varieties of SQL by using in Access a pass-though query or in Base by setting the database to be linked to a specific SQL engine and then use the "Run SQL command directly in the SQL

window.

6. Why query on more than one table?

Suppose we are interested in the total number of patients that the female and male partners have. The doctor table contains a gender field and the patient table provides a foreign key to the doctor table. We could add another field to the patients table specifying the doctors gender however this is not desirable for several reasons, we would have duplicated data and we are mixing data between tables. What we need is some way of linking the two. This is achieved in queries by making use of either pre-existing relationships of adding new ones.

By linking tables we can query records in the child table by any field in the parent table. Considering the doctor patient tables we could look at patients by city, post code, etc. in fact any field that is contained in the doctor table.

Exercise 2. Sensible fields to search on in the doctors table

List which of the fields in the doctors table it would be sensible to search the patient records on.

7. Two table queries

Consider the following question, How many patients are registered with female and male doctors?

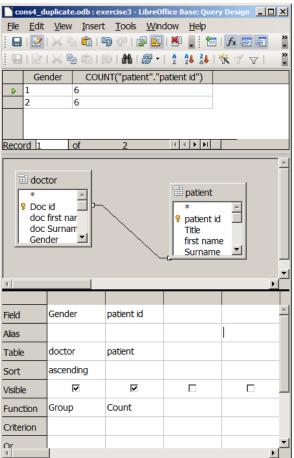
To find this out we need to consider the information in both the DOCTOR and PATIENT tables, the exercise below takes you through the steps to carry out the required query.

Exercise 3. A two table query between the DOCTOR and PATIENT tables

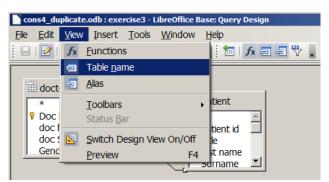
Open the cons4 database

Open up the Task Create Query in Design View to specify a new query

Select the doctor table and then the patient table to add to the top part of the query window as shown below.

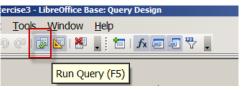


When you are working with more than one table in a query it is a good idea to add the optional additional row specifying the table name to the QBE grid. If the table name row is not already shown in the grid show it by clicking on the "menu option view -> Table name (shown below).



Set the grid up as shown opposite.

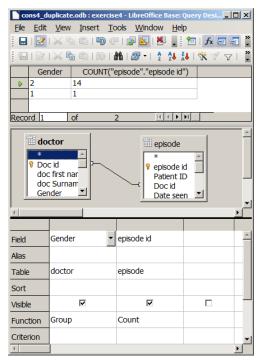
Run the query by clicking on the tick box icon or pressing F5 (shown below).



Remember we gave the doctors the following coding for gender; 1-female:2-male so

1=female;2=male so we have:

Equal numbers of patients are registered with male and female doctors.



Exercise 4. A two table query between the DOCTOR and EPISODE tables

Suppose now that we want to know how many of the consultations (episodes) are with female or male doctors. It would seem logical to use the doctor and episode tables for this task. Setting up the QBE grid this time as shown opposite we obtain the result that:

All but one of the consultations are with male doctors.

So although we have a equal number of patients registered with male and female doctors all but one of the consultations (episodes) are with male doctors!

Does this mean that the female doctors are very successful at preventative medicine, or do they not bother to record their consultations etc. or are the male doctors seeing patients registered with female doctors. Although the first two possibilities can't be answered by examining the current data it is possible to investigate the third possibility. The following exercise shows you how to do this.

"Question 1: For those patients who have a recorded episode. How many of them are registered with a male doctor."

"Question 2: For those patients who have a recorded episode. How many of the consultations (episodes) are with their own doctor."

The SQL versions of the 2 above queries

Exercise 3 Doctor Patient:

SELECT "doctor". "Gender", COUNT("patient". "patient id") FROM "patient", "doctor"

WHERE "patient"."Doc id" = "doctor"."Doc id"

GROUP BY "doctor". "Gender"

ORDER BY "doctor". "Gender" ASC

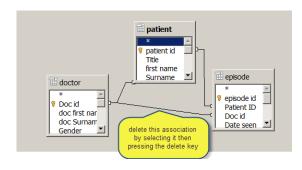
Exercise 4 Doctor Episode

SELECT "doctor". "Gender", COUNT("episode". "episode id") FROM "episode", "doctor" WHERE "episode". "Doc id" = "doctor". "Doc id"

GROUP BY "doctor". "Gender"

Three table queries 8.

Both of the above questions need the introduction of the Patient table as this will provide information about patient registration meaning that we will now have three tables in the query. We will look at each of the above questions in turn now

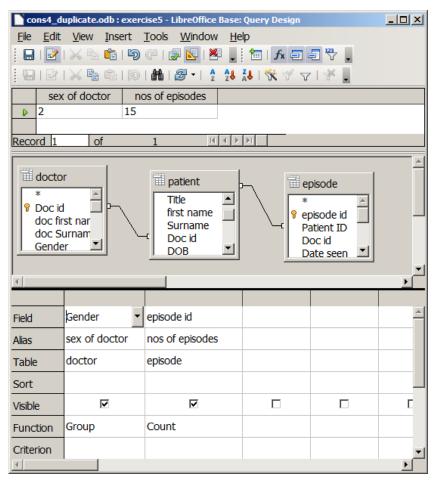


The guery in the exercise **below** answers the first guestion posed. It is the most complex query we have considered so far involving three tables. Notice that the relationship between episode.Doc id and Doctor.doc id (notice that I'm using the convention tablename.fieldname now) has been removed (see opposite. It is important to realise that while we are making this modification to the structure in the guery design window we are not effecting the relationships we have specified in the relationships window.

Exercise 5. What is the gender of the registered doctor of those patients with consultations?

Add the three tables and edit the QBE grid accordingly as shown below.

The final result should look like the picture below. To remove a relationship line In the query design window highlight the relationship line by clicking on it and then choose the menu option edit -> delete. Specifically make sure you delete the episode.doc id to doctor.doc id relationship (shown on previous page).



You can also specify the column heading in the results by editing the **alias row** property.

What does this mean? The nearest I can get in words to it is; grouping the doctors by gender, count the number of episodes.

The result of the above query is shown opposite, with the edited column captions. so we have:

All the patients that have recorded consultations (episodes) are registered with male doctors.

If you don't believe the results check the original tables.

Just out of interest this is the SQL that the above query produces:

In Access (provided for comparison):

SELECT doctor.Gender,

Count(episode.[episode id]) AS [nos of episodes]

FROM (doctor

INNER JOIN patient ON doctor.[Doc id] = patient.[Doc id])

INNER JOIN episode ON patient.[patient id] = episode.[Patient ID]

GROUP BY doctor.Gender;

In Libreoffice Base:

SELECT "doctor". "Gender" AS "sex of doctor",

COUNT("episode". "episode id") AS "nos of episodes"

FROM "episode", "patient", "doctor"

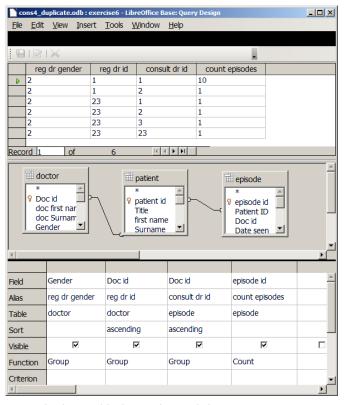
WHERE "episode". "Patient ID" = "patient". "patient id"

AND "patient". "Doc id" = "doctor". "Doc id"

GROUP BY "doctor". "Gender"

Exercise 6. For those patients who have a recorded episode. How many of the consultations (episodes) are with their own doctor."

This time we request additional group by commands setting up the QBE grid as shown below, notice that I have also specified sort ascending for three of the output fields.



three tab	three tables patients seen by own/other doctor										
reg doc gender	reg doc id	consult dr id	count episodes								
2	1	1	10								
2	1	2	1								
2	23	1	1								
2	23	2	1								
2	23	3	1								
2	23	23	1								

The result opposite/above shows that out of the 15 consultations (episodes) 11 are with the registered doctor

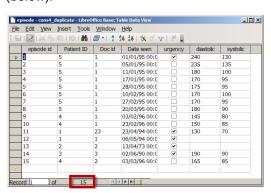
We could also add another group by column to find for each patient the number of consultations with each of the doctors.

Ideally we would like just two values that for the total number of consultations with the registered doctor and those with another one. While this is possible with a sub query not discussed here I will offer a work around which is remarkably simple.

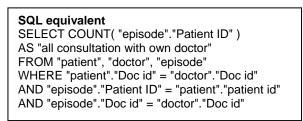
If we just use the original associations between the three tables and then request a Count from any field in the episode table we are given the total number of

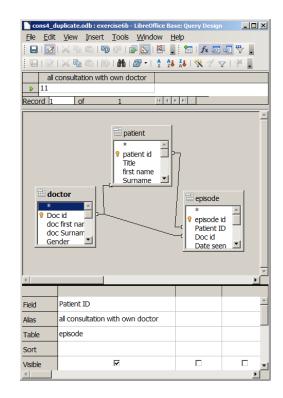
consultations with the registered doctor:

We know from the episode table the total number of episodes just by opening the table and looking at the bottom of the window (below).



So the percentage of recorded consultations with patients who see their registered doctor is 11/15 = 73.3%

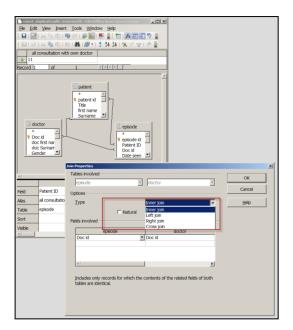


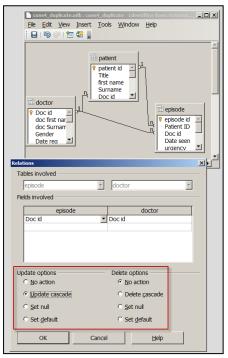


8.1 Joins

You may have been wondering what has been going on in the previous examples. To answer this question you will need to understand what joins are, in a previous chapter we discussed relationships and joins are just another type of relationship.

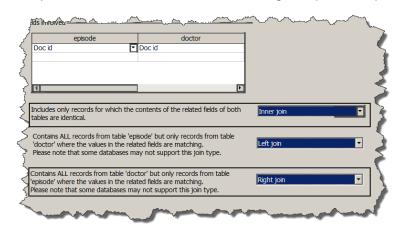
Although the lines between the tables in the above QBE window appear very similar to those we created in the practical chapter on Relationships they actually are different. Below are screenshots of both the query window (left) and the relationships window (right) showing the options for the "doctor". "doc id" and "episode". "doc id"





Notice how you are presented with a different set of options if you are in the Query designer or relationships windows.

The pictures below shows how the text changes dependent upon which join type you select.



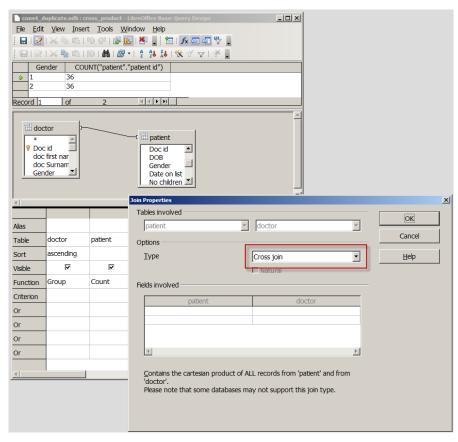
Libreoffice base does NOT indicate which of the join types is being used in the query design window by having a different line style for each.

In contrast in Access the standard relationship line indicates the default inner join (simply a plain line with no arrow or the 1:N) whereas a left/right outer join is indicated by a line with an arrow pointing towards the matching records only table. In contrast the end opposite the arrow indicates the table which provides all the records for the result.

Examples are provided of the results for each of these joins in the *Introduction to the theory behind QBE and SQL* chapter.

From the above it appears that you can have some of the records all of the time but not all the records any of the time. To achieve this you need to use the word union which unfortunately Libreoffice Base does not provide in QBE but only in SQL. In access you also have the opportunity to create various types of **action query** amongst which there is an **append query**. first I would like to give you a warning about a thing called Cross products.

8.2 Cross products - to be avoided

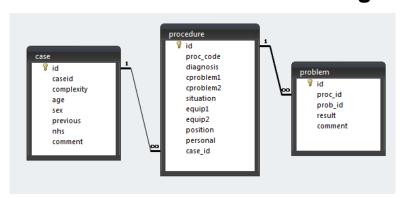


I just said in the above that you can't have all the records from both sides of a relationship in a query but this is not strictly true, what would be fairer to say it that most of the time you don't want them. You can create a type of join called a Cross or Cartesian join (also known as a **product**) where every record of the left table is combined with every record in the right-hand table. Taking our first example of the number of patients registered to males and female doctors we could obtain a Cartesian join of the guery simply by changing the join type to Cross Join in the query design window, we know from the previous exercise that the correct number of records is 6 so a Cartesian product of this number is 6 x 6=36 which is the result opposite.

If you end up with a count which is more than the number of records in a table you are querying it is most likely because you have accidentally specified a Cross join.

Only advanced database managers use cross joins so I will say no more, but those of you who wish to see how they can be used look at appendix A.

9. Extended exercise - Logbook



Consider the three following tables that someone from a unknown specialty has set up to record their cases and procedures with which they have been involved. Also recorded are any associated problems they may have encountered. Basically a simple electronic log book.

Table: Case

Field Name	Field type	Field size	Description
id	autonumber	Long integer	Internal case id- Primary key
case ID*	Text	50	Hospital case ID
complexity	Number	Integer	complexity; 1=simple, 2=some co-morbidity; 3= co-morbidity which may affect outcome, 4=nasty!
age	Number	Long Integer	Years old
sex	Number	Long Integer	1=male, 2 female,
previous	Number	Long Integer	Previous procedure(s) carried out, 0=no, 1=once, 2= twice etc.
nhs	Number	Long Integer	NHS case=1
comment	Text	250	Text comment

Table: Procedure

Field Name	Field type	Field size	Description
id	autonumber	Long Int	Internal procedure ID - primary key of table
proc_id	Text	50	procedure code obtained from a list of codes possibly opcs4 (characters + numbers)
diagnosis	Text	50	Read code
cproblem1	Text	250	Text
cproblem2	Text	250	Text
situation	Number	Long int	Circumstances (1=routine planned, 2=emergency in standard environment, 3=Dire emergency not in usual environment)
eqiup1	Number	Long int	Equipment code used (from list)
eqiup1	Number	Long int	Equipment code used (from list)
position	Number	Long int	Non-participant observer=1; Participant observer=2; Assistant=3; Main performer with suppervision=4; Main performer with no supervision=5
personal	Number	Long int	Satisfaction score with procedure (1 to 10)
case id	Number	Long Int	case ID - Foreign key (for one to many relationship)

Table: Problem

Field Name	Field type	Field size	Description Internal ID Primary key of table					
id	autoumber	Long Int						
proc_id	Number	Long Int	Procedure ID - Foreign key (for one to many relationship)					
prob_id	Number	Long Int	Problem code from list (e.g. 20=inadequate supervision, 21=unrealistic expectations from supervisor)					
result	Number	Long Int	1=resolved satisfactorily with no adverse effect on case. 2=Resolved possibly effected patients outcome. 3=Resolved definitely effected patients outcome. 4=Not resolved and effected patients outcome					
comment	Text	250	Comment					

Create the tables and relationships, and also enter some data. Now consider the following questions:

- Which fields do you feel are superfluous?
- What additional fields, or tables would you want to add?
- Do you feel that any of the fields are in the wrong table give your reasons?
- List the queries you would possibly want to do on INDIVIDUAL tables in the above description?
- List the queries you would possibly want to do on SEVERAL of the above tables?
- Is the Relationship diagram suitable for your use? How might you modify it?
- If certain fields were to be exported to some central database, say your postgraduate institute or the College which fields would you be unhappy about exporting, and which fields would be essential for them?

10. Microsofts Action queries and equivalents in Base/SQL

Action queries allow you to carry out the following actions on tables:



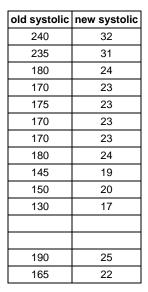
Update queries change (update) one or more fields and /or records whereas Append queries append records to another table rather than writing a set of results to a virtual table. Delete queries remove records while maketable queries actually allow you to create tables from scratch rather than using the table design window we used in the first access practical chapter. Microsoft Access indicates action queries in the Database window by displaying an exclamation point (!) before the name.

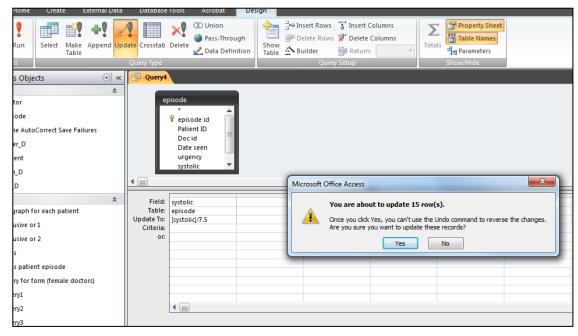
The table below gives some examples (in Access) of possible updating you might want to do.

Expression	Result				
not recorded	Changes values to 'not recorded'				
02/10/94	Changes date values to 02/10/94				
Yes	Changes No values in a Boolean field to Yes				
[first name] & [surname]	Concatenates first name to the beginning of each surname				
[Systolic] - [diastolic]	calculates pulse pressure				

Microsoft Access Example:

Suppose the EEC has decided that all BPs (blood pressure recordings) should now be recorded in Kpascals instead of mmHg. All the old records would need updating in the episodes table (1 kPa = 7.5 mmHg). This is achieved by creating an update query with the episodes table and the following QBE grid setup. When you click on the run exclamation icon you get the warning message shown below, click yes to continue:





SQL equivalent

The SQL equivalent to the above couldn't be simpler (and possibly deadlier!) in Microsoft Access:

UPDATE episode

SET episode.systolic = [systolic]/7.5;

10.1 Equivalent in Base

This is more problematic from the user interface perspective than in Access but let someone else explain, as on the Openoffice online forum someone asked how to run a UPDATE query because it wouldn't run in the SQL window in the query designer:

Now, a Query is used to READ a database (not to change it) and return a SELECT (yes, pun intended) subset of data from the database.

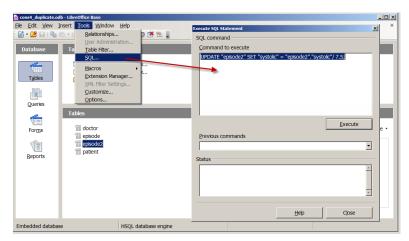
Therefore, after you OPEN a database, you can use the icon on the left . . . Queries icon to READ data in your database.

But . . . what you are attempting to do with the **UPDATE** statement is to **CHANGE** you database. Some of the SQL statements that **CHANGE** a database (either the database definition OR the data) include:

- 1. Create when you want to Create a table
- 2. Alter when you want to modify a table definition
- 3. **Drop** when you want to remove a table
- 4. Insert when you want to add data to a table
- 5. Update when you want to modify already existing data in a table
- 6. **Delete** when you want to remove data in a table

end of quote taken from http://www.oooforum.org/forum/viewtopic.phtml?t=97361 posted by Sliderule 2004

The Solution is simple such SQL commands are run from a special SQL window provided from the tools menu, shown below:



Typing the Base SQL equivalent to that obtained from Access on the previous page:

UPDATE "episode" SET "systolic" = "episode". "systolic" / 7.5;

Note that this is almost identical to that produced by Access on the previous page but with quote marks and the table name.

You may think that it would be better to create a new field in the original table with the new values rather than deleting the original values and I would agree because running a update query can have disastrous consequences. Unfortunately you can't create new fields for the actual table in the QBE grid (in either Base or Access), however you can

easily do so using SQL with a statement like the following in Base:

ALTER TABLE "episode" ADD COLUMN "diastolic kp" FLOAT

we can also rename a column:

ALTER TABLE "episode" ALTER COLUMN "diastolic_old" RENAME ""diastolic_new"

As a preliminary to running the potentially disastrous SQL we can test the proposed update query by creating an equivalent select query first.

10.2 Using a select query to test an update query

For example if you wanted to test a proposed update of the diastolic field first create a **select query** adding the table then in a empty column in the QBE grid add the expression: "episode"."diastolic"/7.5. and in the Alias cell giving the new column the appropriate name such as **new_diastolic**

I personally nearly always create these test select queries first to test any changes I plan to make to the original tables.

10.3 Updating specific records

The cell below the 'update to' cell allows you to enter criteria so that only records meeting the criteria are updated. As you have spent some time working with developing criteria for select queries we will not tread this ground again.

10.4 Append Queries

You might want to also Insert (Access calls this append) records

As you can imagine these are again dangerous queries. The most important thing to do before running such a query is to ALWAYS MAKE A BACKUP FIRST (either copying the entire database or copying the individual tables that might be updated) and then try out the query as a select query first.

Exercise 7. Append and delete queries

I have added to the cons4 database a table consisting of previous doctors called doctor_old

	doctor_old												
Doc id	doc first name	doc Surname	Gender	Date reg	Addrs title	Addrs st name	Addres st no	city	postcodeA	postcodeB	Phone No		
1	adam	ketalby	1	01/06/1940	The old surgery	12 station rd	Walkergate	Newcastle	Ne23	4rr	0191 009282		
2	Claire	bach	2	01/03/1946	Morpeth HC	High st	Morpeth		Mo2	1ts	0127 485739		

Suppose we wish to add these records to the doctors table.

In Access we would run an Append query using the QBE grid which would produce the following SQL:

Access SQL:

INSERT INTO doctor ([Doc id], [doc first name], [doc Surname], Gender, [Date reg], [Addrs title], [Addrs st name], [Addres st no], city, postcodeA, postcodeB, [Phone No])

SELECT doctor_old.[Doc id], doctor_old.[doc first name], doctor_old.[doc Surname], doctor_old.Gender, doctor_old.[Date reg], doctor_old.[Addrs title], doctor_old.[Addrs st name], doctor_old.[Addres st no], doctor_old.city, doctor_old.postcodeA, doctor_old.postcodeB, doctor_old.[Phone No]

FROM doctor old:

By using the find & replace function in any word processor and also adding the double quotes (") for all the field table names we now have:

INSERT INTO "doctor" ("Doc id", "doc first name", "doc Surname", "Gender", "Date reg", "Addrs title", "Addrs st name", "Addres st no", "city", "postcodeA", "postcodeB", "Phone No")

SELECT "doctor_old"."Doc id", "doctor_old"."doc first name", "doctor_old"."doc Surname", "doctor_old"."Gender",

"doctor_old"."Date reg", "doctor_old"."Addrs title",

"doctor_old"."Addrs st name", "doctor_old"."Addres st no",

"doctor_old"."city", "doctor_old"."postcodeA",

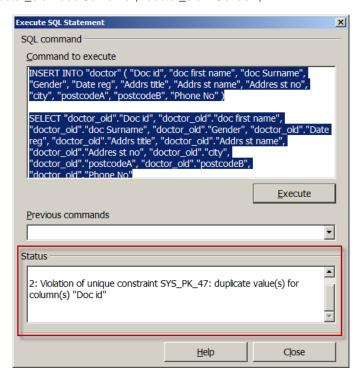
"doctor_old"."postcodeB", "doctor_old"."Phone No"

FROM "doctor_old";

Placing this statement in the Execute SQL statement box (as in previous example) we have the result shown opposite.

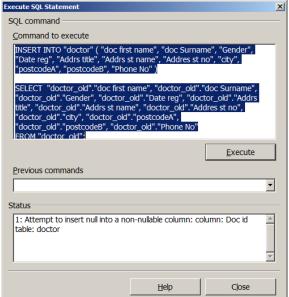
However when we click on the Execute button (ensure you select all the statement first by pressing CTRL+A) we get in effect an error message in the status box.

This is because we have tried to insert into the table two records which have Doc id values that already exist in the table which would result in duplicate values for a index field!

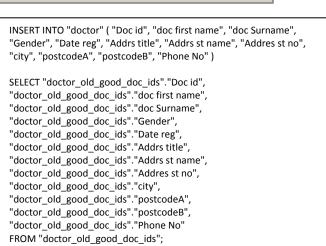


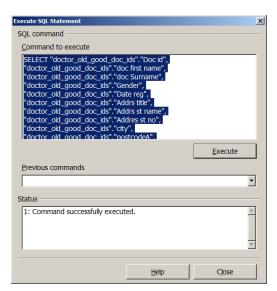
If the Doc id field is the problem the logical next step might be to remove this field from those we wish to append, editing the SQL and then rerunning the query however still results in a error (shown below):

We can't add new records to the table when they have a empty Doc id field because of the way we have setup the field!



The solution is to ensure that the Doc id field values in the Doctor old table do not appear in the doctor table (I have done this for you in the **doctor_old_good_doc_ids** table) and then rerun the query with the Doc id field included (screen shot below) using the new table name rather than the doctor_old name. Alternatively you could obviously delete the doctor_old table then rename doctor_old_good_doc_ids table to doctor_old and run the original SQL.





	Doc id	doc first name	doc Surname	Gender	Date reg	Addrs title	Addrs st name	Addres st no	city	postcodeA	postcodeB	Phone No
D	1	john	SMITH	2	01/04/67 00:	The old surger	12 station rd	Walkergate	Newcast	Ne23	4rr	0191 009282
	2	steve	Jarvis	2	01/05/70 00:	Morpeth HC	High st	Morpeth		Mo2	1ts	0127 485739
	3	Mary	Goodall	1	27/11/67 00:	Seeham HC	The harbour	Seaham		Se1	4jp	0191 345867
	4	Fiona	black	1	30/01/76 00:	Cragside HC	The East Wing	Cragside castle		Cr3	1ws	0146 389431
	5	Anna	Scriabin	1	25/03/61 00:	Uniiverisity MC	Newcastle Univerist		Newcast	Ne1	1aa	0191 445776
	23	mark	Goodall	2	27/02/55 00:	Seaham HC	The harbour	Seaham		Se1	4jp	0191 345867
	30	adam	ketalby	1	01/06/40 00:	The old surger	12 station rd	Walkergate	Newcast	Ne23	4rr	0191 009282
	31	Claire	bach	2	01/03/46 00:	Morpeth HC	High st	Morpeth		Mo2	1ts	0127 485739
**												

So now you will have 8 (i.e.6+2) records in the Doctor table.

Different databases manage **autonumber** fields differently; in Access if we had not included the Doc id field in query and doc id values would have been filled in the receiving table, but obviously they would be different values from those in the original table, however HSQLDB (the database in Base) does not do this and treats autonumber fields as if they were simply non null fields as we have done above.

```
Do I need to always specify the table name as well as the field name?

Actually you could have run the above queries omitting the table name from the select statement as shown below:

INSERT INTO "doctor" ( "Doc id", "doc first name", "doc Surname", "Gender", "Date reg", "Addrs title", "Addrs st name", "Addres st no", "city", "postcodeA", "postcodeB", "Phone No" )

SELECT "Doc id", "doc first name", "doc Surname", "Gender", "Date reg", "Addrs title", "Addrs st name", "Addres st no", "city", "postcodeA", "postcodeB", "Phone No"

FROM "doctor_old_good_doc_ids";
```

10.5 Delete Queries

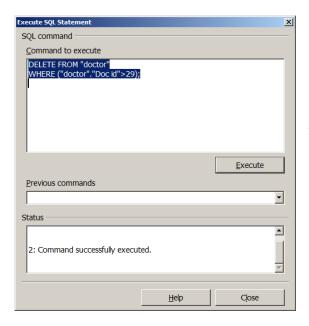
In Base again we use the Execute SQL Statement window (menu option tools -> SQL) not the query design window.

Assuming we wanted to delete all the records in the Doctor table where the Doc id value is more than 29, that is removing the two new records we inserted in the previous exercise. To achieve this we type in the SQL shown on the right below. I have also shown the Access aurogenerated SLQ equivalent if we had carried out the same operation in the Access QBE grid selecting the delete query option (see the equivalent Access chapter to this one for details).

Deleting records:

Access version:

DELETE doctor.[Doc id] FROM doctor WHERE (((doctor.[Doc id])>29)); Base (HSQL) version
DELETE FROM "doctor"
WHERE ("doctor"."Doc id">29);



Your Doctor table should now be back to its original state with 6 records.

11. Summary

This practical chapter has focused on two types of queries, Select queries involving more than one table and so called Action queries.

It has also introduced SQL and show how the Base version differs from that produced by Access.

Key point:

If running a action query always made a backup of the database or at the very least the table(s) affected by the action query.

Now go back to the beginning of the chapter and read through the 'Learning outcomes'. How many can you tick off? If you are not sure about any particular ones try reading through the section again and doing a google search on the topic.

12. Appendix A Uses of the Cross Join

The following is taken from Jeff's SQL server blog (http://weblogs.sqlteam.com/jeffs/archive/2005/09/12/7755.aspx). Jeff aka Jeff Smith, is a software developer in Boston, MA.

"Many SQL books and tutorials recommend that you "avoid cross joins" or "beware of Cartesian products" when writing your SELECT statements, which occur when you don't express joins between your tables. It's true that you need to ensure that your join conditions are adequately stated so that you don't accidentally produce this effect, but it is not true that you should avoid these types of joins in every situation.

Cross Joins produce results that consist of every combination of rows from two or more tables. That means if table A has 3 rows and table B has 2 rows, a CROSS JOIN will result in 6 rows. There is no relationship established between the two tables – you literally just produce every possible combination.

The danger here, of course, is that if you have table A with 10,000 rows and Table B with 30,000 rows, and you accidentally create the product of these two tables, you will end up with a 300,000,000 row result -- probably not a good idea. (Though it is great for creating test data and the like.)

So, how can this ever be useful? Actually, if you do lots of report writing in SQL, a CROSS JOIN can be your best friend.

Suppose you need to write a report that returns total sales for each Store and each Product. You might come up with this:

```
SELECT Store, Product, SUM(Sales) as TotalSales
FROM Sales
GROUP BY Store, Product
```

Easy enough – except when the requirement states "show \$0 if a store had no sales of a particular product". The above query won't do that – it returns no rows at all if a store had no sales for a particular product.

The solution? Well, hopefully in your database you have a table of Stores and a table of Products. A cross join of the two results will return 1 row per combination of Store and Product:

```
SELECT S.Store, P.Product
FROM Stores S
CROSS JOIN Products P
```

That result is the perfect starting point for the results we wish to return -- now we just need to return the sales for each combination. We already have written that in our first attempt, so now we just need to combine the two:

```
SELECT S.Store, P.Product, ISNULL(C.TotalSales,0) as TotalSales
FROM Stores S
CROSS JOIN Products P
LEFT OUTER JOIN
(SELECT Store, Product, SUM(Sales) as TotalSales
FROM Sales
GROUP BY Store, Product) C
ON
S.Store = C.Store AND
P. Product = C. Product
```

The SELECT is derived logically from our requirements. We start by considering all combinations of stores and products, and from there we show any matching sales data. Our primary, driving rowset is actually *not* the transaction table, but rather the cross join of two entity tables! It might seem very counter intuitive if you haven't approached the problem from this angle before, but it leads to very simple and elegant ways to solve rather complicated problems using SQL.

The solution uses what I call "the report writers magic formula":

```
(A x B ) -> (C)
```

In my made up notation, the above reads "A cross joined with B, left outer joined to C". A and B represent master tables of entities in your database, and C represents a summarized derived table of a transactional table in your database.

Some important things to note:

- All criteria for the transactions, such as date ranges and/or transaction types, need to be done in the *inner* transaction summary query.
- The summarized transactional sub-query needs to be properly grouped so that it returns 1 row per combination of A and B. Typically, this means that if the PK of table A is "A_ID" and the PK is table B is "B_ID", then the derived table C should be grouped by A_ID, B_ID.
- All criteria that determines which entities to show on your report i.e., certain regions or only "active" products should be done on the *outer* query.

Take the previous SELECT statement, for example: Note that the inner SELECT is grouped by Product and Store, which ensures that we return 1 row per combination of Product/Store — which perfectly matches what the cross join creates. If we wanted to show only data for 2005, we would put the filter on the TransactionDate column within the *inner* SELECT (since that is the part of the statement in which we collect and summarize our transactions), but if we want only ProductID #23, we do that in the *outer* SELECT (since that is where we determine the population of Stores and Products to return):

```
SELECT S.Store, P.Product, ISNULL(C.TotalSales,0) as TotalSales
FROM Stores S
CROSS JOIN Products P
LEFT OUTER JOIN
(SELECT Store, Product, SUM(Sales) as TotalSales
FROM Sales
WHERE TransactionDate between '1/1/2005' and '12/31/2005'
GROUP BY Store, Product) C
ON
S.Store = C.Store AND
P.Product = C.Product
WHERE
P.Product = 23
```

The CROSS JOIN technique can apply to many situations – to return total labor cost by office by month, even if month X has no labor cost, you can do a cross join of Offices with a table of all months. Another classic example is showing all GL transactions for a specific set of companies and accounts, returning all accounts and companies even when they have no activity.

The important thing is to practice with very small sets of sample data until you get a feel for how it works. Also, you should explicitly state CROSS JOIN in your SELECT so that it is very clear that you intend for this to happen and it is not the result of missing joins.