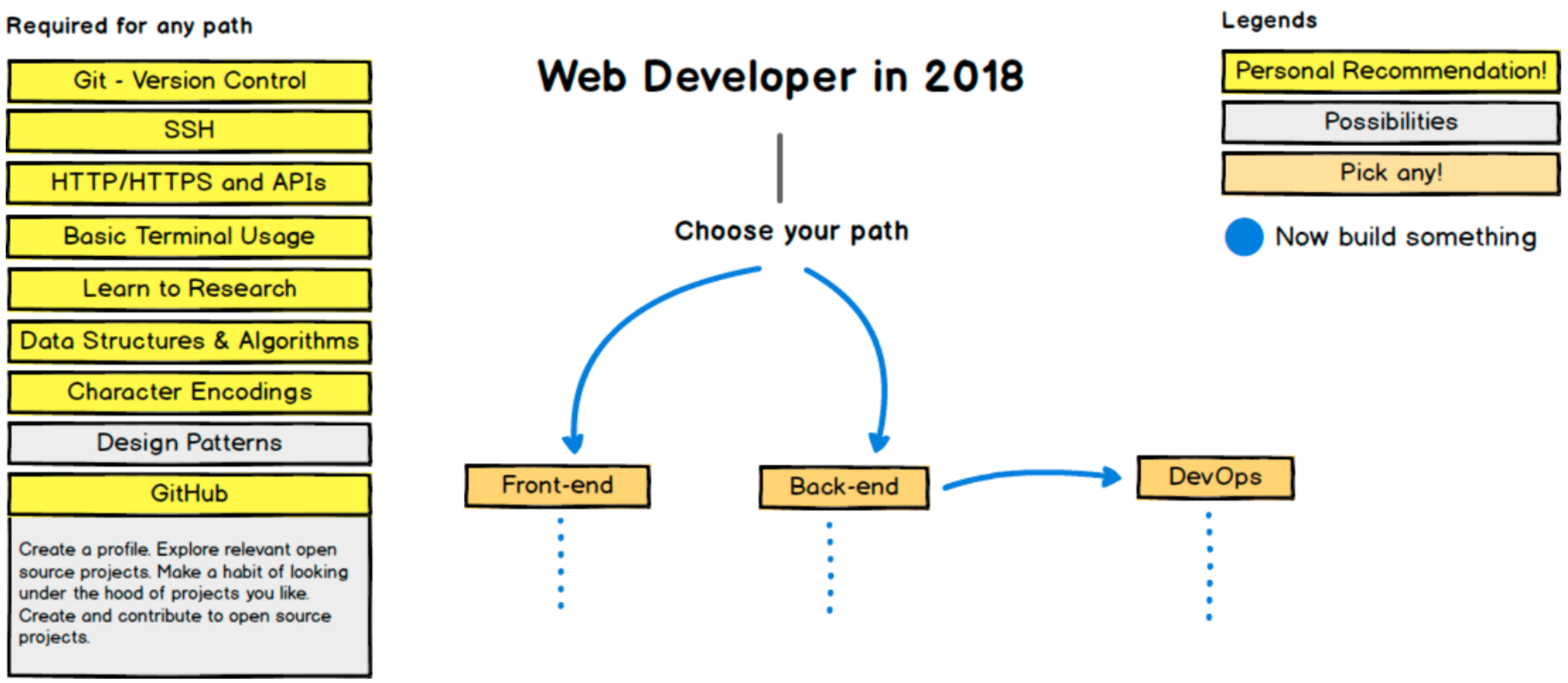
Database administration advanced

Andres Martínez Gutiérrez. 2018-1

http://bit.ly/2CZjZei

🚀 Introduction



Required for any path

- Git - Version Control
- SSH
- HTTP/HTTPS and APIs
- Basic Terminal Usage
- Learn to Research
- Data Structures & Algorithms
- Character Encodings
- Design Patterns
- GitHub

Create a profile. Explore relevant open source projects. Make a habit of looking under the hood of projects you like. Create and contribute to open source projects.

Web Developer in 2018

Choose your path

Front-end

Back-end

DevOps

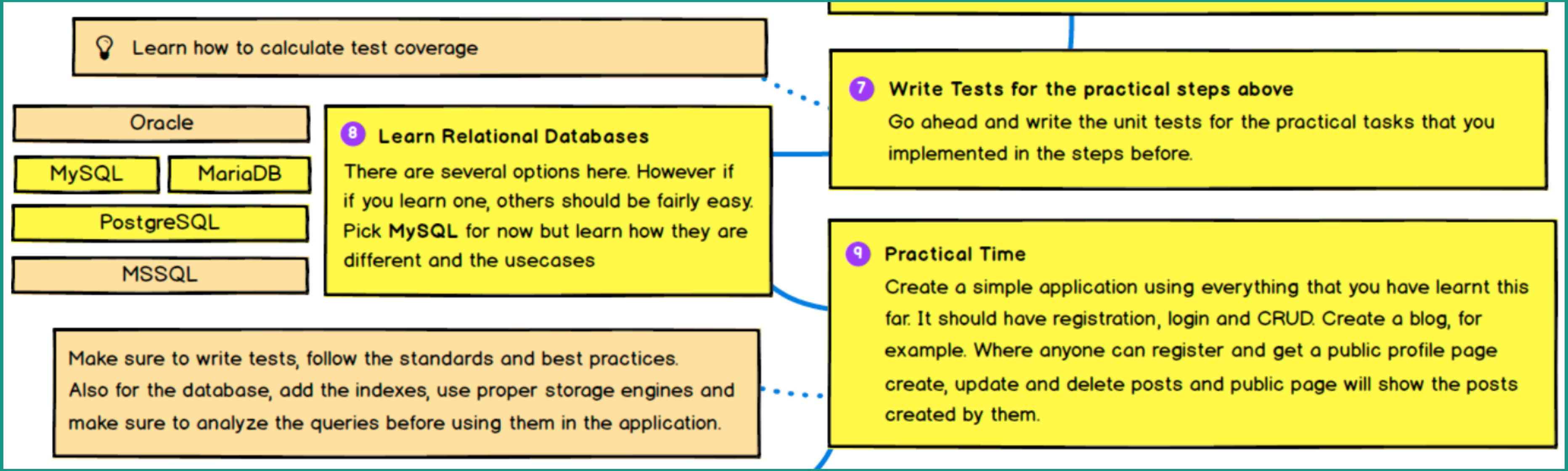
Legends

Personal Recommendation!

Possibilities

Pick any!

● Now build something



Learn how to calculate test coverage

7 Write Tests for the practical steps above

Go ahead and write the unit tests for the practical tasks that you implemented in the steps before.

8 Learn Relational Databases

There are several options here. However if if you learn one, others should be fairly easy. Pick **MySQL** for now but learn how they are different and the usecases

Oracle

MySQL

MariaDB

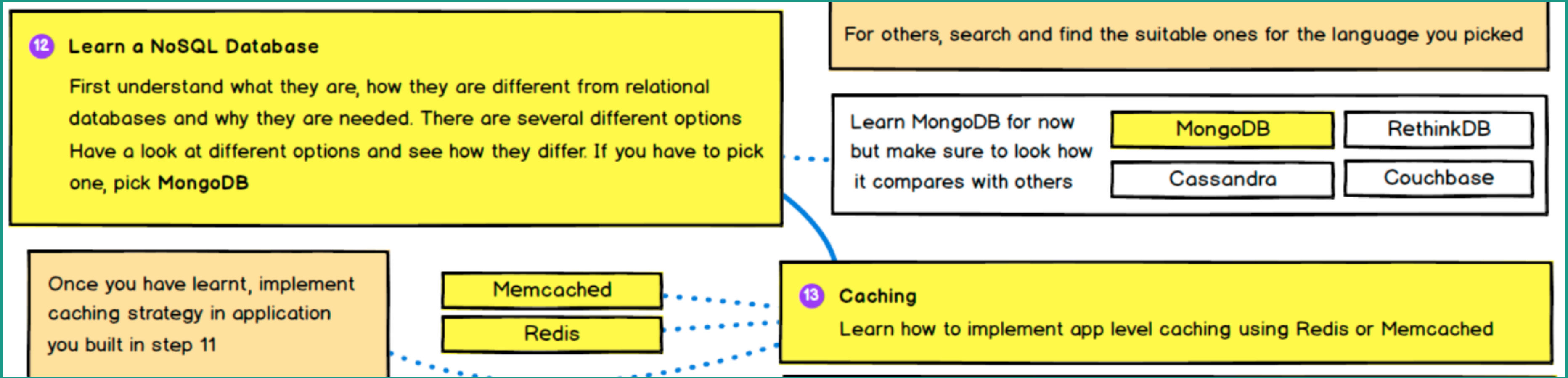
PostgreSQL

MSSQL

9 Practical Time

Create a simple application using everything that you have learnt this far. It should have registration, login and CRUD. Create a blog, for example. Where anyone can register and get a public profile page create, update and delete posts and public page will show the posts created by them.

Make sure to write tests, follow the standards and best practices. Also for the database, add the indexes, use proper storage engines and make sure to analyze the queries before using them in the application.



For others, search and find the suitable ones for the language you picked

12 Learn a NoSQL Database

First understand what they are, how they are different from relational databases and why they are needed. There are several different options Have a look at different options and see how they differ. If you have to pick one, pick **MongoDB**

Learn MongoDB for now but make sure to look how it compares with others

MongoDB

RethinkDB

Cassandra

Couchbase

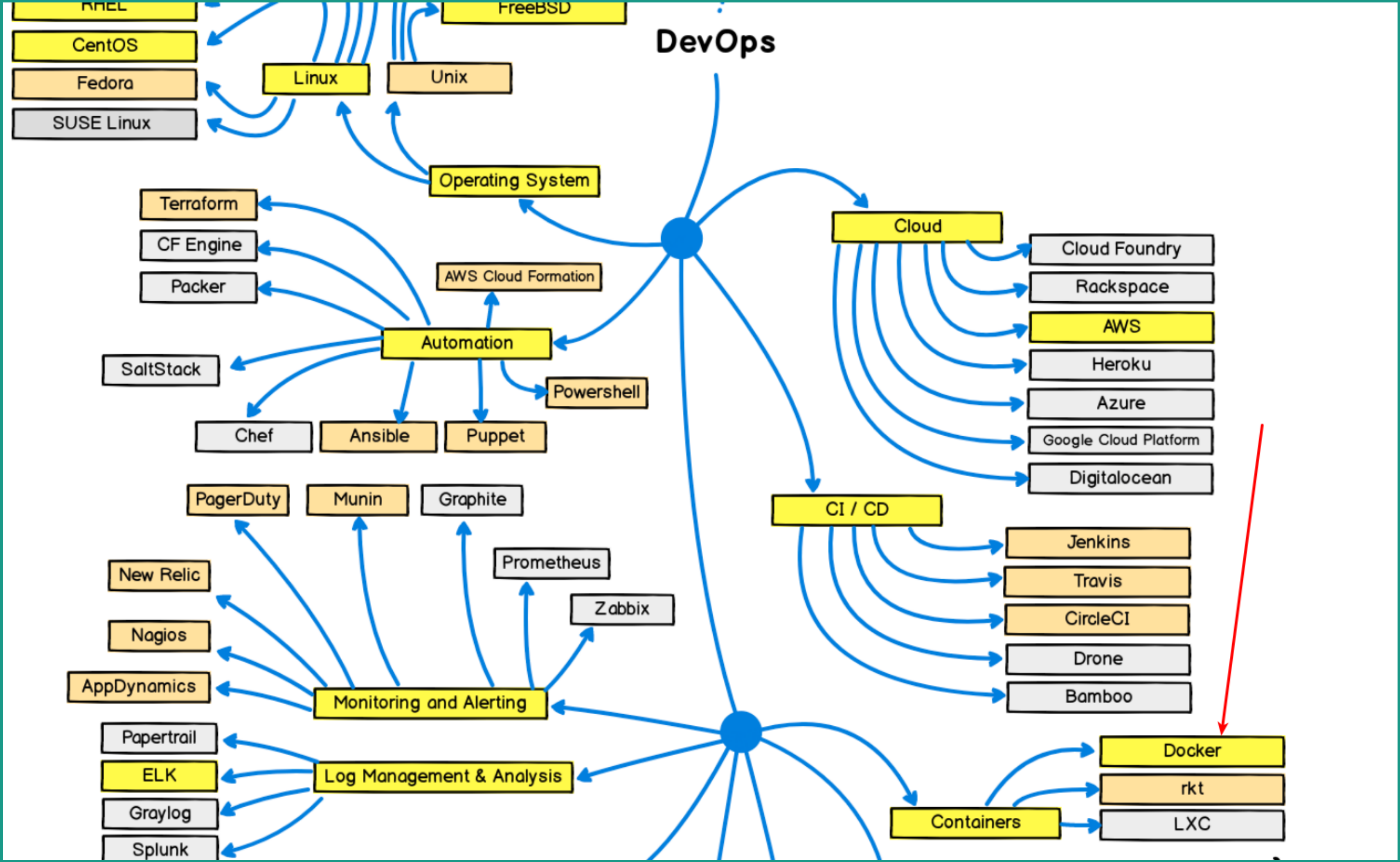
13 Caching

Learn how to implement app level caching using Redis or Memcached

Memcached

Redis

Once you have learnt, implement caching strategy in application you built in step 11


RHEL
CentOS
Fedora
SUSE Linux
Linux
Unix
FreeBSD
DevOps
Operating System
Terraform
CF Engine
Packer
AWS Cloud Formation
Automation
SaltStack
Powershell
Chef
Ansible
Puppet
Cloud
Cloud Foundry
Rackspace
AWS
Heroku
Azure
Google Cloud Platform
Digitalocean
PagerDuty
Munin
Graphite
Prometheus
New Relic
Zabbix
Nagios
AppDynamics
Monitoring and Alerting
CI / CD
Jenkins
Travis
CircleCI
Drone
Bamboo
Papertrail
ELK
Log Management & Analysis
Graylog
Splunk
Docker
rkt
LXC
Containers

https://github.com/kamranahmedse/developer-roadmap

Agenda

1. Arrays
2. Cursors
3. Exceptions

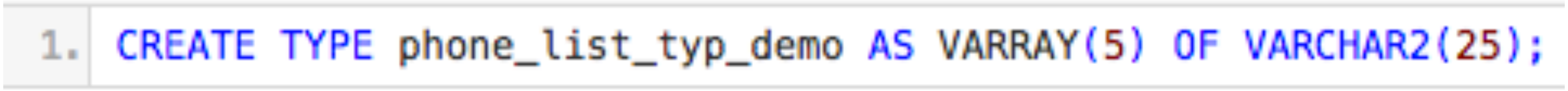
Arrays

Arrays in PL/SQL

```sql
DECLARE
  TYPE varray_type IS VARRAY(5) OF INTEGER;
  TYPE varray_names IS VARRAY(4) OF VARCHAR2(255);
  v2 varray_type;
  names varray_names := varray_names();
BEGIN
  v2 := varray_type(1, 2, 3, 4, 5);
  FOR i IN 1..v2.COUNT LOOP
    DBMS_OUTPUT.put_line(v2(i)**2);
  END LOOP;
  FOR i IN 1..4 LOOP
    names.extend;
    names(i) := ('Andres '||i);
  END LOOP;

  FOR i IN 1..names.COUNT LOOP
    DBMS_OUTPUT.put_line(names(i));
  END LOOP;
END;
```

Arrays in PL/SQL (if you want to share it between functions and procedures)

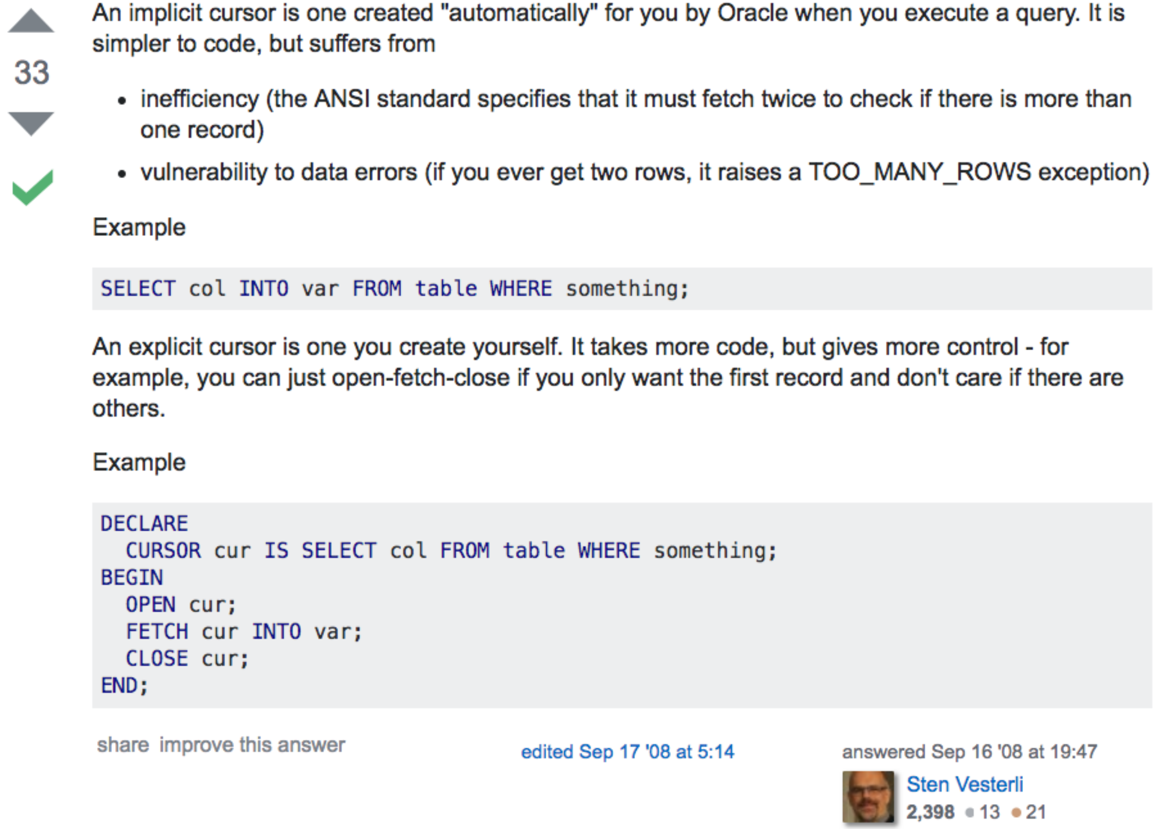


```sql
CREATE TYPE phone_list_typ_demo AS VARRAY(5) OF VARCHAR2(25);
```

https://docs.oracle.com/cloud/latest/db112/LNPLS/create_type.htm#LNPLS01375

Implicit / Explicit Cursors

Implicit /Explicit Cursors



An implicit cursor is one created "automatically" for you by Oracle when you execute a query. It is simpler to code, but suffers from

- inefficiency (the ANSI standard specifies that it must fetch twice to check if there is more than one record)
- vulnerability to data errors (if you ever get two rows, it raises a TOO_MANY_ROWS exception)

Example

```
SELECT col INTO var FROM table WHERE something;
```

An explicit cursor is one you create yourself. It takes more code, but gives more control - for example, you can just open-fetch-close if you only want the first record and don't care if there are others.

Example

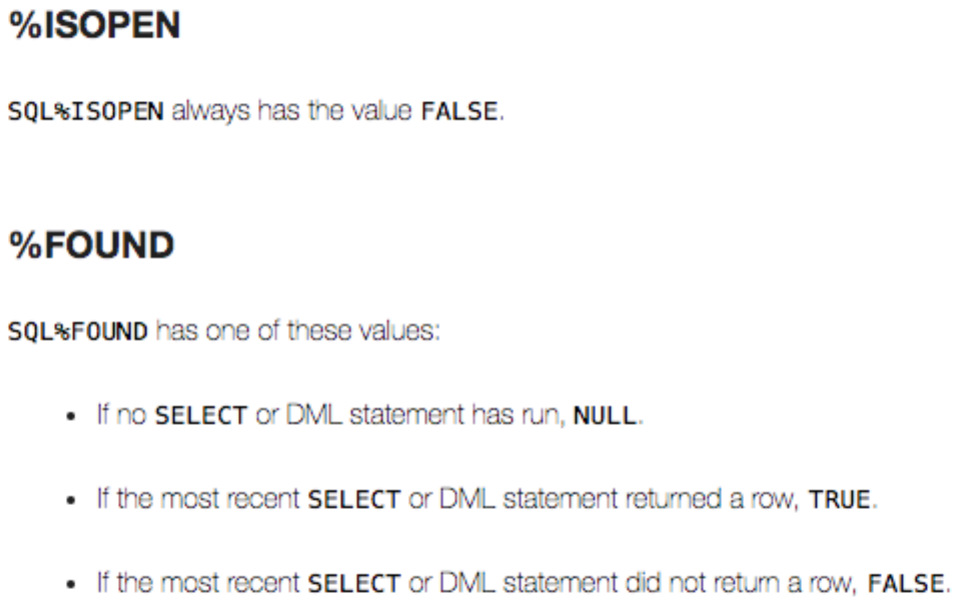
```
DECLARE
  CURSOR cur IS SELECT col FROM table WHERE something;
BEGIN
  OPEN cur;
  FETCH cur INTO var;
  CLOSE cur;
END;
```

edited Sep 17 '08 at 5:14

answered Sep 16 '08 at 19:47

Sten Vesterli
2,398 ● 13 ● 21

Implicit Cursors attributes



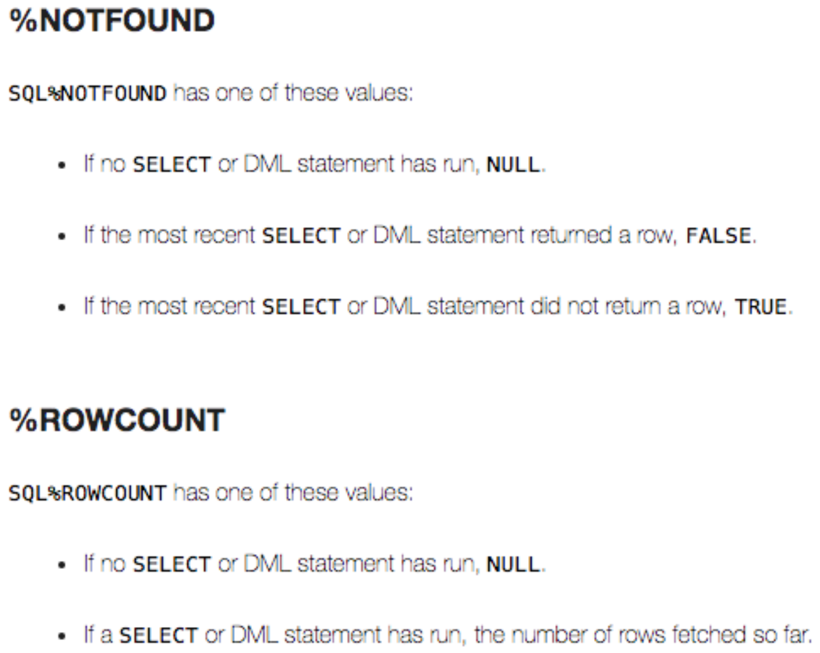
%ISOPEN

SQL%ISOPEN always has the value **FALSE**.

%FOUND

SQL%FOUND has one of these values:

- If no **SELECT** or DML statement has run, **NULL**.
- If the most recent **SELECT** or DML statement returned a row, **TRUE**.
- If the most recent **SELECT** or DML statement did not return a row, **FALSE**.



%NOTFOUND

SQL%NOTFOUND has one of these values:

- If no **SELECT** or DML statement has run, **NULL**.
- If the most recent **SELECT** or DML statement returned a row, **FALSE**.
- If the most recent **SELECT** or DML statement did not return a row, **TRUE**.

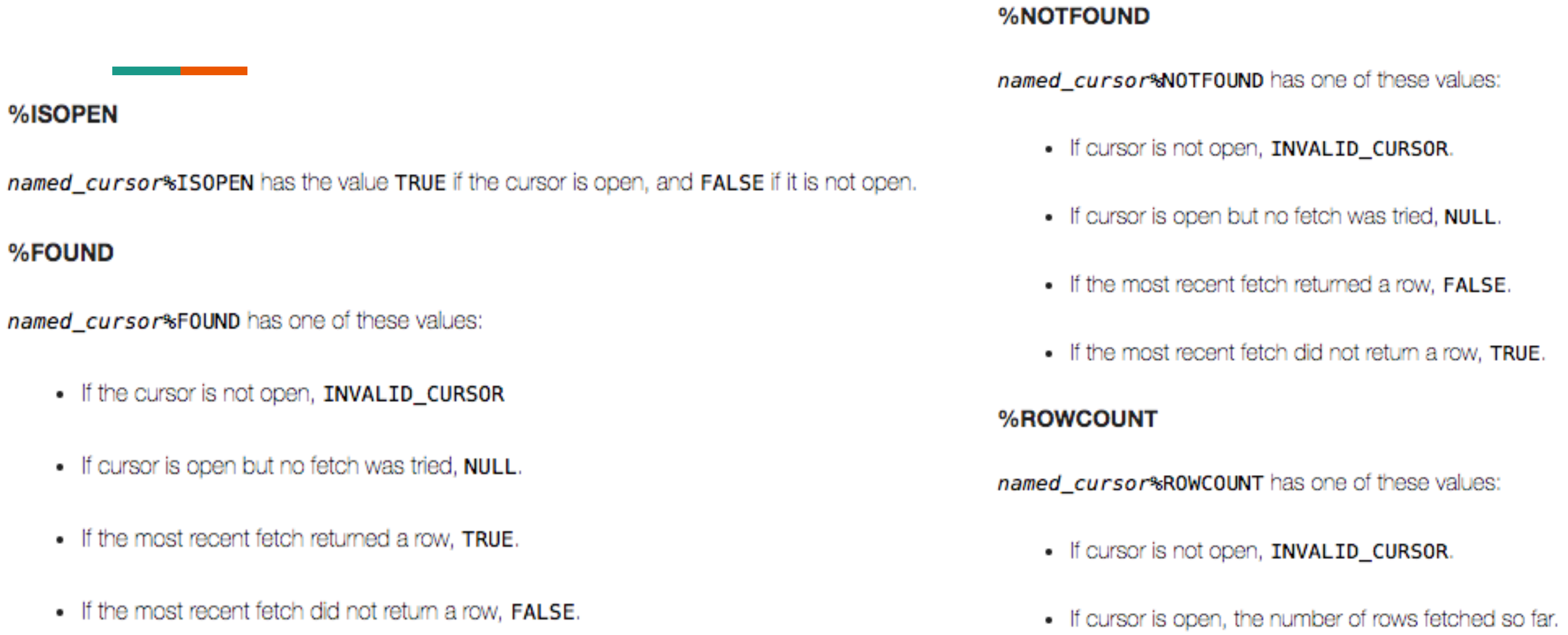
%ROWCOUNT

SQL%ROWCOUNT has one of these values:

- If no **SELECT** or DML statement has run, **NULL**.
- If a **SELECT** or DML statement has run, the number of rows fetched so far.

https://docs.oracle.com/database/122/LNPLS/implicit-cursor-attribute.htm#LNPLS01348

Named Cursors attributes



%ISOPEN

named_cursor%ISOPEN has the value **TRUE** if the cursor is open, and **FALSE** if it is not open.

%FOUND

named_cursor%FOUND has one of these values:

- If the cursor is not open, **INVALID_CURSOR**
- If cursor is open but no fetch was tried, **NULL**.
- If the most recent fetch returned a row, **TRUE**.
- If the most recent fetch did not return a row, **FALSE**.

%NOTFOUND

named_cursor%NOTFOUND has one of these values:

- If cursor is not open, **INVALID_CURSOR**.
- If cursor is open but no fetch was tried, **NULL**.
- If the most recent fetch returned a row, **FALSE**.
- If the most recent fetch did not return a row, **TRUE**.

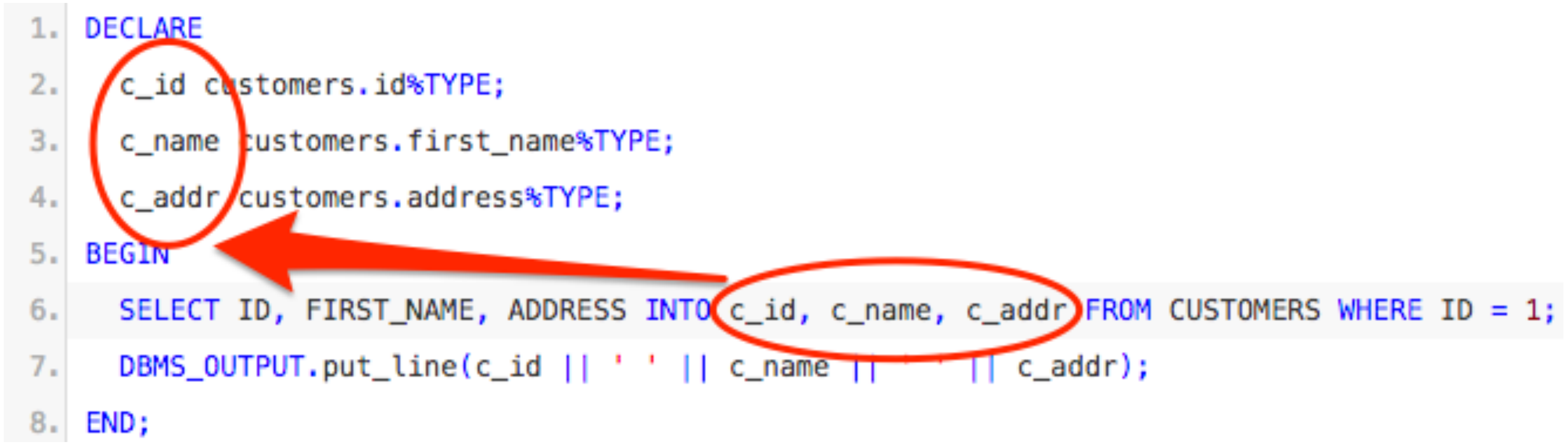
%ROWCOUNT

named_cursor%ROWCOUNT has one of these values:

- If cursor is not open, **INVALID_CURSOR**.
- If cursor is open, the number of rows fetched so far.

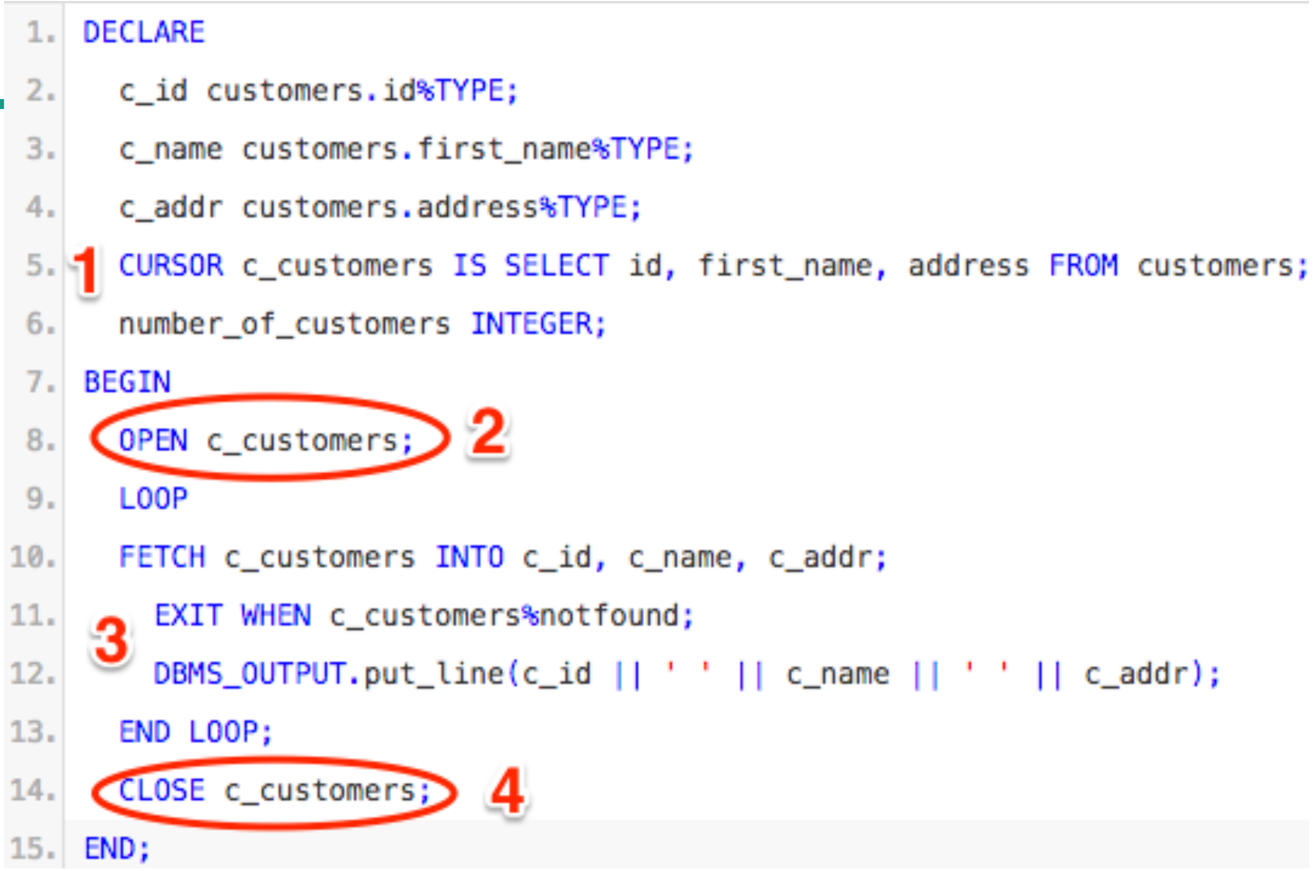
https://docs.oracle.com/database/122/LNPLS/named-cursor-attribute.htm#LNPLS01311

Implicit cursors



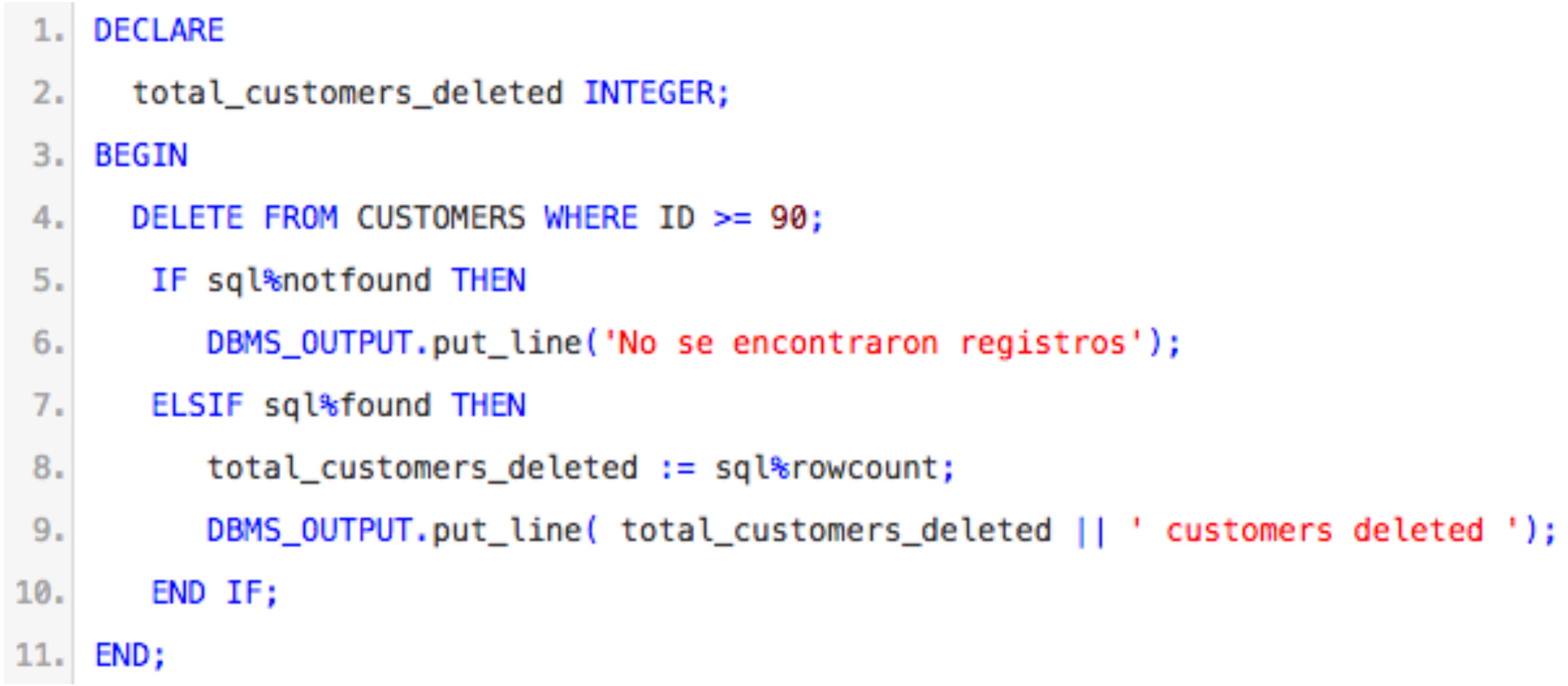
```
DECLARE
  c_id customers.id%TYPE;
  c_name customers.first_name%TYPE;
  c_addr customers.address%TYPE;
BEGIN
  SELECT ID, FIRST_NAME, ADDRESS INTO c_id, c_name, c_addr FROM CUSTOMERS WHERE ID = 1;
  DBMS_OUTPUT.put_line(c_id || ' ' || c_name || ' ' || c_addr);
END;
```

Explicit cursors



```
DECLARE
  c_id customers.id%TYPE;
  c_name customers.first_name%TYPE;
  c_addr customers.address%TYPE;
  CURSOR c_customers IS SELECT id, first_name, address FROM customers;
  number_of_customers INTEGER;
BEGIN
  OPEN c_customers;
  LOOP
  FETCH c_customers INTO c_id, c_name, c_addr;
    EXIT WHEN c_customers%notfound;
    DBMS_OUTPUT.put_line(c_id || ' ' || c_name || ' ' || c_addr);
  END LOOP;
  CLOSE c_customers;
END;
```

Implicit cursors



```sql
DECLARE
  total_customers_deleted INTEGER;
BEGIN
  DELETE FROM CUSTOMERS WHERE ID >= 90;
   IF sql%notfound THEN
      DBMS_OUTPUT.put_line('No se encontraron registros');
   ELSIF sql%found THEN
      total_customers_deleted := sql%rowcount;
      DBMS_OUTPUT.put_line( total_customers_deleted || ' customers deleted ');
   END IF;
END;
```

Explicit Cursors

1. Declare Cursor
2. Open cursor (Mandatory)
3. Do whatever you want / need with the cursor
4. Close cursor (Mandatory)

Exceptions

https://docs.oracle.com/cd/B19306_01/appdev.102/b14261/errors.htm

https://docs.oracle.com/cd/B10500_01/appdev.920/a96624/07_errs.htm

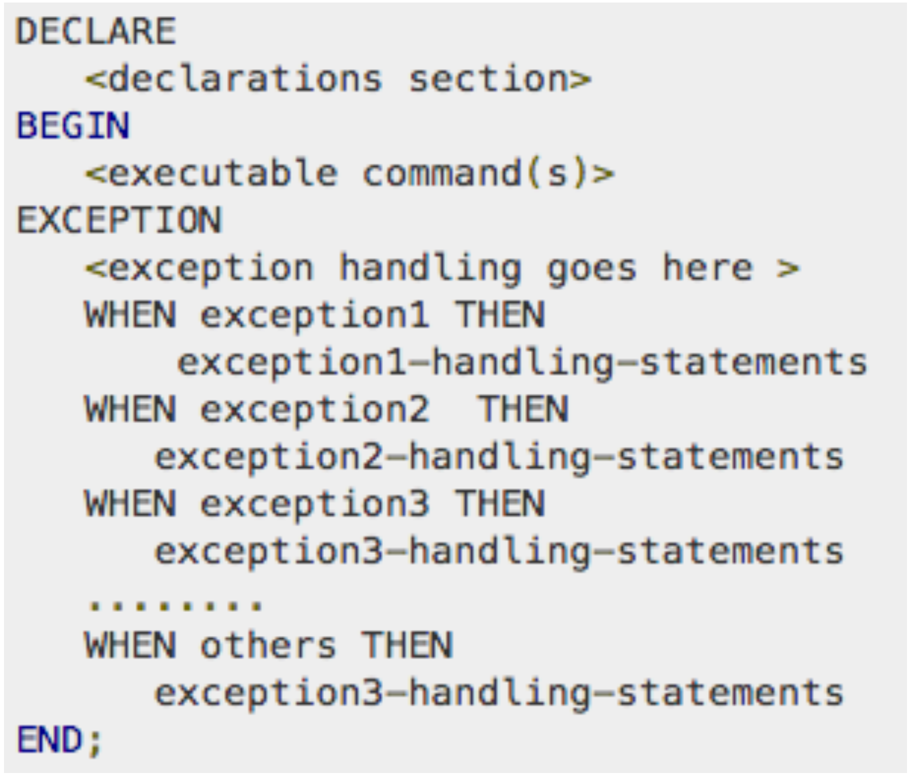
Exceptions

“An error condition during a program execution is called an exception in PL/SQL. PL/SQL supports programmers to catch such conditions using EXCEPTION block in the program and an appropriate action is taken against the error condition”

Exceptions - When to add them?

- Whenever there is any possibility of an error occurring
- Add *error-checking code* whenever you can predict that an error might occur if your code gets bad input data
- Make your programs robust enough to work even if the database is not in the state you expect **(%TYPE)**
- **Handle named exceptions whenever possible, instead of using WHEN OTHERS in exception handlers**

Exceptions

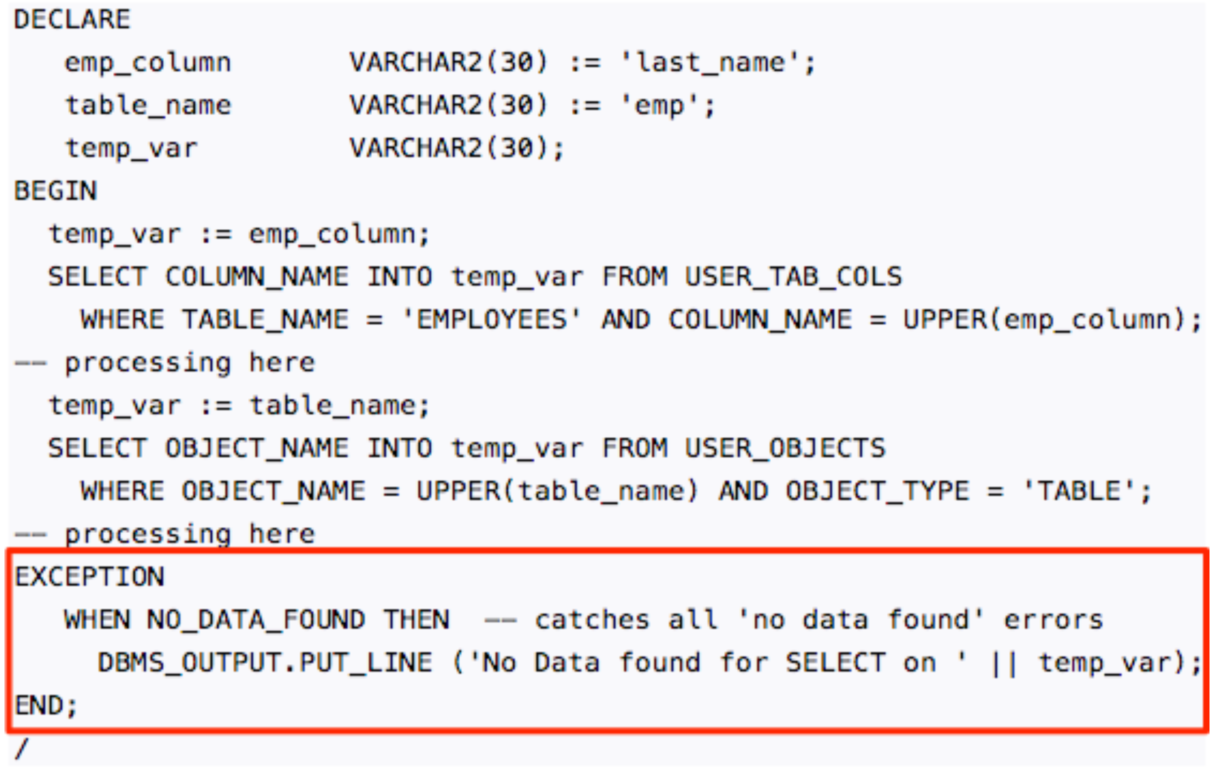


```
DECLARE
   <declarations section>
BEGIN
   <executable command(s)>
EXCEPTION
   <exception handling goes here >
   WHEN exception1 THEN
       exception1-handling-statements
   WHEN exception2  THEN
      exception2-handling-statements
   WHEN exception3 THEN
      exception3-handling-statements
   ........
   WHEN others THEN
      exception3-handling-statements
END;
```

Pre-defined exceptions

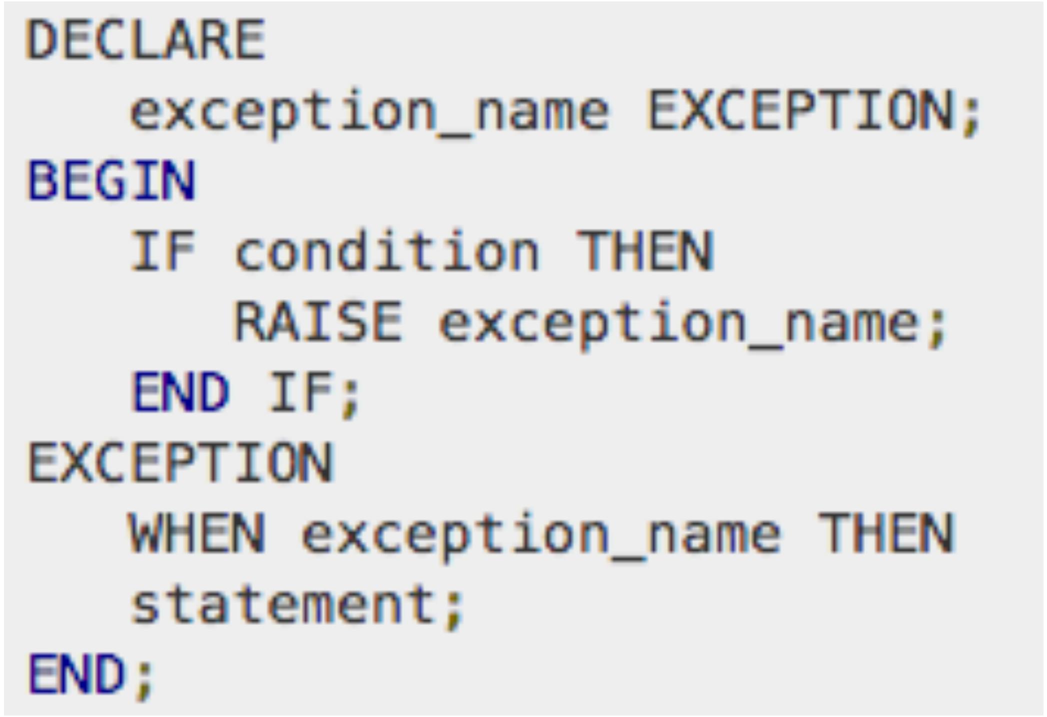
Exception	Oracle Error	SQLCODE Value
ACCESS_INTO_NULL	ORA-06530	-6530
CASE_NOT_FOUND	ORA-06592	-6592
COLLECTION_IS_NULL	ORA-06531	-6531
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	+100
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
SELF_IS_NULL	ORA-30625	-30625
STORAGE_ERROR	ORA-06500	-6500
SUBSCRIPT_BEYOND_COUNT	ORA-06533	-6533
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	-6532
SYS_INVALID_ROWID	ORA-01410	-1410
TIMEOUT_ON_RESOURCE	ORA-00051	-51
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

Exceptions

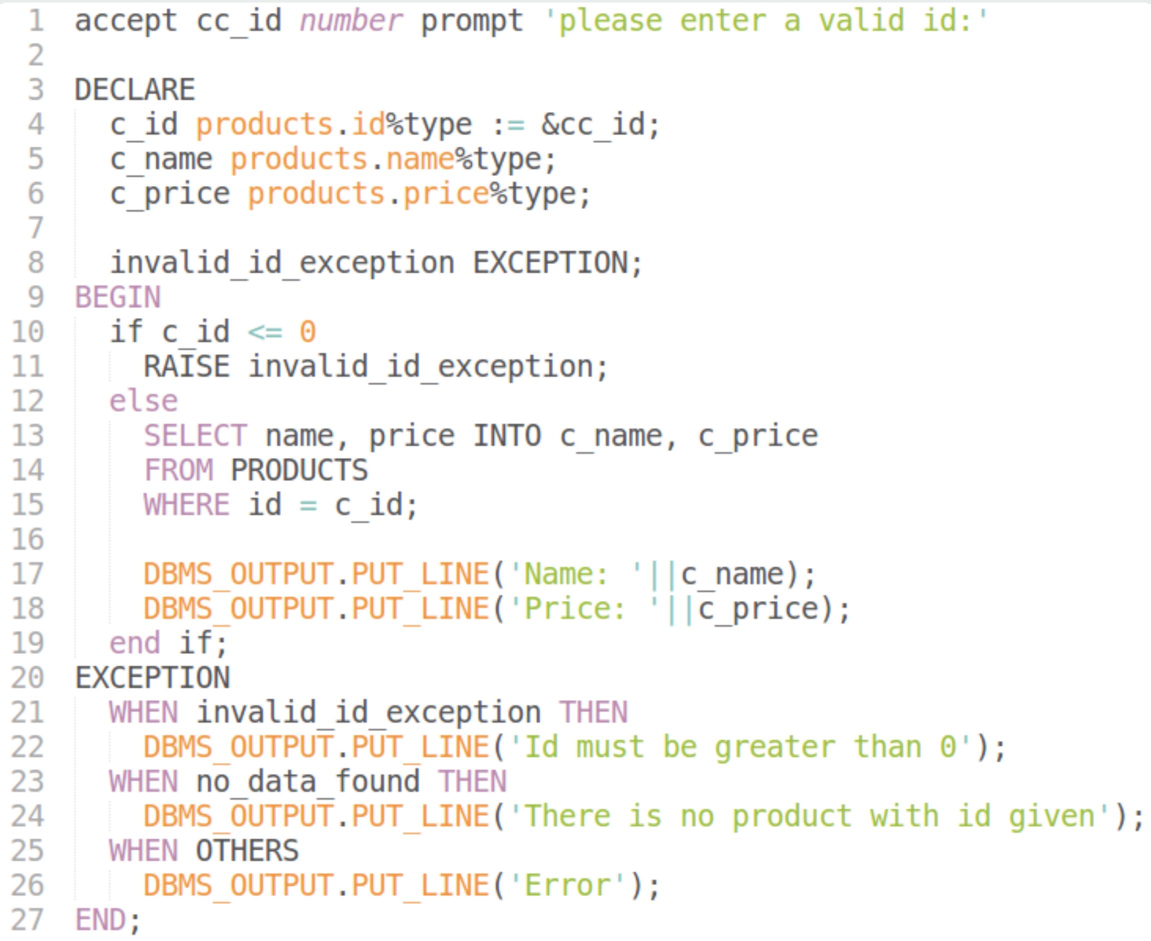


```
DECLARE
   emp_column          VARCHAR2(30) := 'last_name';
   table_name          VARCHAR2(30) := 'emp';
   temp_var            VARCHAR2(30);
BEGIN
  temp_var := emp_column;
  SELECT COLUMN_NAME INTO temp_var FROM USER_TAB_COLS
    WHERE TABLE_NAME = 'EMPLOYEES' AND COLUMN_NAME = UPPER(emp_column);
-- processing here
  temp_var := table_name;
  SELECT OBJECT_NAME INTO temp_var FROM USER_OBJECTS
    WHERE OBJECT_NAME = UPPER(table_name) AND OBJECT_TYPE = 'TABLE';
-- processing here
EXCEPTION
   WHEN NO_DATA_FOUND THEN  -- catches all 'no data found' errors
     DBMS_OUTPUT.PUT_LINE ('No Data found for SELECT on ' || temp_var);
END;
/
```

Raising exceptions



```
DECLARE
   exception_name EXCEPTION;
BEGIN
   IF condition THEN
      RAISE exception_name;
   END IF;
EXCEPTION
   WHEN exception_name THEN
   statement;
END;
```



```sql
accept cc_id number prompt 'please enter a valid id:'

DECLARE
  c_id products.id%type := &cc_id;
  c_name products.name%type;
  c_price products.price%type;

  invalid_id_exception EXCEPTION;
BEGIN
  if c_id <= 0
    RAISE invalid_id_exception;
  else
    SELECT name, price INTO c_name, c_price
    FROM PRODUCTS
    WHERE id = c_id;

    DBMS_OUTPUT.PUT_LINE('Name: '||c_name);
    DBMS_OUTPUT.PUT_LINE('Price: '||c_price);
  end if;
EXCEPTION
  WHEN invalid_id_exception THEN
    DBMS_OUTPUT.PUT_LINE('Id must be greater than 0');
  WHEN no_data_found THEN
    DBMS_OUTPUT.PUT_LINE('There is no product with id given');
  WHEN OTHERS
    DBMS_OUTPUT.PUT_LINE('Error');
END;
```

To practice

https://gist.github.com/amartinezg/d1dffc4349e69f4b573ce895c54a5583

Thank you.