CTS ETL Methodology

Emma Spielfogel, Jerry Gray

2024-08-06

Table of contents

Preface

This is a Quarto book for documenting the CTS' ETL methodology. This book is currently under development.

To learn more about Quarto books visit https://quarto.org/docs/books.

1 Updating This Quarto Site

This site is developed as a Quarto book.

More information about quarto books can be found here: https://quarto.org/docs/books.

1.1 Steps for updating this site:

1. Create a new Quarto document (.qmd) file containing the information you would like to add
    i. Name this file something with no spaces—use hyphens instead.
    ii. The easiest way to create a new file is to make a copy of any existing one then change the name—this way the same information will be present at the top of each document.
2. Open the **_quarto.yml** file and add the name of this new file under chapters, after whichever existing chapter you would like to go.

> ! **Make sure the _quarto.yml file is updated before proceeding, otherwise the site will not render correctly.**

3. When you are ready to publish, click "Render" at the top of the window of your Quarto document. This will generate the html version of the Quarto document into the "docs" subfolder.
    i. You can also type "quarto render" into the Terminal to render all documents at once.
    ii. To preview the entire book, type "quarto preview" into the Terminal.
4. The site is now updated locally. The next step is to commit these changes to git, then push the changes to GitHub. See the git/Github setup section for detailed instructions.

2 RStudio Setup

Purpose: This section will describe how to set up RStudio for development.

2.1 Using Git and GitHub

Follow the instructions here to set up Git and GitHub. As described in these instructions, using R Studio Projects makes working with Git and GitHub easier.

2.2 Git for Command Line

2.2.1 To clone a repository from GitHub to your RStudio session:

- Copy the link from the GitHub repository you would like to clone
  - Click the green "Code" button to see the link
- Go the Terminal tab in RStudio
- Make sure you are at the location you want to clone to—if not, change your directory using cd
- git clone "URL of repository from GitHub"

2.2.2 To add new files to a git repo then push to GitHub:

- Go the Terminal tab in RStudio
- Open the repository file path if you are not already in the project (use cd in the Terminal to change your directory)
- git pull
  - This will pull the repository your current location in order to ensure you are using the most up-to-date version. This is especially impo
- git status
  - This allows you to check the status—doing this often is a good check.
- git add + file name

- This adds files to git (not GitHub).
- git add -u adds anything that is tracked but has changed (the u stands for updated)

- git status
  - Checking the status again here can be helpful to double-check you have added all files and folders you would like to.
- git commit -m “Commit message”
  - This commits the changes you have made. Everything is easily reversible **until this step.** Once committed, changes are part of the hi
- git push
  - This pushes the committed changes to GitHub.

If you have messed up the branches, you can revert (do with mega caution):

- git reset –hard {insert commit ID}
- git log
- git status
- git push -f
- git status

3 Oracle PL/SQL Beginner's Programming Guide

Oracle is CaSe sensitive! Values stored in the database must be matched exactly. The exception to this is Oracle object names. All objects are stored in upper case, but can be referenced in lower, mixed, or upper case.

3.1 Common Commands

There are a bunch of common commands that can be used in Oracle (or that you may see in SQL) . Here are a few:

DUAL: Dual is a system table variable. It is used when creating data from scratch in a SQL statement. Oracle requires that all Select statements come from a table or view. This enables you to select data that doesn't exist in a physical structure.

example: SELECT 'Hello World' as COLUMN_1 from DUAL; SELECT 3 + 4 as RESULT_1 FROM DUAL;

SYSDATE: SYSDATE is a system variable that will return the date and time of the server. It is a DATE data type.

TRUNC(): This is a useful function. It will truncate the data element/attribute and return the first part of the data.

example: TRUNC(SYSDATE) will return only the date and not the time portion TRUNC(12.345) will return the integer portion of the decimal number (i.e.; 12)

3.2 Block Statements

Block statements are logical groupings of code. There are several types: BEGIN…END; IF…END IF; CASE…END CASE; FOR LOOP…END LOOP; and more.

Statement Termination All code blocks and single line statements MUST end with the semi-colin ";".

3.3 Variables

Variables are defined in the DECLARE section of a PL/SQL block. They can also be defined in Functions, Procedures, and Packages. In general, you should avoid using generic variable names like “A”, “X”, “THIS”, “THAT”, etc. Names should be self-documenting in that the name should mean something to the reader. The exception to this is commonly used ones known in the general coding community. Variables should be consistent in term of Case and Naming Standards to avoid confusion.

Assigning value to a variable is fairly simple. All assignments use the following notation (excluding the quotes): “:=”

For Example: MY_VARIABLE_EXAMPLE := ‘Hello World!’;

3.4 Logical Statements

Statements that can be evaluated to TRUE/FALSE and what to do in each case.

IF Statement

The IF statement is one of the most basic logical constructs of all coding languages. It has the following syntax:

```
IF A=B THEN
   Do something here;
ELSE
    Do something here;
END IF;
```

There is also the ELSIF

```
IF A=B THEN
  Do something here;
ELSIF A=C THEN
  Do something here;
ELSE
  Do something here;
END IF;
```

More information can be found here: https://www.techonthenet.com/oracle/loops/if_then.php

CASE Statement

You can use the CASE statement in both a SELECT statement and in PL/SQL code. The syntax for both are the same except for the line termination. In the SELECT statement, no semicolon is used. In the PL/SQL statement, a semi colon is used after each line following the "Then" and at the end of the Case.

```
Two styles:
  1: CASE MY_VARIABLE
        WHEN 'A' THEN Do Something;
        WHEN 'B' THEN Do Something;
        ELSE Do Something;
     END CASE;

  2: CASE
        WHEN MY_VARIABLE = 'A' THEN Do Something;
        WHEN MY_VARIABLE = 'B' THEN Do Something;
        WHEN MY_VARIABLE = 'C' and
             MY_DATE=TRUNC(SYSDATE) THEN Do Something;
        ELSE Do Something;
     END CASE;

More information can be found here:
  https://www.oracletutorial.com/plsql-tutorial/plsql-case-statement/
```

3.5 Loops

Loops are logical structures that allow the process to repeat a block of code until a condition is met and code exists the loop.

FOR LOOP...END LOOP; The For Loop is the loop that I have used the most and is very versatile. There is the standard use of the for loop and a more advance use that I use.

```
1:  FOR Counter IN 1..20
    Loop
        Do Something;
    End Loop;

2:  FOR MY_DATA IN
        (
            SELECT EMPLID, EMPL_NAME, DEPARTMENT, SALARY, HIRE_DATE
            FROM HR
            WHERE TO_DATE(HIRE_DATE, 'YYYYMM') = '202306'
```

```
        )
    LOOP
        INSERT INTO MONTHLY_REVIEW VALUES (MY_DATA.EMPLID, MYDATA.EMPL_NAME);
        Do Somehting else;
    END LOOP;
```

More information on Loops can be found here:

FOR LOOP:
https://www.techonthenet.com/oracle/loops/for_loop.php
https://www.oracletutorial.com/plsql-tutorial/plsql-loop/

CURSOR FOR Loop:
https://www.techonthenet.com/oracle/loops/cursor_for.php
https://www.oracletutorial.com/plsql-tutorial/plsql-cursor-for-loop/

LOOP:
https://www.techonthenet.com/oracle/loops/gen_loop.php
https://www.oracletutorial.com/plsql-tutorial/plsql-loop/

WHILE LOOP:
https://www.techonthenet.com/oracle/loops/while.php
https://www.oracletutorial.com/plsql-tutorial/plsql-while-loop/

REPEAT UNTIL:
https://www.techonthenet.com/oracle/loops/repeat_until.php

3.6 Procedures and Functions

Procedures and functions are a set of code that can be called and executed. The primary difference between the two are Functions return a value and Procedures do not.

Procedure:

```
create or replace PROCEDURE MY_PROC ( PARAM1 VARCHAR2 )

IS
  --variable declaration section
  v_SEQ_NBR NUMBER := 0;

BEGIN

  VALID SQL or PL/SQL;
```

```
END;
```

Functions:

```
create or replace FUNCTION MY_FUNCTION ( PARAM1 NUMBER, PARAM2 VARCHAR2 ) RETURN VARCHAR

IS
  --variable declaration section
  PRAGMA AUTONOMOUS_TRANSACTION; --Allows for multiple types of actions to occur or be cal

  VARIABLE_LIST   VARCHAR2(50) := ''; --VARCHAR varibles must have a length declared
                                      --the := '' defines the default value of the variabl

  exceptionvar    EXCEPTION;


  --IF CURSOR IS USED
  CURSOR C1 IS
      SELECT *
      FROM MY_TABLE
      WHERE MY_COLUMN = PARAM1;

BEGIN
  VALID SQL or PL/SQL;
  RETURN '0';

--------------------------------------------------------------------------

  EXCEPTION
  WHEN exceptionvar THEN return 'something';

END;
```

In both examples, Parameters are not required. If not needed, omit the section "(..)" completely. Also, if variables are not needed, you can omit them. Cursors are defined in the variable section.

For more information on Procedures and Functions go here:

Procedures:

- https://www.techonthenet.com/oracle/procedures.php
- https://www.oracletutorial.com/plsql-tutorial/plsql-procedure/

Functions:

- https://www.techonthenet.com/oracle/functions.php
- https://www.oracletutorial.com/plsql-tutorial/plsql-function/

4 Connecting to Oracle

Purpose: This section will describe how to connect RStudio and Oracle.

4.1 Oracle account creation

4.2 Oracle sign-in

4.3 Database connections

4.4 Other specific steps...

4.5 Embedding code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

References