

Laboratory Handout

Introduction to Oracle

SQL*Plus — schemata — data types — DML & DDL examples — editing commands — using external files — the dual pseudo-table — introduction to transactions — optional exercise — references.

Introduction During this laboratory you will build on your knowledge acquired with Postgres by learning the fundamentals of using *Oracle*, one of the most widely utilised database management system in industry.

SQL*Plus

SQL*Plus is Oracle's command-line interpreter. You may launch SQL*Plus by issuing the `sqlplus` command in UNIX or using the 'start' menu in Windows. In the 'start' menu, SQL*Plus is listed under `programs > oracle > application development > SQL Plus`.

You will be prompted for your username and password, which should both initially be set to your username. You will learn at a later stage how to change your password. The last piece of information required by SQL*Plus is the name (called *host string*) of the database you want to use: type in `mcsora`.

You are now connected to a shared database, on which you have an account (called a *schema*). You do not have to create a new database as you would do with Postgres.

Some Oracle SQL

Table 1 outlines the main Oracle SQL data types, together with their Postgres equivalent. Noteworthy is the `VARCHAR2` type, so called for historical reasons. Oracle does not provide a specific currency data type.

Type description	Oracle SQL	Postgres SQL
variable-length char. string	VARCHAR2(1) ^a	VARCHAR(1)
fixed-length char. string	CHAR(1)	CHAR(1)
number	NUMBER(p,s) ^b	NUMERIC(p,s)
currency	NUMBER(10,2)	MONEY
date	DATE	DATE

^alength.

^bposition, scale.

Table 1: The main SQL data types.

You should now be able to create a few tables and populate them.

```
CREATE TABLE books
(
  title VARCHAR2(60),
  author VARCHAR2(60),
  isbn NUMBER(10,0)
  CONSTRAINT pk_books PRIMARY KEY,
  pub_date DATE DEFAULT SYSDATE
)
/

CREATE TABLE book_reviews
(
  isbn NUMBER(10,0)
  CONSTRAINT fk_books_booksrev REFERENCES books(isbn),
  reviewer VARCHAR2(30),
  comments VARCHAR2(150)
)
/
```

Note the use of the `SYSDATE` function that returns the system's current date in the `DEFAULT` clause above. The `'/'` character terminates an SQL statement and submits it to SQL*Plus.

You should be already familiar with the syntax of the *primary key* and *referential integrity* constraints. They function in Oracle in a similar fashion as in Postgres. `pk_books` and `fk_books_booksrev` are constraint names.

Now check the schema of the tables you have just created (c.f. `\dt` in Postgres) using the `desc <table_name>` command.

Next, we want to insert some data into `books` and `books_reviews`:

```

INSERT INTO books VALUES
(
  'The Importance of Being Earnest',
  'Oscar Wilde', -- this is a comment
  9876543210,
  '14-FEB-1895'
)
/

```

```

INSERT INTO book_reviews VALUES
(
  9876543210,
  'Alice',
  'Excellent work, humorous and witty.'
)
/

```

As shown above, the date format expected by Oracle is DD-MMM-YYYY or DD-MMM-YY. The double hyphen sequence '--' introduces a comment.

Editing Commands

Editing SQL*Plus' buffer. As you may already have experienced, you *cannot* recall statements after they have been submitted to SQL*Plus. The `ed` command allows you to edit the SQL*Plus buffer in the system's default editor. After saving your changes, submit the statement with a `/`. Be aware that only the *last* statement *submitted* to SQL*Plus may be edited.

Using command files. A practical approach not to inadvertently lose your SQL work is to use command files—as you (should) have already done with Postgres (c.f. `\i` command).

1. type in your SQL statements in your favourite editor.
2. save the file with the `.sql` extension in your home directory (e.g. `myfile.sql`)—make sure that you get the correct extension, as some editors will attempt to append a `.txt` extension.
3. type in `@myfile` at the SQL*Plus command prompt to execute your SQL statement(s).

Before starting the next section, you should practise creating and populating some more tables.

More Oracle SQL

You are now armed to attempt some more complex SQL expressions.

The dual pseudo-table. Oracle insists that *all* `SELECT` statements be of the form “`SELECT <attribute> FROM <table>`”—even when the returned value does not depend on data stored in the database. The `DUAL` pseudo-table was introduced to allow such statements.

```
SELECT 'Hello' FROM DUAL    -- shows 'Hello'
/
SELECT SYSDATE FROM DUAL   -- shows the date
/
```

Sequence numbers. A `SEQUENCE` is an Oracle object that generates integers according to a specific pattern. Sequence numbers are commonly utilised to supply auto-generated primary keys. The default behaviour of a `SEQUENCE` is to increment its current value by one to get the next. You may already have used sequences in Postgres, using the `SERIAL` data type. Note however the two differences with Oracle’s sequences:

- in Postgres fields that will be populated using a sequence need to be declared as `SERIAL`, in Oracle they are of type `NUMBER`
- Postgres increments the sequence when required, you do not have to do it explicitly

The following code creates a `SEQUENCE` that we will then use to insert some more values in the `books` table.

```
CREATE SEQUENCE book_seq
/
SELECT book_seq.CURVAL FROM DUAL -- shows the current value
/
SELECT book_seq.NEXTVAL FROM DUAL -- displays the next value
/
INSERT INTO books VALUES
(
  'Oliver Twist',
  'Charles Dickens',
  book_seq.NEXTVAL,
  '12-SEP-1839'
)
/
```

Apart from the the Oracle peculiarities we have already discussed, you can re-use most of your knowledge of SQL. You may want for example to experiment with the `UPDATE` and `DELETE` statements.

Introduction to Transactions

Transaction management is a broad topic to which you have been introduced in the database lectures. You should refer to your notes for a more detailed coverage of the subject, as we will here just remind a few points. A transaction is a *logical unit of work*, that could be for example the placement of an order. On completion, a transaction needs to be either *confirmed*—making all the changes permanent—or *cancelled*—returning the database into the state it was before starting the transaction.

These two actions are performed in SQL by issuing one of the two commands `COMMIT` or `ROLLBACK`.

To experiment with transactions, you will need to work in pairs (say Alice and Bob) and allow the other student to read the data in your `books` table. So Alice will need to enter:

```
GRANT SELECT ON books TO bob
/
```

and Bob to enter:

```
GRANT SELECT ON books TO alice
/
```

Now Alice should enter some data in her `books` table. Bob can then attempt to view the newly inserted data by typing:

```
SELECT * FROM alice.books
/
```

Note how you can prefix the table name with its schema to reference other students' tables. Can Bob view the changes Alice has made? What happens if Alice `COMMITs` the transaction? Try also with `ROLLBACK`.

Try to relate your observations with your understanding of transactions.

Optional Exercise Suggestion

ISBNs (International Standard Book Number) are unique, 10-digit book identifiers used in the publishing industry. With the help of the references given at the end of this document, create a sequence to generate 10-digit integers for use with the `books` table.

References

You can copy & paste the following URIs (note that you will need a username/password to access Oracle's web site. You can use `data@base.com/database`):

Oracle SQL & standard SQL compared:

<http://www-db.stanford.edu/~ullman/fcdb/oracle/or-nonstandard.html>

Oracle SQL reference:

<http://technet.oracle.com/doc/server.815/a67779/toc.htm>

Oracle SQL*Plus quick reference:

<http://technet.oracle.com/doc/server.815/a66735/toc.htm>

Oracle error messages:

<http://technet.oracle.com/doc/server.815/a67785/toc.htm>