Applied 7 Creating and Altering the Database Structure (DDL)

Applied 7 SQL DDL

At the completion of the activity, you should be able to:

- relate the SQL DDL statements such as CREATE TABLE to the relational model theory
- create tables in a relational database
- use the ALTER command to make changes to an existing table
- create appropriate FK constraints based on a supplied model
- remove tables from a database

This activity supports unit learning outcomes 1, 2 and 3.

A7-1 SQL Data Definition Language (DDL)

Creating Tables

When creating schema files, you should always also create a drop file or add the drop commands to the top of your schema file. You should drop the tables using the

```
drop table tablename purge;
```

syntax. If you use this syntax, the drop table statements must be arranged in an order such that the child tables (the ones holding the FK's) are dropped before its matching parent table (the one the PK was copied from to create the FK). Doing it in this correct order will prevent any foreign key errors since when you attempt to drop the parent table there are no longer any FK's remaining. In a complex system with lots of relationships this order can be difficult or even impossible to arrive at.

The alternative which can be used is:

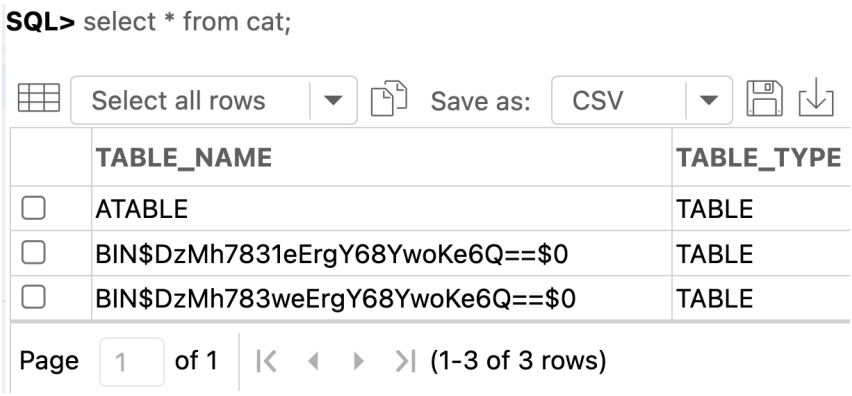
```
drop table tablename cascade constraints purge;
```

This syntax drops the table and removes all FK constraints the table is involved in. If you use this syntax, the order of table deletion then does not matter (it is simplest to do the drops in alphabetical order).

If you drop a table without adding the PURGE clause - the table will be placed into your Oracle account's recycle bin. You can view the recycle bin's contents via the SQL command:

```
select * from cat;
```

cat is an abbreviation for the list of tables you own (the catalogue). The returned table names will be prefaced with BIN$ if the table is currently in your recycle bin (here ATABLE is not in the recycle bin but all other tables shown are).



Since you have limited space in your Oracle account, it is important that you keep the recycle bin empty - you can do this by ensuring, when you drop a table, that you use the PURGE

syntax as above, and/or by regularly using the SQL command:

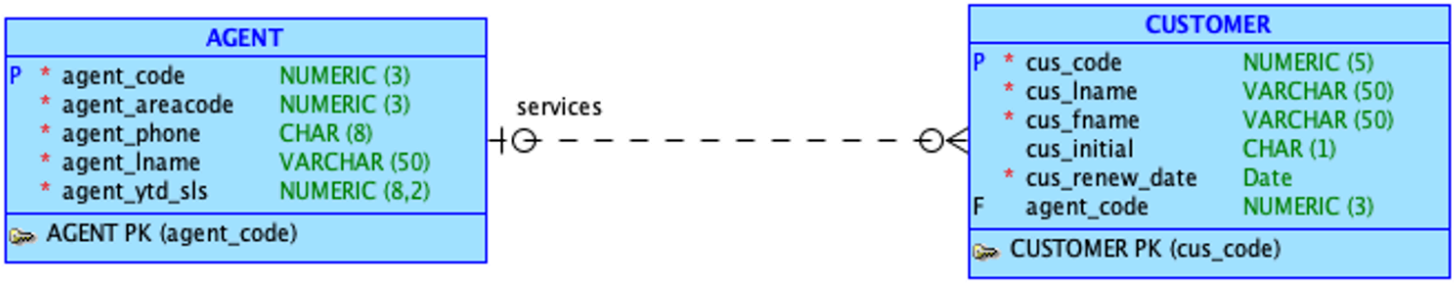
```
purge recyclebin;
```

Should a syntax error occur while testing your schema, you simply need to run the drop commands to remove any tables which may have been created and start afresh.

An excellent summary of the Oracle data types and version restrictions is available from:

https://www.oracletutorial.com/oracle-basics/oracle-data-types/

For this unit, we make use of **CHAR, VARCHAR2, NUMBER** and **DATE ONLY.**



The data model above represents figure 3.3 from Coronel & Morris. Note that this diagram is not a logical model. ***Logical models do not show data types and data sizes.*** Also note this is not an Oracle physical model, since the data types are not standard Oracle types - NUMERIC should be NUMBER and VARCHAR should be VARCHAR2.

There are two different ways of coding this model as a set of create table statements as discussed in the following subsections.

Using table constraints

SQL constraints are classified as column or table constraints; depending on which item they are attached to:

```
CREATE TABLE agent (
    agent_code      NUMBER(3) CONSTRAINT agent_pk PRIMARY KEY,
    agent_areacode  NUMBER(3) NOT NULL,
    agent_phone     CHAR(8) NOT NULL,
    agent_lname     VARCHAR2(50) NOT NULL,
    agent_ytd_sls   NUMBER(8, 2) NOT NULL
);
```

This is a declaration of the primary key as a column constraint.

```
CREATE TABLE agent (
    agent_code      NUMBER(3) NOT NULL,
    agent_areacode  NUMBER(3) NOT NULL,
    agent_phone     CHAR(8) NOT NULL,
    agent_lname     VARCHAR2(50) NOT NULL,
    agent_ytd_sls   NUMBER(8, 2) NOT NULL,
```

```
    CONSTRAINT agent_pk PRIMARY KEY ( agent_code )
);
```

Here the primary key has been declared as a table constraint, at the end of the table after all column declarations have been completed. In some circumstances, for example, a composite primary key you must use a table constraint since a column constraint can only refer to a single column.

The create table statements for the two tables in fig 3-3 would be:

```
CREATE TABLE agent (
    agent_code      NUMBER(3) NOT NULL,
    agent_areacode  NUMBER(3) NOT NULL,
    agent_phone     CHAR(8) NOT NULL,
    agent_lname     VARCHAR2(50) NOT NULL,
    agent_ytd_sls   NUMBER(8, 2) NOT NULL,
    CONSTRAINT agent_pk PRIMARY KEY ( agent_code )
);

COMMENT ON COLUMN agent.agent_code IS
    'agent code (unique for each agent)';

COMMENT ON COLUMN agent.agent_areacode IS
    'area code of agent';

COMMENT ON COLUMN agent.agent_phone IS
    'phone number of agent code';

COMMENT ON COLUMN agent.agent_lname IS
    'last name of agent';

COMMENT ON COLUMN agent.agent_ytd_sls IS
    'year to date sales made by agent';

CREATE TABLE customer (
    cus_code        NUMBER(5) NOT NULL,
    cus_lname       VARCHAR2(50) NOT NULL,
    cus_fname       VARCHAR2(50) NOT NULL,
    cus_initial     CHAR(1),
    cus_renew_date  DATE NOT NULL,
    agent_code      NUMBER(3),
    CONSTRAINT customer_pk PRIMARY KEY ( cus_code ),
    CONSTRAINT agent_customer_fk FOREIGN KEY ( agent_code )
        REFERENCES agent ( agent_code )
            ON DELETE SET NULL
);

COMMENT ON COLUMN customer.cus_code IS
    'customer code (unique for each customer)';

COMMENT ON COLUMN customer.cus_lname IS
    'last name of customer';

COMMENT ON COLUMN customer.cus_fname IS
```

```
    'first name of customer';

COMMENT ON COLUMN customer.cus_initial IS
    'initial of customer (not mandatory)';

COMMENT ON COLUMN customer.cus_renew_date IS
    'insurance renewal date of customer';

COMMENT ON COLUMN customer.agent_code IS
    'agent code (unique for each agent)';
```

The inclusion of the referential integrity rule *on delete set null* in the above create table statement is **appropriate in this scenario** - when an agent leaves, a reasonable approach would be to set the foreign key for that agent's customers to null.

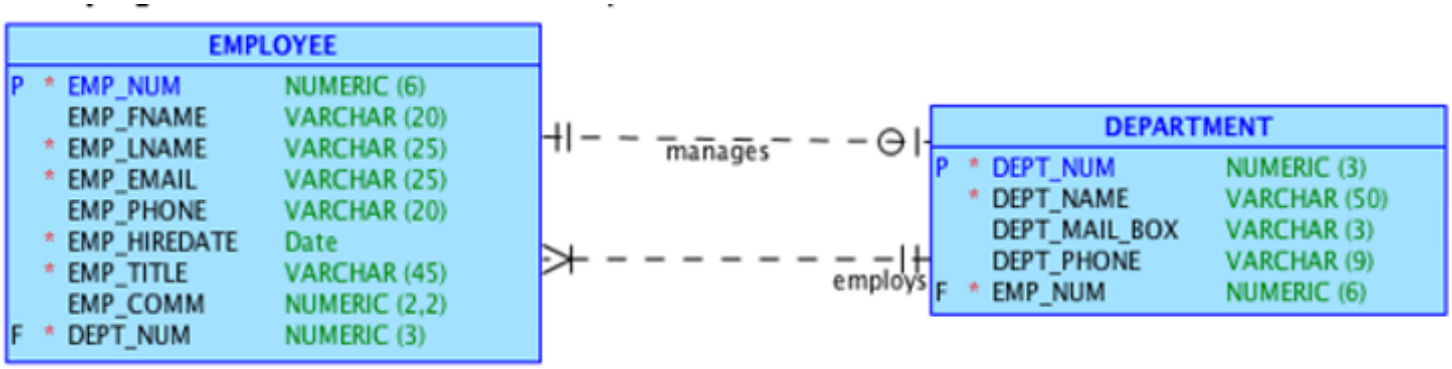
The default on delete restrict (**which you do not specify, simply omit an on delete clause**) would also be an alternative approach.

```
CREATE TABLE customer (
    cus_code        NUMBER(5) NOT NULL,
    cus_lname       VARCHAR2(50) NOT NULL,
    cus_fname       VARCHAR2(50) NOT NULL,
    cus_initial     CHAR(1),
    cus_renew_date  DATE NOT NULL,
    agent_code      NUMBER(3),
    CONSTRAINT customer_pk PRIMARY KEY ( cus_code ),
    CONSTRAINT agent_customer_fk FOREIGN KEY ( agent_code )
        REFERENCES agent ( agent_code )
);
```

Using *on delete cascade* would not be appropriate since this would cause the customers of the agent who left to also be deleted. When coding a foreign key definition, you **must always consider what is a suitable 'on delete' approach (RESTRICT, CASCADE, NULLIFY) for the scenario you are working with**.

Using ALTER table commands

In some circumstances, this approach of defining the foreign keys as part of the table definitions cannot be used. Observe the data model below. Note that this diagram is not a logical model. Logical model does not show data types and data sizes. Can you see what the issue is with trying to create the two tables depicted below?



In such a situation an alternative approach to declaring constraints needs to be adopted.

In this approach, the tables are declared without constraints and then the constraints are applied via the ALTER TABLE command.

```
CREATE TABLE agent (
    agent_code      NUMBER(3) NOT NULL,
    agent_areacode  NUMBER(3) NOT NULL,
    agent_phone     CHAR(8) NOT NULL,
    agent_lname     VARCHAR2(50) NOT NULL,
    agent_ytd_sls   NUMBER(8, 2) NOT NULL
);

COMMENT ON COLUMN agent.agent_code IS
    'agent code (unique for each agent)';

COMMENT ON COLUMN agent.agent_areacode IS
    'area code of agent';

COMMENT ON COLUMN agent.agent_phone IS
    'agent phone number';

COMMENT ON COLUMN agent.agent_lname IS
    'agent last name';

COMMENT ON COLUMN agent.agent_ytd_sls IS
    'year to date sales made by agent';

ALTER TABLE agent ADD CONSTRAINT agent_pk PRIMARY KEY ( agent_code );

CREATE TABLE customer (
    cus_code        NUMBER(5) NOT NULL,
    cus_lname       VARCHAR2(50) NOT NULL,
    cus_fname       VARCHAR2(50) NOT NULL,
    cus_initial     CHAR(1),
    cus_renew_date  DATE NOT NULL,
    agent_code      NUMBER(3)
);

COMMENT ON COLUMN customer.cus_code IS
    'customer code (unique for each customer)';
```

```
COMMENT ON COLUMN customer.cus_lname IS
    'customer last name';

COMMENT ON COLUMN customer.cus_fname IS
    'customer first name';

COMMENT ON COLUMN customer.cus_initial IS
    'customer initial (not mandatory)';

COMMENT ON COLUMN customer.cus_renew_date IS
    'customer insurance renewal date';

COMMENT ON COLUMN customer.agent_code IS
    'agent code (unique for each agent)';

ALTER TABLE customer ADD CONSTRAINT customer_pk PRIMARY KEY ( cus_code );

ALTER TABLE customer
    ADD CONSTRAINT agent_customer_fk FOREIGN KEY ( agent_code )
        REFERENCES agent ( agent_code )
            ON DELETE SET NULL;
```

Remember, from above, when coding a foreign key definition you **must always consider what is a suitable 'on delete' approach (RESTRICT, CASCADE, NULLIFY) for the scenario you are working with**.

Using an ALTER Table approach **is the best method, and the approach we require you to use**, since it cannot fail due to missing required tables for foreign key constraints. Using this approach, the tables can be created in alphabetical order and then the required constraints applied. This is the approach used by Data Modeller (and most commercial software) when creating schema files.

Using CHECK constraints

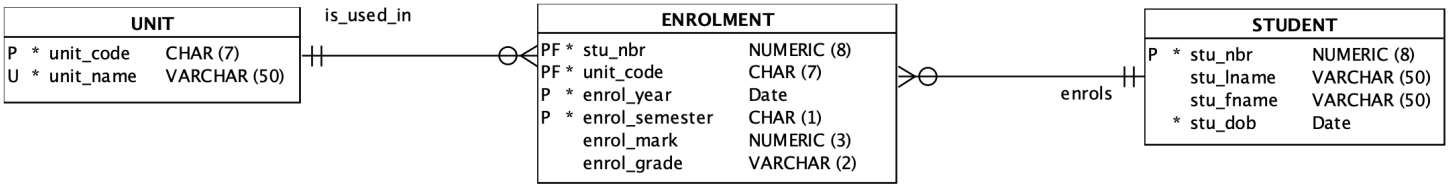
When adding CHECK clauses, the value used for the check values must be a single character (CHAR(1)); the *only exception* to this is where standard abbreviations exist. For example, if you were building a check clause to:

- control the level of a customer (Gold, Silver or Bronze) as you did in the Logical Modeling Applied, we would use values of G, S or B, i.e. CHAR(1)
- control the states of Australia, we would use values of VIC, NSW, QLD, etc, as standard abbreviations exist, i.e. CHAR(3)

A7-2 DDL Create Table Tasks

Note for our unit, **ALL CONSTRAINTS, other than NOT NULL constraints, must be explicitly named** via the syntax CONSTRAINT *constraint_name*

Using the data model for student, unit and enrolment shown below, please complete the following tasks.



1. Download the schema start file (applied7_schema.sql) below and save this under the Applied 7 folder in your local repository:

   applied7_schema.sql

2. Using this file complete the schema to create these three tables, noting the following ***extra*** constraints:
   - stu_nbr > 10000000
   - unit_name is unique in the UNIT table
   - enrol_semester can only contain the value of 1 or 2 or 3.
3. In implementing these constraints, you will need to make use of CHECK clauses (see the relevant workshop) or here: https://www.oracletutorial.com/oracle-basics/oracle-check-constraint/).
4. Ensure your script file has appropriate comments in the header, includes the required drop commands and includes echo on and echo off commands.
5. Run your script and create the three required tables.
6. Save the output from this run.
   - To save the output we make use of the inbuilt Oracle SPOOL command, as you did with your Data Modeller generated schemas.
   - To include the result of the SQL commands in the output we make use of the inbuilt Oracle SET ECHO command.

To use SPOOL and SET ECHO, place as the top line in your schema file:

```
set echo on
spool applied7_schema_output.txt
```

and as the last line in your script file

```
spool off
```

```
set echo off
```

This will produce a file, in the same folder that your script is saved in, called applied7_schema_output.txt which contains the full run of your SQL script. Note that the filename you are spooling to must not contain spaces or special characters (use letters, numbers and _ only)

A7-3 Modifying the structure of a database

Create a new sql file, name the file as **applied7_alter.sql** and save this file under the Applied 7 folder in your local repository.

Write the answer to the following tasks, run the script and save the output using the inbuilt Oracle SPOOL command discussed above.

TASKS

1. Add a new column to the UNIT table which will represent credit points for the unit (hint use the ALTER command). The credit points for a unit must be either 3, 6 or 12. The default value should be 6 points.
2. The client wishes to store the course details from this point forward. For each course, the client wants to store its code, name and total credit points.

course_code	course_name	course_totalpoints
C2000	Bachelor of Information Technology	144
C2001	Bachelor of Computer Science	144
C4001	Graduate Certificate of Cybersecurity	24
…		
C6007	Master of Artificial Intelligence	96

- Each course includes several units, and each unit can be part of multiple courses. For example, FIT1047 and FIT1045 are part of C2000 and C2001 courses.
- Change the structure of the database to implement this new requirement. Note that for this activity you do not need to populate the tables with sample data.

Important

You need to get into the habit of establishing the following as your standard workflow:

- if you are working in your individual repo - before modifying any file/s, pull at the start of your working session, work on the activities you wish to/are able to complete during this session, add all(stage), commit changes and then push the changes back to the FIT GitLab server