*Sheffield* *Hallam* *University*



Faculty of Science, Technology and Arts

**SQL** **WORKBOOK**

**55-500998**

**Database** **Systems** **For**

**Software** **Applications**

**SECTION D**

**2020/2021**



*Sheffield* *Hallam* *University*

Faculty of Science, Technology and Arts

**STRUCTURED QUERY LANGUAGE** **(SQL)**

**2020/2021**

SQL Workbook 1 September 2020

SQL Workbook 2 September 2020

Introduction

**INTRODUCTION**

The purpose of this book is to provide practical exercises in the use of SQL to create, populate and maintain a relational database.

**SQL**

SQL (Structured Query Language) is an ISO and ANSI standard *database* *query* *language.* Most relation databases are SQL-compliant, but in spite of SQL being a standard, SQL code is not completely portable among different database management systems.

**Oracle Relational Database**

For this module we will be using Oracle Database 11g Enterprise Edition.

Students will need to create a user account for Oracle, - how to do this is shown in Appendix X. If you have an account from a previous module, it will still be valid – to reset the password or unlock an account, also see Appendix X

**Oracle SQL Developer**

Students will need to connect to their Oracle database account using SQL Developer which is a **free** graphical tool for database development. How to do this is shown in Appendix Y.

With SQL Developer, you can create, browse and manage database objects, execute SQL statements and SQL scripts, and import, manipulate, and export data.

**Sample Tables**

All new accounts have a set of default tables (EMP, DEPT and SALGRADE) already created and populated with data. These are referred to as the **PERSONNEL** **SYSTEM**. Many of the **exercises** throughout this workbook are based on the Personnel System tables.

Some lecture material and various **worked** **examples** in this text are based on part of a simple **BANK** **ACCOUNTING** **SYSTEM** and use the tables CUST, CUSTACC and ACC. These tables represent the fact that a bank customer may have many accounts, and that an account may be held jointly by more than one customer.

Details of these tables can be found in the following pages. If your Personnel System tables get changed/corrupted the instructions to delete and recreate them can be found at the bottom of page 5.

**Workflow**

Topics should be tackled in sequence. This is because exercises in later sections may rely on changes you make to your data or data structures in earlier sections to work properly. Topics must be completed before the scheduled session of the next topic.

SQL Workbook 3 September 2020

Introduction

**The Bank Accounting System**

CUST Owns CUSTACC Allocated ACC

Table: **CUST**

|  |  |  |  |
| --- | --- | --- | --- |
| REFNO | NAME | ADDRESS | AREA |
| A123 A124 B127 B128  C371 | J Doe J Smith R Best J Best  R Done | 1 High Street 2 West Street 4 East Row  4 East Row  23 Middle Avenue | Sheffield Sheffield Rotherham Rotherham  Barnsley |

Table: **CUSTACC**

|  |  |
| --- | --- |
| REFNO | ACCNO |
| A123 A123 B127  B128 | 1245890 1494315 5418490  5418490 |

Table: **ACC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACCNO | BALANCE | BRANCH | OPENED | BONUS |
| 1245890 1494315  5418490 | 234.50 0.50  1789.40 | Broomhill Tinsley  Broomhill | 12 Nov 2003 15 Dec 1999  6 May 1988 | 100.00 0.00 |

SQL Workbook 4 September 2020

Introduction

**THE PERSONNEL SYSTEM**

Table: **EMP**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
| 7369 7499 7521 7566 7654 7698 7782 7788 7839 7844 7876 7900 7902  7934 | SMITH ALLEN WARD JONES MARTIN BLAKE CLARK SCOTT KING TURNER ADAMS JAMES FORD  MILLER | CLERK SALESMAN SALESMAN MANAGER SALESMAN MANAGER MANAGER ANALYST PRESIDENT SALESMAN CLERK CLERK ANALYST  CLERK | 7902 7698 7698 7839 7698 7839 7839 7566  7698 7788 7698 7566  7782 | 17-DEC-80 20-FEB-81 22-FEB-81 02-APR-81 28-SEP-81 01-MAY-81 09-JUN-81 09-DEC-82 17-NOV-81 08-SEP-81 12-JAN-83 03-DEC-81 03-DEC-81  23-JAN-82 | 800.00 1600.00 1250.00 2975.00 1200.00 2850.00 2450.00 3000.00 5000.00 1500.00 1100.00 950.00 3000.00  1300.00 | 300.00 500.00  1250.00  0.00 | 20 30 30 20 30 30 10 20  30 20 30 20  10 |

Table: **DEPT**

|  |  |  |
| --- | --- | --- |
| DEPTNO | DNAME | LOC |
| 10 20 30 40 | ACCOUNTING RESEARCH SALES OPERATIONS | NEW YORK DALLAS CHICAGO BOSTON |

Table: **SALGRADE**

|  |  |  |
| --- | --- | --- |
| GRADE | LOSAL | HISAL |
| 1 2 3 4  5 | 700.00 1201.00 1401.00 2001.00  3001.00 | 1200.00 1400.00 2000.00 3000.00  9999.00 |

Should data in the tables become corrupt, they may be restored to their original status by issuing each of the following statements for the appropriate table:

**DROP** **TABLE** EMP ;

**CREATE** **TABLE** EMP **AS** **SELECT** \* **FROM** EXAMPLE.EMP ;

SQL Workbook 5 September 2020

Introduction

This page is intentionally blank.

SQL Workbook 6 September 2020

Section D

**CONSTRAINTS**

So far, our database tables have no protection from simple mistakes. We can insert duplicate rows, we can set any values that we like for a column, we can drop entire tables, or we can create employees (say) for a department which does not exist.

Errors of this kind threaten to compromise the integrity of the data, but relational databases provide various checking mechanisms – known as *constraints* which can (and should) be used to help ensure the integrity of the data.

Data integrity can be considered in terms of three categories:

**Entity** **integrity** is concerned with ensuring that each instance of a row in a table has a unique identifier. (eg. to prevent creating two employees with the same EMPNO)

Entity integrity is addressed by defining a Primary Key for a table, and by implementing a Primary Key Constraint. In some cases, a Unique Key and a Unique Key Constraint is also required.

**Referential** **integrity** is concerned with ensuring that related data in different tables remains consistent. (eg. to prevent having an employee with a DEPTNO which does not occur in the DEPT table).

Referential integrity is addressed by defining one or more Foreign Key(s) for a table, and by implementing Foreign Key Constraints.

**Domain** **integrity** is concerned with ensuring that column data values fall within an allowable value range. (eg. to prevent SAL being less than a specified minimum)

Domain integrity is addressed by defining one or more constraints for a column within a table – to limit the column value. (Note that the column datatype is also a constraint in that it limits the format of the data.)

Constraints are defined within the table CREATE statement or may be added (or removed) from an existing table using the ALTER statement (see later).

**Within** **the** **CREATE** **statement,** **constraints** **can** **be** **defined** **at** **column-level** **or** **table-level.**

Domain constraints are usually defined at column level. Primary Key may be defined at column-level for a single column. However, Primary Key and Foreign Key constraints are best defined at table-level, - where they can be specifically named (see later).

SQL Workbook 39 September 2020

Section D

**Column-Level:** **Domain & Primary Key Constraints**

At column-level the syntax is:

CREATE TABLE tablename (

Column\_name1 datatype1 constraint1 constraint2 c3 . . . etc, Column\_name2 datatype2 constraint1 constraint2,

etc );

Several column-level constraints may be applied to a column, in any order, and without commas between them, but the last one must have a following comma to end the column definition (unless it is the last column defined).

A column-level constraint can be any of the following:

|  |  |
| --- | --- |
| **Constraint** | **Effect** |
| PRIMARY KEY | this column is identified as the Primary Key, and by default this will enforce Unique and Not Null constraints |
| NOT NULL | column cannot be null |
| UNIQUE | column value must be unique among all rows in table |
| DEFAULT *expression* | if not specified, value defaults to *expression* |
| CHECK (*this\_column\_name* *operator* *expression)* | value must conform to the rule given |

**Example** CREATE TABLE ACC (

ACCNO NUMBER(7,0) PRIMARY KEY, BALANCE NUMBER(7,2) DEFAULT 0, BRANCH VARCHAR2(20) NOT NULL, OPENED DATE DEFAULT sysdate,

BONUS NUMBER(7,2) NOT NULL CHECK (BONUS < 500)

)

- ACCNO is Primary Key - default balance is zero

- must specify branch name - date opened = ‘today’

- bonus cannot be null, and has a max of £500

The domain and primary key constraints are checked automatically every time a row is inserted or updated.

SQL Workbook 40 September 2020

Section D

**Exercises**

D1 Create a table named ORDERS, with these columns and data types: OrderNo number, OrderDate date, CustomerID char(4).

In the create statement use column-level constraints to achieve the following: - OrderNo is the primary key

- OrderDate defaults to the current date - CustomerID cannot be null

CREATE TABLE ORDERS (

OrderNo NUMBER PRIMARY KEY,

OrderDate DATE DEFAULT sysdate,

CustomerID CHAR(4) NOT NULL

);

D2 Write a statement to insert this data into the table: OrderNo = 25 CustomerID = ‘A123’

INSERT INTO ORDERS (ORDERNO, CUSTOMERID)

VALUES (25, ‘A123’);

If / when it works, use SELECT \* FROM ORDERS to display the row. Note the default date.

D3 Experiment by editing the insert statement to try values which violate the constraints, and then observe the error messages.

D4 You can exploit the fact that the Run Statement executes only the statement containing the cursor to experiment very easily with different constraints, . . . by arranging the above statements in the code window like this . .

DROP TABLE ORDERS; CREATE TABLE ORDERS( etc..

):

INSERT INTO ORDERS etc ); SELECT \* FROM ORDERS;

. . . and then edit and execute so as to, drop the table, create a new version with different constraints, try different inserts, and select to view the effects.

Confirm that you cannot enter the same row more than once with the primary key defined, but that you can otherwise.

SQL Workbook 41 September 2020

Section D

**Table-Level:** **Primary Key & Foreign Key Constraints**

The table-level syntax for Primary and Foreign Keys is illustrated in three forms:

**1.** **For** **a** **Primary** **Key** **as** **a** **single** **column,** **or** **a** **Foreign** **Key** **as** **a** **single** **column.**

CREATE TABLE tablename ( column\_name datatype, column\_name datatype, column\_name datatype,

. . . etc then

PRIMARY KEY (column\_name1),

FOREIGN KEY (column\_name2) REFERENCES table\_name3(column\_name4), );

1 name of the column in this table which is the primary key 2 name of the column in this table which is the foreign key 3 name of the other (foreign) table

4 name of the primary key column of the other (foreign) table

The Foreign Key constraint ensures that the value of the foreign key column in this table matches the primary key value of a row in the other table.

In this example, the foreign key on deptno in the EMP table ensures that there will be a matching row in the DEPT table. (That is, if deptno is 10, then department 10 must exist in DEPT).

**Example** CREATE TABLE EMP (

empno NUMBER(4), . . . .

deptno NUMBER (2,0), PRIMARY KEY (empno),

FOREIGN KEY (deptno) REFERENCES dept(deptno) );

**Exercises**

D5 Drop and recreate the ORDERS table from the last exercise, but specify the primary key as a table-level constraint. The constraint definition follows the list of column definitions. It must specify the column intended as the key.

CREATE TABLE ORDERS (

OrderNo NUMBER,

OrderDate DATE DEFAULT sysdate,

CustomerID CHAR(4) NOT NULL,

PRIMARY KEY (ORDERNO)

);

SQL Workbook 42 September 2020

Section D

D6 Drop and recreate the ORDERS table, but this time also specify the CUSTOMERID column as a foreign key (to the CUST table) using a table-level constraint. The constraint definition must specify the column intended as the key, the other table, and its primary key.

CREATE TABLE ORDERS(

OrderNo NUMBER,

OrderDate DATE DEFAULT sysdate,

CustomerID CHAR(4) NOT NULL,

PRIMARY KEY (orderno),

FOREIGN KEY (customerid) REFERENCES cust(refno) );

This should fail, because there is no primary key defined for the CUST table, -and a foreign key must refer to a primary key.

To alter the existing table cust without dropping and recreating it, issue this statement:

ALTER TABLE CUST

MODIFY REFNO CHAR(4) PRIMARY KEY;

Now re-submit the create ORDERS table statement. It should succeed and the foreign key relationship is now established.

D7 Repeat the earlier statement to insert this data into the table: OrderNo = 25 CustomerID = ‘A123’. (It should work).

Now edit the statement to insert OrderNo 36 for Customer ‘X999’. The error message arises because the customer does not exist in the CUST table.

Now, - try to execute this statement:

DELETE FROM CUST WHERE REFNO = ‘A123’;

The error message arises because deleting this customer would leave an order in the ORDERS table without a valid customer.

The last two exercises demonstrate how the use of primary and foreign keys and constraints enforces referential integrity in three ways:

• foreign keys must refer to primary keys.

• child records cannot be created without a parent.

• parent records cannot be deleted while child records exist

(We have also had a preview of ALTER from a later section)

SQL Workbook 43 September 2020

Section D

**2.** **Primary** **Key** **and/or** **Foreign** **Key** **as** **multiple** **columns.**

A Primary Key may consist of more than one column. If it does, it MUST be specified at table-level, and the constraint must refer to each column, like this:

PRIMARY KEY (column\_name, column\_name)

And, if that primary key is the target of a foreign key from another table, then the corresponding foreign key constraint (on the other table) will need to refer to each column of the primary key, and will look like this:

FOREIGN KEY (column\_name, column\_name) REFERENCES table\_name(column\_name, column\_name)

**Exercises**

D8 Create a table named ORDERLINE, with these columns and data types: OrderNo number, ProductID number, Quantity number.

This table allows us to have several different products as part of the same order, and store the quantity of each one. It enables a 1-to-many relationship. The unique identifier is the combination of two columns (orderid, productid) and together these form a compound primary key. The orderid by itself refers to the primary key of the ORDERS table, so it is also a foreign key

In the create statement use column-level constraints to achieve the following: - ProductID can’t be null

- Quantity must be non-zero but less than 100

Also in the create statement use table-level constraints to define the primary and foreign keys.

CREATE TABLE ORDERLINE(

ORDERNO NUMBER,

PRODUCTID NUMBER NOT NULL,

QUANTITY NUMBER CHECK (QUANTITY > 0 AND QUANTITY < 100),

PRIMARY KEY (orderno, productid),

FOREIGN KEY (orderno) REFERENCES orders(orderno) );

D9 Write some insert statements to create products and ORDERLINES table. Experiment with valid and invalid constraints.

quantities in the data to test the

SQL Workbook 44 September 2020

Section D

**3.** **Named** **Constraints**

Referential integrity constraints sometimes need to be changed or removed, or disabled. It is therefore important to be able to identify them, but Oracle simply assigns a number (such as SYS\_C00198655) to each one.

It is possible to assign names to constraints, and it is good practise to do this and to use a systematic naming scheme. The syntax for naming requires use of the CONSTRAINT keyword followed by your chosen name, and placed before the constraint definition. Like this:

CONSTRAINT constraint\_name PRIMARY KEY (column\_name) or

CONSTRAINT constraint\_name FOREIGN KEY (column\_name) REFERENCES table\_name(column\_name)

This example names the primary key of the EMP table as **emp\_pk**

and the foreign key from the EMP table to the DEPT table as **emp\_fk\_dept**

**Example** CREATE TABLE EMP (

empno NUMBER(4), . . . .

deptno NUMBER (2,0),

CONSTRAINT emp\_pk PRIMARY KEY (empno), CONSTRAINT emp\_fk\_deptno FOREIGN KEY (deptno) REFERENCES dept(deptno)

);

**Exercises**

D10 Execute these statements, either separately, or use Run Script .

PURGE RECYCLEBIN;

SELECT \* FROM USER\_CONSTRAINTS;

The first will drop all deleted tables and constraints, the second will display all of your live constraints. Notice that the constraint ‘names’ assigned by Oracle are not meaningful, and that the ‘type Rs’ (foreign key constraints) do not say what they refer to.

D11 Drop the tables ORDERS and ORDERLINES, and recreate them using the syntax shown above to assign names to the primary and foreign key constraints. Use a systematic naming scheme which indicates what the constraint is for, - eg. ‘orders\_pk’.

Display the constraints again to see the new names. Check constraints can also be given names by defining them at table-level using the CONSTRAINT keyword.

SQL Workbook 45 September 2020

Section D

**Database** **Export** **/** **Edit** **/** **Import** **exercise**

D12 Follow this procedure to export the Accounting System table structures and data to a file. Then edit it to add constraints, save it, and then execute it as a script to recreate the tables and data:

D12-A

Choose Tools> Database Export from the top menu.

Click Browse, go to F:/mywork/oracle folder and specify a filename (eg. accounts.sql) Click Connection, and choose yours (if not set already)

Check the box marked Include Drop Statement Click Next

Uncheck Toggle All

Check Tables, and Data, then click Next

Click Go, then highlight the ACC, CUST and CUSTACC tables

Move them to the right hand column, then click Next. (selects tables)

Click Go, then highlight the ACC, CUST and CUSTACC tables

Move them to the right hand column, then click Next. (selects data)

Review the ‘export summary’ then click Finish.

The file is saved to disk but also displayed in a code window.

Notice the comments, and the DROP statements, and the data INSERT statements at the bottom .

Execute this file as a script simply by clicking the Run Script button. It will drop all the tables and recreate them exactly as they were.

D12-B

Edit the script to add some appropriate domain constraints, and add named primary key and foreign key constraints to all three tables.

Hint: ACC and CUST need PK’s. CUSTACC needs a compound PK and two different FK’s.

Keep running the script until it is error free. Save it with File>Save As, or by trying to close the code window.

Test the constraints by trying these insert statements:

INSERT INTO CUST VALUES ('A125', 'J Cole', 'New York', 'USA'); INSERT INTO CUST VALUES ('B128', 'S Cole', 'New York', 'USA'); INSERT INTO CUSTACC VALUES ('A125', 1494315);

INSERT INTO CUSTACC VALUES ('B128', 5418490); INSERT INTO CUSTACC VALUES ('B128', 1234567);

Make sure that the error messages match what you expect.

SQL Workbook 46 September 2020

Section D

**The ALTER statement**

The ALTER statement is used to alter structure of an existing table, by adding table elements, modifying column definitions and adding or dropping constraints. The syntax is variable , but the following examples should make the possibilities clear.

E.g. **ALTER** **TABLE** DEPT2

**ADD** NO\_OF\_EMPLOYEES NUMBER(3) ; Creates a new column

**ALTER** **TABLE** DEPT2 **DROP** **COLUMN** LOC ;

**ALTER** **TABLE** DEPT2

**MODIFY** DEPTNO NUMBER(3) ;

**ALTER** **TABLE** DEPT2

**DROP** **CONSTRAINT** PK1 ;

Removes a column

Changes the data type

Removes a named constraint

**ALTER** **TABLE** DEPT2

**ADD** **CHECK** (LOC **IN** ('NEW YORK', 'DALLAS', 'CHICAGO', 'BOSTON') ; Adds a CHECK constraint

**ALTER** **TABLE** DEPT2

**ADD** **CONSTRAINT** PK1 **PRIMARY** **KEY** (DEPTNO) ;

Adds a named constraint

**ALTER** **TABLE** DEPT2

**MODIFY** (DNAME **NOT** **NULL**) ; Adds constraint to one column

**ALTER** **TABLE** DEPT2

**MODIFY** (DNAME **NOT** **NULL,** LOC **NOT** **NULL)** ;

Adds constraint to two columns

Also,

**ALTER** **TABLE** DEPT2 **RENAME** **TO** DEPT3;

Renames the table

**ALTER** **TABLE** DEPT3 **RENAME** **COLUMN** DNAME **TO** DEPTNAME; Renames the column

SQL Workbook 47 September 2020