# *Database Management II (420-D20-HR)*

# *Happy Valley Kennels Project*

# *Assignment 2 - Cursors and Exceptions*

Date assigned: Thursday, February 23, 2017

Date due: **Friday, March 10, 2017, 11:50PM**

**Learning Objectives**

After completing this assignment, the student will be able to:

1. Use the PL/SQL looping and selection control structures in a block;
2. Use SQL DML statements in a PL/SQL block;
3. Use implicit cursors;
4. Use an explicit cursor in a PL/SQL block;
5. Use a CURSOR FOR loop;
6. Use the FOR UPDATE option with a Cursor;
7. Handle SQL exceptions.

**To Be Handed in:**

1. The***username***\_**D20\_A02** foldershould be zipped and uploaded to **Moodle**. It should contain:
   1. The ***username***\_**D20\_A02\_Cursors.docx** file containing the SQL source code and the test cases for this assignment
   2. The ***username*\_Self\_Assessment.docx** file.

**To Start:**

1. Rename this document to ***username*\_D20\_A02\_Cursors.docx**. Add your answers and output in the appropriate sections.
2. If you’re unsure of the state of your HVK database, back it up before you start this assignment. You can then rebuild a clean copy by running:
   1. HVK Database Backup\_2017, Add\_VACCINATION\_CHECKED\_STATUS, D20\_A02\_Updates.

**Name all identifiers according to the naming standards shown below.**

**Use ISO/ANSI standard joins in all your SELECT statements.**

**Format all blocks using the SQL Navigator Formatter**

**Include output from program execution**

**Naming Conventions:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Prefix** | **Suffix** |
| local variable | **lv\_** |  |
| local type | **ltyp\_** |  |
| local record | **lrec\_** |  |
| local cursor | **lcur\_** |  |
| parameter variable | **pv\_** |  |
| exception | **e\_** |  |

**Marking and Time management:**

|  |  |  |
| --- | --- | --- |
| Section | Question | Mark |
| 1. Kennel Log | 1 | 28 |
| 1. Add shared reservation | Test case definition | 12 |
|  | Code | 24 |
| English and handed in properly (coding/naming standards, all output messages are meaningful and clear, effort tracking) |  | 6 |
| Totals |  | 70 |

# Explicit Cursors and Control Structures

Purpose: 1. Learn to use cursors to retrieve multiple rows from a database

1. Learn to use cursor attributes to determine the result of a fetch
2. Learn to use a while loop or a basic loop to retrieve rows from a cursor
3. Learn to use a cursor for loop to retrieve rows from a cursor

To Do:

## If you have not already, **D20\_A02\_Updates.sql** in SQL Developer to make updates necessary to test the block in this section.

## Create an anonymous PL/SQL block to create the daily kennel log to be posted on the wall in the kennel.. Note the following:

### The date should be read from the user in the format dd-Mon-yy

### Create four cursors: one for the reservation and pet information, one for the pet's food, one for the extra services to be provided and one for medications. The pet food, extra services and medication cursors should have a parameter for the pet reservation number.

### Use a **cursor for loop** to retrieve the reservations and the food.

### Use a **while loop** or a **basic loop** to retrieve the extra service records and use the other type of loop for the medication records.

### Display a heading line at the beginning with the date.

### For each pet, display the run number, pet name followed by owner last name on one line. Display food brand and variety, frequency (if twice a day), and quantity on the next line. If there are extra services aside from medication, display all the services separated by commas on the next line. If there are any medications to be given, display each medication on a line of its own. Put a blank line between each pet. The output should be in run number sequence.

### If a run has not been assigned, display "unassigned" for the run number.

A sample of the output:

**Sample output:**

**KENNEL LOG FOR 05-Mar-17**

**------------------------**

**Run: 1 Pet: Skarpa Wolfe**

**Food: Iams Mini Chunks Quantity: .25 cup**

**Extra Services: Grooming**

**Run: 1 Pet: Bothvar Wolfe**

**Food: Iams Mini Chunks Quantity: .25 cup**

**Extra Services: Grooming**

**Run: 13 Pet: Willie Que**

**Food: Iams Mini Chunks Quantity:**

**Extra Services: Walk**

**Medication: Medicam Dosage: 48 kg once a day**

**Run: 14 Pet: Sam Morfek**

**Food: Iams Large Dog Quantity: 1 cup**

**Extra Services: Walk, Playtime**

**Run: 14 Pet: Snoop Dogg Morfek**

**Food: Purina Moist Meaty Burger with Cheddar Cheese Burger with Cheddar Cheese Quantity: .5 cup**

**Extra Services: Walk, Playtime**

**Run: 21 Pet: Max Piper**

**Food: Iams Large Dog Quantity: 1 cup**

**Run: 21 Pet: Kitoo Piper**

**Food: Iams Large Dog Quantity: 1 cup**

## Copy the anonymous block below:

**SQL:**

**SET SERVEROUTPUT ON;**

**DECLARE**

**--Test date: 05-Mar-17**

**lv\_log\_date hvk\_kennel\_log.kennel\_log\_date%TYPE := to\_date('&date', 'dd-Mon-yy');**

**CURSOR lcur\_petRes**

**IS SELECT pr.run\_run\_number, p.pet\_name || ' ' || o.owner\_last\_name AS name, pr.pet\_res\_number**

**FROM hvk\_reservation r, hvk\_pet\_reservation pr, hvk\_pet p, hvk\_owner o**

**WHERE o.owner\_number = p.own\_owner\_number**

**AND p.pet\_number = pr.pet\_pet\_number**

**AND r.reservation\_number = pr.res\_reservation\_number**

**AND r.reservation\_start\_date < lv\_log\_date**

**AND r.reservation\_end\_date > lv\_log\_date;**

**CURSOR lcur\_food (pv\_petRes HVK\_PET\_RESERVATION.PET\_RES\_NUMBER%TYPE)**

**IS SELECT DISTINCT f.food\_brand || ' ' || f.food\_variety AS food\_name,**

**pf.pet\_food\_quantity as quantity**

**FROM hvk\_pet\_food pf, hvk\_food f, hvk\_pet\_reservation pr, hvk\_reservation r**

**WHERE pv\_petRes = pf.pr\_pet\_res\_number**

**AND pf.food\_food\_number = f.food\_number;**

**CURSOR lcur\_services (pv\_petRes HVK\_PET\_RESERVATION.PET\_RES\_NUMBER%TYPE)**

**IS SELECT DISTINCT s.service\_description**

**FROM hvk\_service s, hvk\_pet\_reservation\_service prs**

**WHERE pv\_petRes = prs.pr\_pet\_res\_number**

**AND prs.serv\_service\_number = s.service\_number;**

**CURSOR lcur\_meds (pv\_petRes HVK\_PET\_RESERVATION.PET\_RES\_NUMBER%TYPE)**

**IS SELECT DISTINCT m.medication\_name, m.medication\_dosage, m.medication\_special\_instruct,**

**m.medication\_end\_date, m.pr\_pet\_res\_number**

**FROM hvk\_medication m**

**WHERE pv\_petRes = m.pr\_pet\_res\_number;**

**lrec\_service lcur\_services%ROWTYPE;**

**lrec\_med lcur\_meds%ROWTYPE;**

**lv\_services VARCHAR2(100);**

**lv\_meds VARCHAR2(100);**

**BEGIN**

**FOR petRes IN lcur\_petRes LOOP**

**--Run numben, pet name**

**DBMS\_OUTPUT.PUT\_LINE('Run: ' || petRes.run\_run\_number || ' Pet: ' || petRes.name);**

**--Food loop**

**FOR food IN lcur\_food(petRes.pet\_res\_number) LOOP**

**--food name, quantity**

**DBMS\_OUTPUT.PUT\_LINE('Food: ' || food.food\_name || ' Quantity: ' || food.quantity);**

**END LOOP;**

**--Loop for servies**

**OPEN lcur\_services(petRes.pet\_res\_number);**

**FETCH lcur\_services INTO lrec\_service;**

**WHILE lcur\_services%FOUND LOOP**

**IF lrec\_service.service\_description != 'Boarding' THEN**

**IF lv\_services is null THEN**

**lv\_services := 'Extra Services: ' || lrec\_service.service\_description;**

**ELSE**

**lv\_services := lv\_services || ', ' || lrec\_service.service\_description;**

**END IF;**

**END IF;**

**FETCH lcur\_services INTO lrec\_service;**

**END LOOP;**

**CLOSE lcur\_services;**

**--Print out string built up of services**

**DBMS\_OUTPUT.PUT\_LINE(lv\_services);**

**lv\_services := null;**

**--Loop for Medication**

**OPEN lcur\_meds(petRes.pet\_res\_number);**

**LOOP**

**FETCH lcur\_meds INTO lrec\_med;**

**EXIT WHEN lcur\_meds%NOTFOUND;**

**lv\_meds := 'Medication: ' || lrec\_med.medication\_name;**

**IF lrec\_med.medication\_dosage is not null THEN**

**lv\_meds := lv\_meds || ' Dosage: ' || lrec\_med.medication\_dosage;**

**END IF;**

**IF lrec\_med.medication\_special\_instruct is not null THEN**

**lv\_meds := lv\_meds || ' Instructions: ' || lrec\_med.medication\_special\_instruct;**

**END IF;**

**--print out string made up of medication information**

**DBMS\_OUTPUT.PUT\_LINE(lv\_meds);**

**lv\_meds := null;**

**END LOOP;**

**CLOSE lcur\_meds;**

**--New Line**

**DBMS\_OUTPUT.PUT\_LINE('');**

**END LOOP;**

**EXCEPTION**

**WHEN no\_data\_found THEN**

**DBMS\_OUTPUT.PUT\_LINE('No data found');**

**END;**

**Sample output for 05-Mar-17:**

**Run: 1 Pet: Skarpa Wolfe**

**Food: Iams Mini Chunks Quantity: .25 cup**

**Extra Services: Grooming**

**Run: 1 Pet: Bothvar Wolfe**

**Food: Iams Mini Chunks Quantity: .25 cup**

**Extra Services: Grooming**

**Run: 14 Pet: Sam Morfek**

**Food: Iams Large Dog Quantity: 1 cup**

**Extra Services: Walk, Playtime**

**Run: 14 Pet: Snoop Dogg Morfek**

**Food: Purina Moist Meaty Burger with Cheddar Cheese Burger with Cheddar Cheese Quantity: .5 cup**

**Extra Services: Walk, Playtime**

**Run: 21 Pet: Max Piper**

**Food: Iams Large Dog Quantity: 1 cup**

**Run: 21 Pet: Kitoo Piper**

**Food: Iams Large Dog Quantity: 1 cup**

## 

# Exceptions

Purpose: 1. Learn to handle Oracle pre-defined exceptions

1. Learn to create and handle user-defined exceptions
2. Learn to use an if statement to choose alternative actions

To Do:

## Write an anonymous block to add a reservation for two dogs sharing. Note the following:

### Prompt for the reservation start date, the end date and the two pet numbers.

### The start date must be today or later.

### The end date must be after the start date.

### Two dogs who are sharing must have the same owner.

### Use the reservation number sequence for the reservation number.

### When you add a reservation, you must also add a row to the pet reservation and a row to pet services for boarding for each pet. Use the pet reservation sequence for the pet reservation.

### If the reservation add is successful, display a message with the new reservation number, the owner name, the pet names, the start and end dates.

### Either both pets are added or none. You cannot fulfil a partial reservation where only one of the intended pets was successfully booked.

### Handle the following error conditions:

#### The start date occurs after the end date. (Raise an application error)

#### The start date occurs before today. (Raise an application error)

#### The pet number is invalid. (Use an Oracle pre-defined error)

#### Two pets who are sharing do not have the same owner. (Use a user-defined error.)

#### Any other Oracle error.

## Write a test case to test your block that covers all the rules and constraints:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test #** | **Scenario Description** | **Start Date** | **End Date** | **Dog1Id** | **Dog2Id** | **Expected Results** |
| **1** | **Happy path** | **05-Mar-18** | **10-Mar-18** | **1** | **2** | **The reservation was added successfully** |
| **2** | **Missing start date** |  | **10-Mar-18** | **1** | **2** | **The reservation was missing a start date and could not be added.** |
| **3** | **Missing end date** | **05-Mar-18** |  | **1** | **2** | **The reservation was missing an end date and could not be added** |
| **4** | **Missing pet 1** | **05-Mar-18** | **10-Mar-18** |  | **2** | **Only one pet was included in the reservation. Please add a second.** |
| **5** | **Missing pet 2** | **05-Mar-18** | **10—Mar-18** | **1** |  | **Only one pet was included in the reservation. Please add a second.** |
| **6** | **Reservation start date before current date** | **05-Mar-17** | **30-May-17** | **1** | **2** | **Reservation start date cannot be before the current date.** |
| **7** | **Reservation end date before start date** | **10-Mar-18** | **5-Mar-18** | **1** | **2** | **Reservation end date cannot be before the reservation start date.** |
| **8** | **Reservation start date more than 1 year in future** | **5-Mar-18** | **10-Mar-18** | **1** | **2** | **Reservation start date must be within one year from now.** |
| **9** | **Reservation end date more than 6 months after start date** | **5-May-17** | **10-Mar-18** | **1** | **2** | **Reservation must be less than 6 months.** |
| **10** | **Pet 1 and 2 the same** | **5-Mar-18** | **10-Mar-18** | **1** | **1** | **Pet 1 and 2 must be different pets.** |
| **11** | **Pet 1 and 2 have different owners** | **5-Mar-18** | **10-Mar-18** | **1** | **13** | **Pet 1 and 2 must have the same owner.** |
| **12** | **Pet 1 does not exist** | **05-Mar-18** | **10-Mar-18** | **999** | **1** | **One of the pets does not exist** |
| **13** | **Pet 2 does not exist** | **05-Mar-18** | **10-Mar-18** | **1** | **999** | **One of the pets does not exist** |

## Copy the anonymous block below:

**SQL:**

**SET SERVEROUTPUT ON;**

**DECLARE**

**lv\_start\_date hvk\_reservation.reservation\_start\_date%TYPE := to\_date('&start\_date', 'dd-Mon-yy');**

**lv\_end\_date hvk\_reservation.reservation\_end\_date%TYPE := to\_date('&end\_date', 'dd-Mon-yy');**

**lv\_pet\_1 hvk\_pet.pet\_number%TYPE := TO\_NUMBER(NVL('&pet\_number\_1' , 0));**

**lv\_pet\_2 hvk\_pet.pet\_number%TYPE := TO\_NUMBER(NVL('&pet\_number\_2', 0));**

**lv\_owner1 hvk\_owner.owner\_number%TYPE;**

**lv\_owner2 hvk\_owner.owner\_number%TYPE;**

**lv\_owner\_name VARCHAR2(50);**

**lv\_res\_number hvk\_reservation.reservation\_number%TYPE;**

**lv\_pet\_res\_num1 hvk\_pet\_reservation.pet\_Res\_number%TYPE;**

**lv\_pet\_res\_num2 hvk\_pet\_reservation.pet\_Res\_number%TYPE;**

**BEGIN**

**--ERROR HANDLING**

**IF (lv\_start\_date is null) THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Please enter a start date.');**

**ELSIF (lv\_end\_date is null) THEN**

**RAISE\_APPLICATION\_ERROR(-20002, 'Please enter an end date.');**

**ELSIF (lv\_pet\_1 = 0) THEN**

**RAISE\_APPLICATION\_ERROR(-20003, 'Please enter a number for pet 1.');**

**ELSIF (lv\_pet\_2 = 0) THEN**

**RAISE\_APPLICATION\_ERROR(-20004, 'Please enter a number for pet 2.');**

**ELSE**

**IF (lv\_start\_date > lv\_end\_date) THEN**

**RAISE\_APPLICATION\_ERROR(-20005, 'The start date occurs after the end date.');**

**ELSIF (lv\_start\_date < SYSDATE) THEN**

**RAISE\_APPLICATION\_ERROR(-20006, 'The start date occurs before today.');**

**ELSIF (lv\_start\_date > add\_months(SYSDATE, 12)) THEN**

**RAISE\_APPLICATION\_ERROR(-20007, 'The start date must be within 1 year from now.');**

**ELSIF (MONTHS\_BETWEEN(lv\_end\_date, lv\_start\_date) > 6) THEN**

**RAISE\_APPLICATION\_ERROR(-20008, 'The reservation cannot be more than 6 months long.');**

**END IF;**

**IF (lv\_pet\_1 = lv\_pet\_2) THEN**

**RAISE\_APPLICATION\_ERROR(-20009, 'Pleas enter two different pet numbers');**

**ELSE**

**SELECT p1.own\_owner\_number, p2.own\_owner\_number**

**INTO lv\_owner1, lv\_owner2**

**FROM hvk\_pet p1, hvk\_pet p2**

**WHERE p1.pet\_number = lv\_pet\_1**

**AND p2.pet\_number = lv\_pet\_2;**

**IF lv\_owner1 != lv\_owner2 THEN**

**RAISE\_APPLICATION\_ERROR(-20010, 'The two pets do not have the same owner.');**

**END IF;**

**END IF;**

**END IF;**

**--Grab the reservatio\_number to be used in inserts**

**SELECT hvk\_reservation\_seq.NEXTVAL INTO lv\_res\_number**

**FROM dual;**

**--MAKE INSERT INTO RESERVATION**

**INSERT INTO hvk\_reservation (**

**reservation\_number, reservation\_start\_date, reservation\_end\_date**

**) VALUES (**

**lv\_res\_number, lv\_start\_date, lv\_end\_date**

**);**

**IF SQL%FOUND THEN**

**--find the pet\_res\_number for pet1**

**SELECT hvk\_pet\_res\_seq.NEXTVAL**

**INTO lv\_pet\_res\_num1**

**FROM dual;**

**--Find the pet\_res\_number for pet2**

**SELECT hvk\_pet\_res\_seq.NEXTVAL**

**INTO lv\_pet\_res\_num2**

**FROM dual;**

**--INSERT PET 1 INTO PET\_RESERVATION TABLE**

**INSERT INTO hvk\_pet\_reservation (**

**pet\_res\_number, pet\_pet\_number, res\_reservation\_number,**

**run\_run\_number, pr\_sharing\_with**

**) VALUES (**

**lv\_pet\_res\_num1, lv\_pet\_1, lv\_res\_number,**

**null, null**

**);**

**--INSERT PET\_RES 1 INTO PET\_RESERVATION\_SERVICE TABLE**

**IF SQL%FOUND THEN**

**INSERT INTO hvk\_pet\_reservation\_service (**

**SERVICE\_FREQUENCY, PR\_PET\_RES\_NUMBER, SERV\_SERVICE\_NUMBER**

**) VALUES (**

**null, lv\_pet\_res\_num1, 1**

**);**

**END IF;**

**--INSERT PET 2 INTO PET\_RESERVATION TABLE**

**INSERT INTO hvk\_pet\_reservation (**

**pet\_res\_number, pet\_pet\_number, res\_reservation\_number,**

**run\_run\_number, pr\_sharing\_with**

**) VALUES (**

**lv\_pet\_res\_num2, lv\_pet\_2, lv\_res\_number,**

**null, lv\_pet\_res\_num1**

**);**

**IF SQL%FOUND THEN**

**--Update pet\_res 1 so that it's sharing with is set properly**

**UPDATE hvk\_pet\_reservation**

**SET pr\_sharing\_with = lv\_pet\_res\_num2**

**WHERE pet\_res\_number = lv\_pet\_res\_num1;**

**--INSERT PET\_RES 2 INTO PET\_RESERVATION\_SERVICE TABLE**

**INSERT INTO hvk\_pet\_reservation\_service (**

**service\_frequency, pr\_pet\_res\_number, serv\_service\_number**

**) VALUES (**

**null, lv\_pet\_res\_num2, 1**

**);**

**--Print out all the data for the reservation added.**

**SELECT o.owner\_first\_name || ' ' || o.owner\_last\_name into lv\_owner\_name**

**FROM hvk\_owner o**

**WHERE o.owner\_number = lv\_owner1;**

**DBMS\_OUTPUT.PUT\_LINE('Reservation ' || lv\_res\_number**

**|| ' has been successfully added for ' || lv\_owner\_name || '.');**

**DBMS\_OUTPUT.PUT\_LINE('The reservation is from ' || lv\_start\_date**

**|| ' to ' || lv\_end\_date || '.');**

**END IF;**

**END IF;**

**DBMS\_OUTPUT.PUT\_LINE('');**

**EXCEPTION**

**WHEN no\_data\_found THEN**

**DBMS\_OUTPUT.PUT\_LINE('One of the pets does not exist');**

**WHEN INVALID\_NUMBER THEN**

**DBMS\_OUTPUT.PUT\_LINE('Please enter a valid pet number.');**

**WHEN VALUE\_ERROR THEN**

**DBMS\_OUTPUT.PUT\_LINE('Please enter a valid pet number.');**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('An unknown error occured.');**

**END;**

**Update the table below with your test case and results:**

## 

|  |  |  |  |
| --- | --- | --- | --- |
| **Test #** | **Test Passed (Y/N)** | **Expected Results** | **Actual Results** |
| **1** | **Y** | **Reservation xxxx has been successfully added for owner\_name.**  **The reservation is from 18-03-05 to 18-03-10.** | **Reservation 2007 has been successfully added for Jane Smith.**  **The reservation is from 18-03-05 to 18-03-10.** |
| **2** | **Y** | **Insert never occurs, tells user to enter start date** | **Insert never occurs, tells user to enter start date** |
| **3** | **Y** | **Insert never occurs, tells user to enter end date** | **Insert never occurs, tells user to enter end date** |
| **4** | **Y** | **Insert never occurs, tells user to enter pet 1** | **Insert never occurs, tells user to enter pet 1** |
| **5** | **Y** | **Insert never occurs, tells user to enter pet 2** | **Insert never occurs, tells user to enter pet 2** |
| **6** | **Y** | **Insert never occurs, tells user to enter start date that’s after current date** | **Insert never occurs, tells user to enter start date that’s after current date** |
| **7** | **Y** | **Insert never occurs, tells user to enter start date that’s before end date** | **Insert never occurs, tells user to enter start date that’s before end date** |
| **8** | **Y** | **Insert never occurs, tells user to enter start date that’s less than 1 year in the future** | **Insert never occurs, tells user to enter start date that’s less than 1 year in the future** |
| **9** | **Y** | **Insert never occurs, tells user to enter end date less than 6 months after start date** | **Insert never occurs, tells user to enter end date less than 6 months after start date** |
| **10** | **Y** | **Insert never occurs, tells user to enter different pet 1 and 2** | **Insert never occurs, tells user to enter different pet 1 and 2** |
| **11** | **Y** | **Insert never occurs, tells user to enter 2 pets with the same owner** | **Insert never occurs, tells user to enter 2 pets with the same owner** |
| **12** | **Y** | **Insert never occurs, tells user pet doesn’t exist** | **Insert never occurs, tells user pet doesn’t exist** |
| **13** | **Y** | **Insert never occurs, tells user pet doesn’t exist** | **Insert never occurs, tells user pet doesn’t exist** |

# To Finish

## Complete the **Assignment\_Self\_Evaluation.docx** form..