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

# *Assignment 4 – MS SQL & CRUD Optimization*

Date assigned: Tuesday April 04, 2017

Date due: **Tuesday, April 18, 2017, 11:50PM**

**1628 words OG**

**Learning Objectives**

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

1. Create and populate a database in SQL Server
2. Perform CRUD on a SQL Server database using SQL.
3. Analyze the costs and benefits of various optimization techniques

**To Be Handed in:**

* 1. The completed ***username***\_**D20\_A04\_MSSQL\_Optimization.docx** file containing your responses.

**To Start:**

1. Rename this document to ***username***\_**D20\_A04\_MS\_SQL\_CRUD\_Optimization.docx**. Add your answers and output in the appropriate sections.

**Marking:**

|  |  |  |
| --- | --- | --- |
| Section | Question | Mark |
| 1. MS SQL Server | 1. Migration – data type table | 6 |
|  | 1. Migration SQL Create script | 16 |
|  | 1. Queries | 8 |
|  | 1. Insert, Update | 8 |
|  | 1. Delete | 4 |
| 1. qCRUD analysis | CRUD Matrix | 12 |
| 1. CRUD Assessment and Optimization | 5 x 4 | 20 |
| English and handed in properly, assessment |  | 6 |
| Totals |  | 80 |

# MS SQL Server Database Migrations and Creation

Purpose: Create a MS SQL server database, and perform CRUD functions on it.

Reference: See [here](https://docs.microsoft.com/en-us/sql/t-sql/queries/queries) for details and syntax from Microsoft.

Handy [Tutorial](http://www.quackit.com/sql_server/tutorial/) is also provided.

Read [here](https://www.mssqltips.com/sqlservertip/1061/using-identity-insert-to-keep-sql-server-table-keys-in-sync/) on how to deal with identity columns when migrating databases.

See Appendix III for the database schema.

To Do:

## Take a look at the movieTablesDDL\_and\_DataSubset.sql file. This is an Oracle SQL file.

Without using any automated tools, your goal is to get this database onto SQL Server.

The goal is to learn the differences in commands, syntax and data types between the two environments.

Your database is to be called username\_D20A04\_Movies. When you are done, leave it in SQL Server for marking.

Back it up (Using Tasks->Generate scripts) and save it as an embedded zip file here: ->



Hints:

You can create the database strictly from SQL, or a mix of UI and SQL. It’s the end result we’re looking for. My approach was to do one table manually. Right Click on the new table, “Script Table as-> CREATE to”, to see how SQL SERVER SQL generates tables, then I edited the rest of “create table” in Oracle script to make the other tables in SQL Server.

To see samples of MS SQL and the creation of a database, use Tasks->Generate scripts) to see an existing database and see the SQL syntax for defining and populating

Maintain the key and field value (null, not-null) constraints.

Use the Identity property for all the primary numeric keys.

Fill out the following table on mapping Oracle Data types to SQL Server types. This must include all the types that you converted/migrated:

|  |  |  |  |
| --- | --- | --- | --- |
| Oracle Data Type | SQL Server Data type | Oracle sample (from create table) | SQL Server Sample  (from create table) |
| NUMBER(1, 0) | TINYINT | MOVIE\_GENRE\_NO NUMBER(1,0) | MOVIE\_GENRE\_NO TINYINT |
| NUMBER(2, 0) | TINYINT | NO\_AWARDS\_NOMINATED NUMBER(2,0) | NO\_AWARDS\_NOMINATED TINYINT |
| NUMBER(3, 0) | SMALLINT | DIRECTOR\_NO NUMBER(3,0) | DIRECTOR\_NO SMALLINT |
| NUMBER(4, 0) | SMALLINT | BIRTH\_YEAR NUMBER(4,0) | BIRTH\_YEAR SMALLINT |
| NUMBER(5, 2) | DECIMAL(5,2) | MOVIE\_GENRE\_FINE NUMBER(5,2) | MOVIE\_GENRE\_FINE DECIMAL(2) |
| VARCHAR2(30) | VARCHAR(30) | LASTNAME VARCHAR2(30) | LASTNAME VARCHAR(30) |
| VARCHAR2(50) | VARCHAR(50) | MOVIE\_GENRE\_DESCRIPTION VARCHAR2(50) | MOVIE\_GENRE\_DESCRIPTION VARCHAR(50) |

## Provide the SQL for the following queries (also show your output).

List the full name of each director and the number of movies for each director. Show in most to least movies.

SQL:

SELECT d.firstname + ' ' + d.lastname AS Director,

COUNT(m.movie\_no) AS [Number of movies]

FROM MM\_DIRECTOR d, MM\_MOVIE m

WHERE d.director\_no = m.dir\_director\_no

GROUP BY d.firstname + ' ' + d.lastname

ORDER BY COUNT(m.movie\_no) desc;

Output:

Alfred Hitchcock 6

Otto Preminger 4

Stanley Kramer 3

Stanley Kubrick 3

Woody Allen 3

John Ford 3

Cecil B. De Mille 2

Gary Winick 2

Robert De Niro 1

List the occupation type (Director or Actor), name and age of all directors and stars who are still alive. Sort the output by age.

SQL:

SELECT d.firstname + ' ' + d.lastname AS Director,

YEAR(GETDATE()) - d.birth\_year AS Age,

'Director' AS Occupation

FROM MM\_DIRECTOR d WHERE d.death\_year is null

union

SELECT s.firstname + ' ' + s.lastname AS Director,

YEAR(GETDATE()) - s.birth\_year AS Age,

'Actor' AS Occupation

FROM MM\_MOVIE\_STAR s WHERE s.death\_year is null

ORDER BY Age;

Output:

Matt Damon 47 Actor

Julia Roberts 50 Actor

Steven Soderbergh 54 Director

Gary Winick 56 Director

George Clooney 56 Actor

Diane Keaton 71 Actor

Malcolm McDowell 74 Actor

Robert De Niro 74 Director

Keir Dullea 78 Actor

Tippi Hedren 82 Actor

Woody Allen 82 Actor

Woody Allen 82 Director

Kim Novak 84 Actor

Rod Taylor 87 Actor

Eva Marie Saint 88 Actor

## Add yourself as a movie star. Add yourself as starring in an existing movie.

SQL:

Insert into MM\_MOVIE\_STAR (STAR\_NO,LASTNAME,FIRSTNAME,BIRTH\_PLACE,BIRTH\_YEAR,DEATH\_YEAR) values (31,'Philip','Dumaresq','Ottawa',1998,2015);

Insert into MM\_MOVIE\_MOVIE\_STAR (MOV\_MOVIE\_NO,MS\_STAR\_NO) values (27,31);

Output:

STAR\_NO LASTNAME FIRSTNAME BIRTH\_PLACE BIRTH\_YEAR DEATH\_YEAR

31 Philip Dumaresq Ottawa 1998 2015

MOV\_MOVIE\_NO MS\_STAR\_NO

27 31

## Delete all movies directed by Woody Allen.

SQL:

DECLARE

@lv\_fname varchar(25);

SET @lv\_fname='Woody';

DECLARE

@lv\_lname varchar(25);

SET @lv\_lname='Allen';

BEGIN

DELETE FROM MM\_MOVIE\_MOVIE\_STAR

WHERE mov\_movie\_no = ANY(

SELECT MOVIE\_NO FROM MM\_MOVIE

WHERE dir\_director\_no = (

SELECT d.director\_no

FROM MM\_DIRECTOR d

WHERE d.firstname = @lv\_fname AND d.lastname = @lv\_lname

)

);

DELETE FROM MM\_MOVIE

WHERE DIR\_DIRECTOR\_NO = (

SELECT d.director\_no

FROM MM\_DIRECTOR d

WHERE d.firstname = @lv\_fname AND d.lastname = @lv\_lname

);

END

Output:

(5 row(s) affected)

(3 row(s) affected)

MOV\_MOVIE\_NO MS\_STAR\_NO

2 3

2 4

3 5

4 6

4 7

5 8

6 9

6 10

8 11

8 12

9 13

10 14

10 15

12 8

12 17

13 14

14 16

15 17

15 18

16 8

17 14

17 19

18 20

19 21

20 22

20 23

21 24

22 8

22 26

23 25

23 26

24 27

25 28

26 16

26 30

27 29

## Logout, log back in, were your changes in the above questions persistent? Explain when SQL Server performed the commits. (You may need to research the transaction functions and when SQL Server commits).

Answer:

Yes, my changes were persistent. SQL Server’s default is to auto-commit any changes once a transaction has completed successfully. Should a transaction fail, it will automatically rollback, thought these defaults can be overridden.

# CRUD Matrix

## Back to Oracle and the HVK database.

## Create a CRUD Matrix for the Check in process.

## At check in, the clerk performs

### Confirm there is a reservation given the pet’s name and today’s date.

### Lookup the vaccination status of the pet to confirm all vaccinations are up to date.

Record the results of your analysis below. (You may add or remove tables/colums as you see fit, I’ve just provided a template as a starting point):

| **Table/column** | **Dog checkin for reservation** |
| --- | --- |
| hvk\_owner |  |
| .owner\_number [PK] |  |
| .owner\_last\_name |  |
| .owner\_first\_name |  |
| .owner\_street\_address |  |
| .owner\_city |  |
| .owner\_province |  |
| .owner\_postal\_code |  |
| .owner\_phone |  |
| .owner\_email |  |
| hvk\_pet |  |
| .pet\_number [PK] | dR(4) |
| .own\_owner\_number [FK] |  |
| .pet\_name |  |
| .dog\_size |  |
| hvk\_reservation |  |
| .reservation\_number [PK] | r |
| .reservation\_start\_date | sR(1) |
| .reservation\_end\_date |  |
| .reservation\_status |  |
| hvk\_pet\_reservation |  |
| .pet\_res\_number [PK] |  |
| .pet\_pet\_number [FK] | sR(3) |
| .res\_reservation\_number [FK] | sR(2) |
| .run\_run\_number [FK] |  |
| .pr\_sharing\_with [FK] |  |
| hvk\_pet\_reservation\_service |  |
| .pr\_pet\_res\_number [PK] [FK] |  |
| .serv\_service\_number [PK] [FK] |  |
| .service\_frequency |  |
| hvk\_service |  |
| .service\_number [PK] |  |
| .service\_description |  |
| hvk\_daily\_rate |  |
| .daily\_rate\_number [PK] |  |
| .daily\_rate |  |
| .serv\_service\_number [FK] |  |
| .daily\_rate\_dog\_size |  |
| hvk\_contract |  |
| .contract\_number [PK] |  |
| .res\_reservation\_number [FK] |  |
| .contract\_date |  |
| hvk\_vaccination |  |
| .vaccination\_number [PK] | dR(6) |
| .vaccination\_type | r |
| hvk\_pet\_vaccination |  |
| .pet\_pet\_number [PK] [FK] | dR(5) |
| .vacc\_vaccination\_number [PK] [FK] | r |
| .vaccination\_expiry\_date | r |
| .vaccination\_is\_validated | r |
| hvk\_run |  |
| .run\_number [PK] |  |
| .run\_size |  |
| hvk\_medication |  |
| .medication\_number [PK] |  |
| .pr\_pet\_res\_number [FK] |  |
| .medication\_name |  |
| .medication\_dosage |  |
| hvk\_pet\_food |  |
| .food\_food\_number [PK][FK] |  |
| .pr\_pet\_res\_number [PK] [FK] |  |
| .pet\_food\_frequency |  |
| .pet\_food\_quantity |  |
| hvk\_food |  |
| .food\_number [PK] |  |
| .food\_brand |  |
| .food\_variety |  |
| hvk\_reservation\_discount |  |
| .disc\_discount\_number [PK][FK] |  |
| .res\_reservation\_number [PK] [FK] |  |
| hvk\_pet\_reservation\_discount |  |
| .disc\_discount\_number [PK][FK] |  |
| .pr\_pet\_res\_number [PK] [FK] |  |
| hvk\_discount |  |
| .discount\_number [PK] |  |
| .discount\_description |  |
| .discount\_type |  |
| .discount\_percentage |  |
| Type of Process (**B**atch, **O**nline) | **O**nline |
| Frequency (times per week) | 1.5 |
| **Totals:** |  |
| Tables Created | 0 |
| Tables Read | 5 |
| Tables Updated | 0 |
| Tables Deleted | 0 |

# CRUD Assessment and Optimization

## Analyze the costs and benefits of the following optimizations relative to the processes in the CRUD.

### Add run\_size to hvk\_pet\_reservation table

### Add an alternate key (index) for owner\_email to the hvk\_owner table

### Add a new column for total charge to the hvk\_pet\_service table.

### Subdivide the hvk\_reservation table into hvk\_completed\_reservation and hvk\_reservation

### Add a direct relationship from hvk\_pet\_service to hvk\_daily\_rate.

Record the results of your analysis below:

1. **Optimization:** Add run\_size to hvk\_pet\_reservation table

***Processes improved:***

Booking a run will be improved

***How it improves performance:***

This would improve performance by minimizing the amount the amount that you have to go to the run table. For pets that already have booked runs, you can check a little more quickly how many large runs are available by just looking at the pet\_reservations in the time frame using large runs.

***Costs:***

The cost of this is pretty small, you just have to add a single column to the pet\_reservation table that is likely to never change. However, should HVK go through renovations, it means having inconsistent data by showing the wrong sized pets in different runs.

***Your recommendation:***🞏 I recommend this optimization

🞏 I do not recommend this optimization

***Justification:***

The cost of going to the run table is minimal, since there’s only 13 records to look through in there. Considering you’d need to change a lot of data should hvk undergo some changes, plus you have so few reservations with reserved runs and there’s only a small portion of the query that goes towards checking the run in the pet\_reservation table against it’s size in the run table. The benefit is really quite minimal, and not really worth the extra work.

1. **Optimization:** Add an alternate key (index) for customer\_email to the hvk\_customer table

***Processes improved:***

This will improve the process of logging on for a customer, and searching for a customer by email address.

***How it improves performance:***

This improves performance by making it so you no longer need to do a sequential read through the owner table for an email address, you can now do that automatically. This will mean that it’s a direct read every time you want to log on.

***Costs:***

The cost is very small, you’re just adding a constraint on a column, so there’s no extra maintenance involved. The keys will take up space, so should HVK grow, they could end up taking a lot of storage. You just need to have validation on the Owner table to make sure there’s no two owners with the same email address. Which that validation should be there anyways.

***Your recommendation:***🞏 I recommend this optimization

🞏 I do not recommend this optimization

***Justification:***

It greatly improves the process of logging into the system and allows clerks to search by email much more quickly, while taking minimal storage space for the time being since HVK is a small system. It’s also a much more frequent transaction than modifying the Owner table.

1. **Optimization:** Add a new column for total charge to the hvk\_pet\_service table

***Processes improved:***

*Generating an invoice for the customer*

***How it improves performance:***

This would improve the performance having a much smaller calculation to do when pricing the reservation for the customer. Instead of referencing the DailyRate table every time you want to print an invoice to get the amount, it’s all already stored, and you just need to add.

***Costs:***

Every time the Owner makes a change to the reservation, this cost needs to be recalculated.

***Your recommendation:***🞏 I recommend this optimization

🞏 I do not recommend this optimization

***Justification:***

If you just make the calculation when printing out the invoice, then you will only have to make this calculation once. However, if you chose to make this a column of its own, the calculation will still need to be made at least once when adding the row, and then it will need to be recalculated every time a customer chooses to change the reservation information, so the way it is, it will only get calculated once, whereas this has potential to get done many more times.

1. **Optimization:** Subdivide the hvk\_reservation table into hvk\_completed\_reservation and hvk\_reservation

***Processes improved:***

It would improve anything where you want to see upcoming or active reservations.

***How it improves performance:***

Any time you want to see upcoming or active reservations, you have a much smaller pool of data to be looking though now.

***Costs:***

It means that now you have two similar pieces of data that you have maintain and organize the flow of data from one table to another. You have the same data spread across two tables as well now, which isn’t efficient.

***Your recommendation:***🞏 I recommend this optimization

🞏 I do not recommend this optimization

***Justification:***

With this optimization, you have a lot less data to dig through for anything upcoming. Reservations that have ended are likely to never be touched much, so why dig through all of that extra data when looking at upcoming or active reservations. Filtering through reservations is one of the transactions that will happen the most often, so to drop the size of what you’re looking through from 65 to 10, or in a real system from several thousand to maybe 50, that would significantly increase performance. And the benefit of this change will become more and more significant the longer the system is around.

1. **Optimization:** Add a direct relationship from hvk\_pet\_service to hvk\_daily\_rate

***Processes improved:***

Calculating costs when generating an invoice

***How it improves performance:***

It doesn’t really improve performance. If you need to join the two tables together, they both contain the service number, which is really all you need. You would probably never go through the service table anyways, unless you searched by the service description.

***Costs:***

*It means that you just have another dependency between tables. There’s not likely going to be much of a need to go through the service table most of the time anyways, so there’s just another dependency that exists.*

***Your recommendation:***🞏 I recommend this optimization

🞏 I do not recommend this optimization

***Justification:***

Increasing the amount of dependencies across table for a transaction that is barely improved at all. Searching by service description is just as easy since you’re linking the tables together the exact same way you would be either way.

# Assessment

1. What did you learn in completing this assignment?

I learned about CRUD matrixes and thinking about just how many tables often needed to be crossed to find a single, relatively simple, piece of data, and then how that can often be simplified by optimizing the database to improve its performance when making frequent calculations.

1. What did you have difficulty with?

I had the most difficulty with the CRUD matrix. I know which data needs to be accessed and in what order, it’s just a matter of making sure I have the right reeds in the tables and the right numbering on them.

1. What did you do well?

I think I answered all the questions about optimization pretty well. I took my time on then almost a half-page for each question, so I feel like I went pretty in depth about it.

1. How many hours did you spend in completing this assignment?

This whole assignment took me about 6 hours. It took about 3 hours for translating the SQL, then another 2 hours to answer all the optimization questions, and another hour for the CRUD matrix.

1. What took you the most time?

Translating the Oracle SQL into SQL server took the longest time. It was challenging, because there’s some things that Oracle allows that SQL server doesn’t tolerate, and it meant re-ordering a lot of the script. It wasn’t that hard getting the right syntax everywhere, the hard part was really getting it to all work together by shuffling around the altering of the tables in the script.

Appendix I – Samples

**Exhibit 1.1:** Sample Run of the Daily Kennel Logquery

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Enter value for date: 12-Nov-2007 | | | |  |
| Run  ----- | Dog  ----------------- | Service  ----------- | Food  ----------------------------------------------------- | Medication  -------------------------------- |
| 35 | "Pete" Piper | Boarding | Pedigree Choice Cuts in Sauce Country Stew Once a day |  |
| 35 | "Pete" Piper | Boarding | Iams Weight Control Once a day |  |
| 35 | "Pete" Piper | Walk |  |  |
| 36 | "Max" Piper | Boarding | Own Food Once a day |  |
| 36 | "Max" Piper | Walk |  |  |
| 36 | "Max" Piper | Grooming |  |  |
| 36 | "Kitoo" Piper | Boarding | Own Food Once a day |  |
| 36 | "Kitoo" Piper | Walk |  |  |
| 36 | "Kitoo" Piper | Grooming |  |  |
| 29 | "Scrabble" Smith | Boarding | Science Diet Allergy Formula Twice a day |  |
| 29 | "Scrabble" Smith | Walk |  |  |
| 29 | "Scrabble" Smith | Medication |  | 1 tablet Prednisone twice daily |
| 29 | "Archie" Smith | Boarding | Iams Mini Chunks Twice a day |  |
| 29 | "Archie" Smith | Walk |  |  |

**Exhibit 1.2**: Sample Boarding Invoice

|  |  |
| --- | --- |
| Happy Valley Kennels  Your pet’s home away from home  25 Wonder Dog Lane  Gatineau, QC J8Y1T3  Phone (819) 555-1234 Fax (819) 555-9342 | INVOICE |
| Invoice #1  Date: Nov. 19, 2014 |

|  |  |
| --- | --- |
| To:  Jane Smith  202 Poodle Path,  Gatineau, QC J8A1R2  (819) 555-1111 | Pet Boarding and related services For:  Scrabble and Archie  Boarding period:  12-Nov-2014 to 19-Nov-2014 |

|  |  |  |  |
| --- | --- | --- | --- |
| DESCRIPTION | DAYS/FREQUENCY | RATE | AMOUNT |
| Boarding – Archie | 7 | $11.00 | $77.00 |
| - 10% discount for shared run |  |  | -$7.70 |
| Walk – Archie | 7 | $3.00 | $21.00 |
| Boarding – Scrabble | 7 | $10.00 | $70.00 |
| - 10% discount for shared run |  |  | -$7.00 |
| Walk – Scrabble | 7 | $2.00 | $14.00 |
| Medication – Scrabble | 7 | $1.00 | $7.00 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | sub-Total | | $174.30 |
|  | GST | | $8.72 |
|  | TOTAL | | $183.02 |
| Make all checks payable to Happy Valley Kennels  Total due in 15 days. Overdue accounts subject to a service charge of 1% per month. | | | |
| Thank you for your business! | | | |

Appendix II

Data-to-Process CRUD Matrix

| **Table/column** | **Dog reservation process** | **Print daily kennel log** | **End visit** |
| --- | --- | --- | --- |
| hvk\_owner |  |  |  |
| .owner\_number [PK] | R | dR (4) | dR (4) |
| .owner\_last\_name | R | R | R |
| .owner\_first\_name | R |  | R |
| .owner\_street\_address |  |  | R |
| .owner\_city |  |  | R |
| .owner\_province |  |  | R |
| .owner\_postal\_code |  |  | R |
| .owner\_phone |  |  | R |
| .owner\_email | sR (1) |  |  |
| hvk\_pet |  |  |  |
| .pet\_number [PK] | R  dR (7) | dR (3) | dR (3) |
| .own\_owner\_number [FK] | dR (2) | R | R |
| .pet\_name |  | R | R |
| .dog\_size | R |  | R |
| hvk\_reservation | sR (5)  C (9) | sR(1) |  |
| .reservation\_number [PK] | R | R | dR (1) |
| .reservation\_start\_date | R | R | R |
| .reservation\_end\_date | R | R | R |
| .reservation\_status | R | R | U (12) |
| hvk\_pet\_reservation | C (10) |  |  |
| .pet\_res\_number [PK] |  | R | R |
| .pet\_pet\_number [FK] | R | R | R |
| .res\_reservation\_number [FK] | dR (6) | dR(2) | dR(2) |
| .run\_run\_number [FK] |  | R |  |
| .pr\_sharing\_with [FK] | R |  |  |
| hvk\_pet\_reservation\_service | C (12) |  |  |
| .pr\_pet\_res\_number [PK] [FK] |  | dR (5) | dR (5) |
| .serv\_service\_number [PK] [FK] |  | R | R |
| .service\_frequency |  | R | R |
| hvk\_service | sR(11) |  |  |
| .service\_number [PK] | R | dR (6) | dR (6) |
| .service\_description | R | R | R |
| hvk\_daily\_rate |  |  |  |
| .daily\_rate\_number [PK] |  |  |  |
| .daily\_rate |  |  | R |
| .serv\_service\_number [FK] |  |  | dR (7) |
| .daily\_rate\_dog\_size |  |  | R |
| hvk\_contract |  |  |  |
| .contract\_number [PK] |  |  |  |
| .res\_reservation\_number [FK] |  |  |  |
| .contract\_date |  |  |  |
| hvk\_vaccination | sR (3) |  |  |
| .vaccination\_number [PK] | R |  |  |
| .vaccination\_type |  |  |  |
| hvk\_pet\_vaccination |  |  |  |
| .pet\_pet\_number [PK] [FK] | dR (4) |  |  |
| .vacc\_vaccination\_number [PK] [FK] | dR (4) |  |  |
| .vaccination\_expiry\_date | R |  |  |
| .vaccination\_is\_validated | R |  |  |
| hvk\_run | sR (8) |  |  |
| .run\_number [PK] |  |  |  |
| .run\_size | R |  |  |
| hvk\_medication |  |  |  |
| .medication\_number [PK] |  |  |  |
| .pr\_pet\_res\_number [FK] |  | dR (9) |  |
| .medication\_name |  | R |  |
| .medication\_dosage |  | R |  |
| hvk\_pet\_food |  |  |  |
| .food\_food\_number [PK][FK] |  | R |  |
| .pr\_pet\_res\_number [PK] [FK] |  | dR (7) |  |
| .pet\_food\_frequency |  | R |  |
| .pet\_food\_quantity |  | R |  |
| hvk\_food |  |  |  |
| .food\_number [PK] |  | dR (8) |  |
| .food\_brand |  | R |  |
| .food\_variety |  | R |  |
| hvk\_reservation\_discount |  |  |  |
| .disc\_discount\_number [PK][FK] |  |  | R |
| .res\_reservation\_number [PK] [FK] |  |  | dR (10) |
| hvk\_pet\_reservation\_discount |  |  |  |
| .disc\_discount\_number [PK][FK] |  |  | R |
| .pr\_pet\_res\_number [PK] [FK] |  |  | dR (8) |
| hvk\_discount |  |  |  |
| .discount\_number [PK] |  |  | dR (9)  dR (11) |
| .discount\_description |  |  | R |
| .discount\_type |  |  | R |
| .discount\_percentage |  |  | R |
| Type of Process (**B**atch, **O**nline) | O | B | O |
| Frequency (times per week) | 70 | 7 | 70 |
| **Totals:** |  |  |  |
| Tables Created | 3 | 0 | 0 |
| Tables Read | 9 | 9 | 10 |
| Tables Updated | 0 | 0 | 1 |
| Tables Deleted | 0 | 0 | 0 |

Appendix III

