Ali Nawaz Maan
Student ID: 44688813

RELATIONAL DATABASES

ORACLE SQL ASSIGNMENT

DOGGO PARADISE

Ali Nawaz Maan
Student ID: 44688813

TASK 1: DATABASE CONSTRAINTS

A: ALL CONSTRAINTS ON DATABASE

```
SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME, SEARCH_CONDITION,
INDEX_NAME FROM USER_CONSTRAINTS;
```

SQL Plus

```
SQL> SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME, SEARCH_CONDITION,INDEX_NAME FROM USER_CONSTRAINTS;

OWNER
 CONSTRAINT_NAME
  TABLE_NAME
   SEARCH_CONDITION
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------
INDEX_NAME
--------------------------------------------------------------------------------
C##S4468881
 FK_SERVICE_NAME
  SERVICE_HISTORY_DETAIL


C##S4468881
 FK_SH_STORE_ID
  SERVICE_HISTORY


C##S4468881
 FK_DOG_BREED
```



```
INDEX_NAME
--------------------------------------------------------------------------------
C##S4468881
 NN_PRICE
  SERVICES
   PRICE IS NOT NULL



13 rows selected.

Elapsed: 00:00:01.94
SQL>
```

B: MISSING CONSTRAINTS

Ali Nawaz Maan
Student ID: 44688813

```
ALTER TABLE CUSTOMERS
ADD CONSTRAINT PK_CUSTOMERS  PRIMARY KEY (C_ID);

ALTER TABLE DOGS ADD CONSTRAINT FK_C_ID FOREIGN KEY (C_ID)
REFERENCES CUSTOMERS (C_ID);

ALTER TABLE DOGS MODIFY DOG_NAME VARCHAR2(50) NOT NULL;

ALTER TABLE CUSTOMERS MODIFY DOB DATE NOT NULL;

ALTER TABLE SERVICES MODIFY PRICE NUMBER(4,2) NOT NULL;

ALTER TABLE SERVICE_HISTORY ADD CONSTRAINT  CK_FINISHED
CHECK ((FINISHED = 'T') OR (FINISHED = 'F'));

ALTER TABLE CUSTOMERS ADD CONSTRAINT  CK_DOB
CHECK (EXTRACT(YEAR FROM DOB) < 1999);

ALTER TABLE SERVICE_HISTORY_DETAIL ADD CONSTRAINT  CK_START_TIME_END_TIME
CHECK (END_TIME > START_TIME);

ALTER TABLE SERVICE_HISTORY_DETAIL ADD CONSTRAINT  CK_SERVICE_DATE
CHECK (EXTRACT(YEAR FROM END_TIME) < 2018);
```

```
SQL> ALTER TABLE CUSTOMERS ADD CONSTRAINT PK_CUSTOMERS  PRIMARY KEY (C_ID);

Table altered.

SQL> ALTER TABLE DOGS ADD CONSTRAINT FK_C_ID FOREIGN KEY (C_ID)REFERENCES CUSTOMERS (C_ID);

Table altered.

SQL> ALTER TABLE DOGS MODIFY DOG_NAME VARCHAR2(50) NOT NULL;

Table altered.

SQL> ALTER TABLE CUSTOMERS MODIFY DOB DATE NOT NULL;

Table altered.

SQL> ALTER TABLE SERVICES MODIFY PRICE NUMBER(4,2) NOT NULL;

Table altered.

SQL> ALTER TABLE SERVICE_HISTORY ADD CONSTRAINT  CK_FINISHED CHECK ((FINISHED = 'T') OR (FINISHED = 'F'));

Table altered.

SQL> ALTER TABLE CUSTOMERS ADD CONSTRAINT  CK_DOB CHECK (EXTRACT(YEAR FROM DOB) < 1999);

Table altered.

SQL> ALTER TABLE SERVICE_HISTORY_DETAIL ADD CONSTRAINT  CK_START_TIME_END_TIME  CHECK (END_TIME > START_TIME);

Table altered.

SQL> ALTER TABLE SERVICE_HISTORY_DETAIL ADD CONSTRAINT  CK_SERVICE_DATE  CHECK (EXTRACT(YEAR FROM END_TIME) < 2018);

Table altered.

SQL>
```

TASK 2: TRIGGERS

A: SEQ_CUSTOMER AND TR_CUSTOMER_ID

```
CREATE SEQUENCE "SEQ_CUSTOMER" MINVALUE 10000 MAXVALUE 999999999999
INCREMENT BY 1 START WITH 10000;
```

Ali Nawaz Maan
Student ID: 44688813

```
CREATE OR REPLACE TRIGGER "TR_CUSTOMER_ID"
BEFORE INSERT ON "CUSTOMERS"
FOR EACH ROW
BEGIN
 SELECT "SEQ_CUSTOMER".NEXTVAL INTO :NEW.C_ID FROM DUAL;
END;
/
```

```
SQL> CREATE SEQUENCE "SEQ_CUSTOMER" MINVALUE 10000 MAXVALUE 999999999999
  2  INCREMENT BY 1 START WITH 10000;

Sequence created.

SQL>
SQL> CREATE OR REPLACE TRIGGER "TR_CUSTOMER_ID"
  2  BEFORE INSERT ON "CUSTOMERS"
  3  FOR EACH ROW
  4  BEGIN
  5   SELECT "SEQ_CUSTOMER".NEXTVAL INTO :NEW.C_ID FROM DUAL;
  6  END;
  7  /

Trigger created.

SQL>
```

B: SEQ_SERVICE_HISTORY AND TR_SERVICE_ID

```
CREATE SEQUENCE "SEQ_SERVICE_HISTORY" MINVALUE 125000 MAXVALUE 999999999999
INCREMENT BY 1 START WITH 125000;

CREATE OR REPLACE TRIGGER "TR_SERVICE_ID"
BEFORE INSERT ON "SERVICE_HISTORY"
FOR EACH ROW
BEGIN
 SELECT "SEQ_SERVICE_HISTORY".NEXTVAL INTO :NEW.SERVICE_ID FROM DUAL;
END;
/
```

SQL Plus

```
SQL> CREATE SEQUENCE "SEQ_SERVICE_HISTORY" MINVALUE 125000 MAXVALUE 999999999999
  2  INCREMENT BY 1 START WITH 125000;

Sequence created.

Elapsed: 00:00:00.00
SQL>
SQL> CREATE OR REPLACE TRIGGER "TR_SERVICE_ID"
  2  BEFORE INSERT ON "SERVICE_HISTORY"
  3  FOR EACH ROW
  4  BEGIN
  5   SELECT "SEQ_SERVICE_HISTORY".NEXTVAL INTO :NEW.SERVICE_ID FROM DUAL;
  6  END;
  7  /

Trigger created.

Elapsed: 00:00:00.05
SQL>
```

C: TR_SERVICE_HISTORY_MESSAGE

```
CREATE OR REPLACE TRIGGER "TR_SERVICE_HISTORY_MESSAGE"
```

Ali Nawaz Maan
Student ID: 44688813

```
BEFORE INSERT OR UPDATE ON SERVICE_HISTORY
FOR EACH ROW
DECLARE
   CUSTOMER_F VARCHAR2(50);
 CUSTOMER_L VARCHAR2(50);
   DOG VARCHAR2(50);
   BREED VARCHAR2(50);
 STORE VARCHAR2(50);
 STORE_MESSAGE VARCHAR2(200);
BEGIN

   SELECT DISTINCT F_NAME
     INTO CUSTOMER_F
     FROM CUSTOMERS, DOGS, SERVICE_HISTORY
     WHERE DOGS.DOG_ID= :NEW.DOG_ID
  AND DOGS.C_ID = CUSTOMERS.C_ID;

  SELECT DISTINCT L_NAME
     INTO CUSTOMER_L
     FROM CUSTOMERS, DOGS, SERVICE_HISTORY
     WHERE DOGS.DOG_ID= :NEW.DOG_ID
  AND DOGS.C_ID = CUSTOMERS.C_ID;

  SELECT DISTINCT DOG_NAME
     INTO DOG
     FROM DOGS, SERVICE_HISTORY
     WHERE DOGS.DOG_ID= :NEW.DOG_ID;

  SELECT DISTINCT DOG_BREED
     INTO BREED
     FROM DOGS, SERVICE_HISTORY
     WHERE DOGS.DOG_ID= :NEW.DOG_ID;

  SELECT DISTINCT STORE_AREA
     INTO STORE
     FROM STORES, SERVICE_HISTORY
     WHERE STORES.STORE_ID= :NEW.STORE_ID;

   IF (:NEW.FINISHED ='T') THEN
  STORE_MESSAGE := ' is ready for pick-up at ' || STORE || '.';
   ELSE
     STORE_MESSAGE := ' is not ready to be picked up yet.';
   END IF;

   :NEW.MESSAGE := 'Hi ' || CUSTOMER_F || ' ' || CUSTOMER_L || ' your dog ' || DOG || ' of breed: ' || BREED ||
' ' || STORE_MESSAGE;

   END;
/
```

```
SQL> CREATE OR REPLACE TRIGGER "TR_SERVICE_HISTORY_MESSAGE" BEFORE INSERT OR UPDATE ON SERVICE_HISTORY FOR EACH ROW DECLARE     CUSTOMER_F VARCHAR2(5
0); CUSTOMER_L VARCHAR2(50);     DOG VARCHAR2(50);     BREED VARCHAR2(50); STORE VARCHAR2(50); STORE_MESSAGE VARCHAR2(200); BEGIN     SELECT DISTINCT F
_NAME        INTO CUSTOMER_F        FROM CUSTOMERS, DOGS, SERVICE_HISTORY        WHERE DOGS.DOG_ID= :NEW.DOG_ID  AND DOGS.C_ID = CUSTOMERS.C_ID;     SEL
ECT DISTINCT L_NAME        INTO CUSTOMER_L        FROM CUSTOMERS, DOGS, SERVICE_HISTORY        WHERE DOGS.DOG_ID= :NEW.DOG_ID  AND DOGS.C_ID = CUSTOMER
S.C_ID;     SELECT DISTINCT DOG_NAME        INTO DOG        FROM DOGS, SERVICE_HISTORY        WHERE DOGS.DOG_ID= :NEW.DOG_ID;     SELECT DISTINCT DOG_BRE
ED        INTO BREED        FROM DOGS, SERVICE_HISTORY        WHERE DOGS.DOG_ID= :NEW.DOG_ID;     SELECT DISTINCT STORE_AREA        INTO STORE        FROM
 STORES, SERVICE_HISTORY        WHERE STORES.STORE_ID= :NEW.STORE_ID;     IF (:NEW.FINISHED ='T') THEN  STORE_MESSAGE := ' is ready for pick-up at ' |
| STORE || '.';     ELSE      STORE_MESSAGE := ' is not ready to be picked up yet.';     END IF;     :NEW.MESSAGE := 'Hi ' || CUSTOMER_F || ' ' || CUST
OMER_L || ' your dog ' || DOG || ' of breed: ' || BREED || ' ' || STORE_MESSAGE;     END;
  2  /

Trigger created.

Elapsed: 00:00:00.06
SQL>
```

D: TRIGGER TESTING

Ali Nawaz Maan
Student ID: 44688813

```
INSERT INTO CUSTOMERS (F_NAME, L_NAME, DOB)
VALUES ('Luke', 'Cheung', '08-OCT-1996');

INSERT INTO SERVICE_HISTORY (DOG_ID, STORE_ID, FINISHED)
VALUES (1234, 30, 'F');
```

```
SQL> INSERT INTO CUSTOMERS (F_NAME, L_NAME, DOB)
  2  VALUES ('Luke', 'Cheung', '08-OCT-1996');

1 row created.

Elapsed: 00:00:00.03
SQL> INSERT INTO SERVICE_HISTORY (DOG_ID, STORE_ID, FINISHED)
  2  VALUES (1234, 30, 'F');

1 row created.

Elapsed: 00:00:00.11
SQL>
```

TASK 3: VIEWS

A: V_DOG_BREED_STATISTICS

```
CREATE VIEW V_DOG_BREED_STATISTICS AS
SELECT DOG_BREED, SUM(PRICE) AS TOTAL, AVG(PRICE) AS MEAN, STDDEV(PRICE) AS
STANDARD_DEV
FROM SERVICE_HISTORY, DOGS, DOG_BREEDS, SERVICE_HISTORY_DETAIL, SERVICES
WHERE
SERVICE_HISTORY.DOG_ID = DOGS.DOG_ID AND
DOGS.DOG_BREED = DOG_BREEDS.BREED AND
SERVICE_HISTORY.SERVICE_ID = SERVICE_HISTORY_DETAIL.SERVICE_ID AND
SERVICE_HISTORY_DETAIL.SERVICE_NAME = SERVICES.SERVICE_NAME
GROUP BY DOG_BREED;
```

```
SQL> CREATE VIEW V_DOG_BREED_STATISTICS AS
  2  SELECT DOG_BREED, SUM(PRICE) AS TOTAL, AVG(PRICE) AS MEAN, STDDEV(PRICE) AS STANDARD_DEV
  3  FROM SERVICE_HISTORY, DOGS, DOG_BREEDS, SERVICE_HISTORY_DETAIL, SERVICES
  4  WHERE
  5  SERVICE_HISTORY.DOG_ID = DOGS.DOG_ID AND
  6  DOGS.DOG_BREED = DOG_BREEDS.BREED AND
  7  SERVICE_HISTORY.SERVICE_ID = SERVICE_HISTORY_DETAIL.SERVICE_ID AND
  8  SERVICE_HISTORY_DETAIL.SERVICE_NAME = SERVICES.SERVICE_NAME
  9  GROUP BY DOG_BREED;

View created.
```

B: MV_DOG_BREED_STATISTICS

```
CREATE MATERIALIZED VIEW MV_DOG_BREED_STATISTICS AS
SELECT DOG_BREED, SUM(PRICE) AS TOTAL, AVG(PRICE) AS MEAN, STDDEV(PRICE) AS
STANDARD_DEV
FROM SERVICE_HISTORY, DOGS, DOG_BREEDS, SERVICE_HISTORY_DETAIL, SERVICES
WHERE
SERVICE_HISTORY.DOG_ID = DOGS.DOG_ID AND
DOGS.DOG_BREED = DOG_BREEDS.BREED AND
SERVICE_HISTORY.SERVICE_ID = SERVICE_HISTORY_DETAIL.SERVICE_ID AND
```

Ali Nawaz Maan
Student ID: 44688813

```
SERVICE_HISTORY_DETAIL.SERVICE_NAME = SERVICES.SERVICE_NAME
GROUP BY DOG_BREED;
```

```
SQL> CREATE MATERIALIZED VIEW MV_DOG_BREED_STATISTICS AS
  2  SELECT DOG_BREED, SUM(PRICE) AS TOTAL, AVG(PRICE) AS MEAN, STDDEV(PRICE) AS STANDARD_DEV
  3  FROM SERVICE_HISTORY, DOGS, DOG_BREEDS, SERVICE_HISTORY_DETAIL, SERVICES
  4  WHERE
  5  SERVICE_HISTORY.DOG_ID = DOGS.DOG_ID AND
  6  DOGS.DOG_BREED = DOG_BREEDS.BREED AND
  7  SERVICE_HISTORY.SERVICE_ID = SERVICE_HISTORY_DETAIL.SERVICE_ID AND
  8  SERVICE_HISTORY_DETAIL.SERVICE_NAME = SERVICES.SERVICE_NAME
  9  GROUP BY DOG_BREED;

Materialized view created.

SQL>
```

C: Performance of Materialised views vs Virtual views

```
SQL> SELECT * FROM V_DOG_BREED_STATISTICS;

DOG_BREED                                          TOTAL       MEAN STANDARD_DEV
-------------------------------------------------- ---------- ---------- ------------
Treeing Walker Coonhound                          14743.91 22.3731563   11.2318752
Austrian Black and Tan Hound                       8242.78 22.1580108   11.0173533
Vanjari Hound                                     19833.82 21.6054684   10.8367866
Shiloh Shepherd Dog                               11763.14 21.9461567   11.0396668
Black Russian Terrier                             16880.86 22.0953665   10.8655713
Old English Sheepdog                              18706.93 21.8283897   11.0955318
Cesky Fousek                                      19380.63 21.8496392   10.9779573
Mackenzie River Husky                             24414.08 22.3572161     10.99651
Danish Swedish Farmdog                            10939.59  22.280224   11.0627056
Wirehaired Vizsla                                 17891.92 22.1434653   11.2989933
Papillon                                          18665.42 21.7545688   10.8643349

DOG_BREED                                          TOTAL       MEAN STANDARD_DEV
-------------------------------------------------- ---------- ---------- ------------
Dandie Dinmont Terrier                            10261.25 21.6026316   11.0616535
```

```
Rat Terrier                                       18151.97  22.605193   11.3098557
German Spaniel                                    13897.25    22.2356   11.0714499
Drever                                            21690.06 21.8209859   10.9373436

515 rows selected.

Elapsed: 00:00:01.40
SQL>
```

SQL Plus

```
SQL> SELECT * FROM MV_DOG_BREED_STATISTICS;

DOG_BREED                                          TOTAL       MEAN STANDARD_DEV
-------------------------------------------------- ---------- ---------- ------------
Treeing Walker Coonhound                          14743.91 22.3731563   11.2318752
Austrian Black and Tan Hound                       8242.78 22.1580108   11.0173533
Vanjari Hound                                     19833.82 21.6054684   10.8367866
Shiloh Shepherd Dog                               11763.14 21.9461567   11.0396668
Black Russian Terrier                             16880.86 22.0953665   10.8655713
Old English Sheepdog                              18706.93 21.8283897   11.0955318
Cesky Fousek                                      19380.63 21.8496392   10.9779573
Mackenzie River Husky                             24414.08 22.3572161     10.99651
Danish Swedish Farmdog                            10939.59  22.280224   11.0627056
Wirehaired Vizsla                                 17891.92 22.1434653   11.2989933
Papillon                                          18665.42 21.7545688   10.8643349

DOG_BREED                                          TOTAL       MEAN STANDARD_DEV
-------------------------------------------------- ---------- ---------- ------------
Dandie Dinmont Terrier                            10261.25 21.6026316   11.0616535
```

```
German Spaniel                                  13897.25    22.2356   11.0714499
Drever                                          21690.06 21.8209859   10.9373436

515 rows selected.

Elapsed: 00:00:01.21
SQL>
```

The query on Materialised view is running faster than that of Virtual view because Materialised views are separate tables where results of the view queries are saved on physical memory. Whereas, Virtual views are nothing but a save query where the results are displayed by querying on an actual table.

As the size of Materialised view table is much smaller in size than the actual table, the query on Materialized view runs faster as compared to Virtual view where the query is on actual table.

D: View Updatability

The first view is a virtual view or a "Named Query". This means that each time a query is made on view, the DBMS gets the result by query modification method and querying on base tables. This ultimately means that this view is always updated as the base table is updated.

On the other hand, the second view is Materialised view which is an actual table with view query results stored on physical memory. This view is not automatically updated whenever there is a change in base tables. In order to update this materialised view, we have to define a refresh policy to trigger when we want the view to get refreshed.

A simple example can be to use "REFRESH FAST ON COMMIT;" command when creating the materialised view to get the view refreshed.

TASK 4: FUNCTION BASED INDEXES

A: DENTAL CHECKUP QUERY

```
SELECT * FROM (

SELECT DOGS.DOG_ID, DOGS.DOG_NAME, STORES.STORE_ID, STORES.STORE_AREA, START_TIME-
END_TIME AS TIME_TAKEN
FROM SERVICE_HISTORY_DETAIL, SERVICE_HISTORY, DOGS, STORES
WHERE
SERVICE_HISTORY_DETAIL.SERVICE_ID = SERVICE_HISTORY.SERVICE_ID AND
SERVICE_HISTORY.STORE_ID = STORES.STORE_ID AND
SERVICE_HISTORY.DOG_ID = DOGS.DOG_ID AND
SERVICE_NAME = 'Dental Checkup'
ORDER BY TIME_TAKEN DESC)

WHERE ROWNUM=1;
```

Ali Nawaz Maan
Student ID: 44688813

```
SQL> SELECT * FROM (
  2  SELECT DOGS.DOG_ID, DOGS.DOG_NAME, STORES.STORE_ID, STORES.STORE_AREA, START_TIME-END_TIME AS TIME_TAKEN
  3  FROM SERVICE_HISTORY_DETAIL, SERVICE_HISTORY, DOGS, STORES where
  4  SERVICE_HISTORY_DETAIL.SERVICE_ID = SERVICE_HISTORY.SERVICE_ID AND
  5  SERVICE_HISTORY.STORE_ID = STORES.STORE_ID AND
  6  SERVICE_HISTORY.DOG_ID = DOGS.DOG_ID AND
  7  SERVICE_NAME = 'Dental Checkup'
  8  ORDER BY TIME_TAKEN DESC)
  9  WHERE ROWNUM=1;

    DOG_ID DOG_NAME                                           STORE_ID STORE_AREA                                         TIME_TAKEN
---------- -------------------------------------------------- ---------- -------------------------------------------------- ----------
----------
      2301 Bella                                                     114 Mount Gravatt East                                 -000000000 00:41:00.000000

Elapsed: 00:00:00.16
SQL>
```

B: FUNCTION BASED INDEX

```
CREATE INDEX SUBTRACT_START_END ON SERVICE_HISTORY_DETAIL(START_TIME-END_TIME);
```

```
SQL> CREATE INDEX SUBTRACT_START_END ON SERVICE_HISTORY_DETAIL(START_TIME-END_TIME);

Index created.

Elapsed: 00:00:01.31
SQL>
```

C: QUERY EXECUTION TIMES AFTER INDEX

```
SQL> SELECT * FROM (
  2  SELECT DOGS.DOG_ID, DOGS.DOG_NAME, STORES.STORE_ID, STORES.STORE_AREA, START_TIME-END_TIME AS TIME_TAKEN
  3  FROM SERVICE_HISTORY_DETAIL, SERVICE_HISTORY, DOGS, STORES where
  4  SERVICE_HISTORY_DETAIL.SERVICE_ID = SERVICE_HISTORY.SERVICE_ID AND
  5  SERVICE_HISTORY.STORE_ID = STORES.STORE_ID AND
  6  SERVICE_HISTORY.DOG_ID = DOGS.DOG_ID AND
  7  SERVICE_NAME = 'Dental Checkup'
  8  ORDER BY TIME_TAKEN DESC)
  9  WHERE ROWNUM=1;

    DOG_ID DOG_NAME                                           STORE_ID STORE_AREA                                         TIME_TAKEN
---------- -------------------------------------------------- ---------- -------------------------------------------------- ----------
----------
      2301 Bella                                                     114 Mount Gravatt East                                 -000000000 00:41:00.000000

Elapsed: 00:00:00.15
SQL>
```

There is not much but a slight difference in query execution time before creating the function based index which computes the difference of two columns. The Function based indices actually is useful where it involves complex and expensive computations when querying the data and performing those operations simultaneously. Here in out query, the difference operation is not much expensive in terms of computing power. This is the reason we are not seeing any significance difference in query execution times.

This is also true because of the approach used to get the result. If we could have used MAX operation on the column differences, creating function based index on that operation might speed things up.

TASK 5: BITMAP INDEXING

A: TOTAL NUMBER OF EACH SERVICE PERFORMED

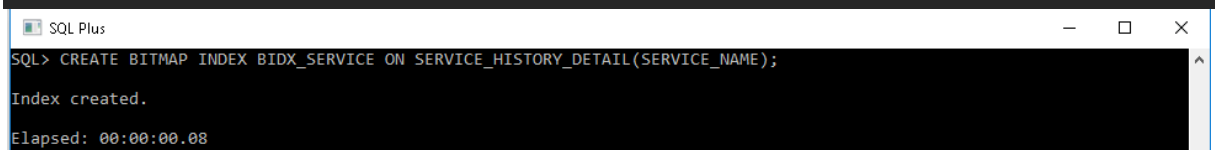
```
SELECT COUNT(*), SERVICE_NAME
FROM SERVICE_HISTORY_DETAIL
GROUP BY SERVICE_NAME;
```

Ali Nawaz Maan
Student ID: 44688813

The query execution time to find the total number of each service performed is 4 milliseconds.

B: BITMAP INDEX BDX_SERVICE

```
CREATE BITMAP INDEX BIDX_SERVICE ON SERVICE_HISTORY_DETAIL(SERVICE_NAME);
```



C: DIFFERENCE IN QUERY EXECUTION TIMES

For finding the total number of each service performed, we chose service name column for the bitmap index. This is because this is the only column present in service_history_detail that can be categorised. But the problem here is that the number of categories of services is too small as compared to the data. For this reason, the created bitmap index does not affect query execution time significantly.

This tells us that in order to have an efficient and effective bitmap index, the column should not have too sparse or too dense categorisation.

D: ADVANTAGES AND DISADVANTAGES OF CONSTRUCTING THIS INDEX

ADVANTAGES

- The biggest advantage for this index is that it is created on service_name column which contains the service name categorization. This means that whenever a query involves operations to fetch the data related to category of serive_name, this index is a great help to speed up query execution time.

- Moreover, this bitmap index is update efficient in terms of service name updatability. This is true because the business Doggo Paradise may not change their service names too frequently. This will ultimately require less update operations on index.
- The index can be compressed which means it offers space efficiency as well. Furthermore, it breaks down complex data into simple and easy to read data to aid processor work efficiently.

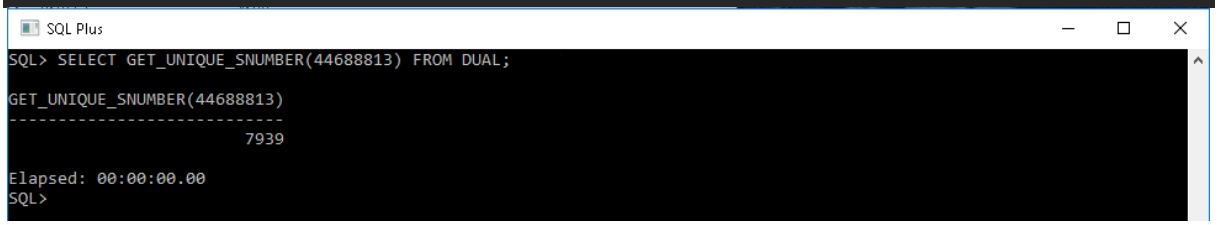
DISADVANTAGES

- This bitmap index affects query execution times less significantly. This is because of the number of actual data records and the number of categories service_name should be in some optimal range in order to feel the difference between execution times. There might be a significant difference if the actual records were in gigabytes of size.
- This index is update inefficient in terms of adding new service_history_detail records. This is true because each time the business performs a service on some dog, the index needs to be updated and cater for new dog service details. This accounts for expensive operations of rearranging the data and hence accounts for update inefficiency.

TASK 6: EXECUTION PLAN AND ANALYSIS

A: GETTING UNIQUE DOG_ID

```
SELECT GET_UNIQUE_SNUMBER(44688813) FROM DUAL;
```



The unique DOG_ID is 7939.

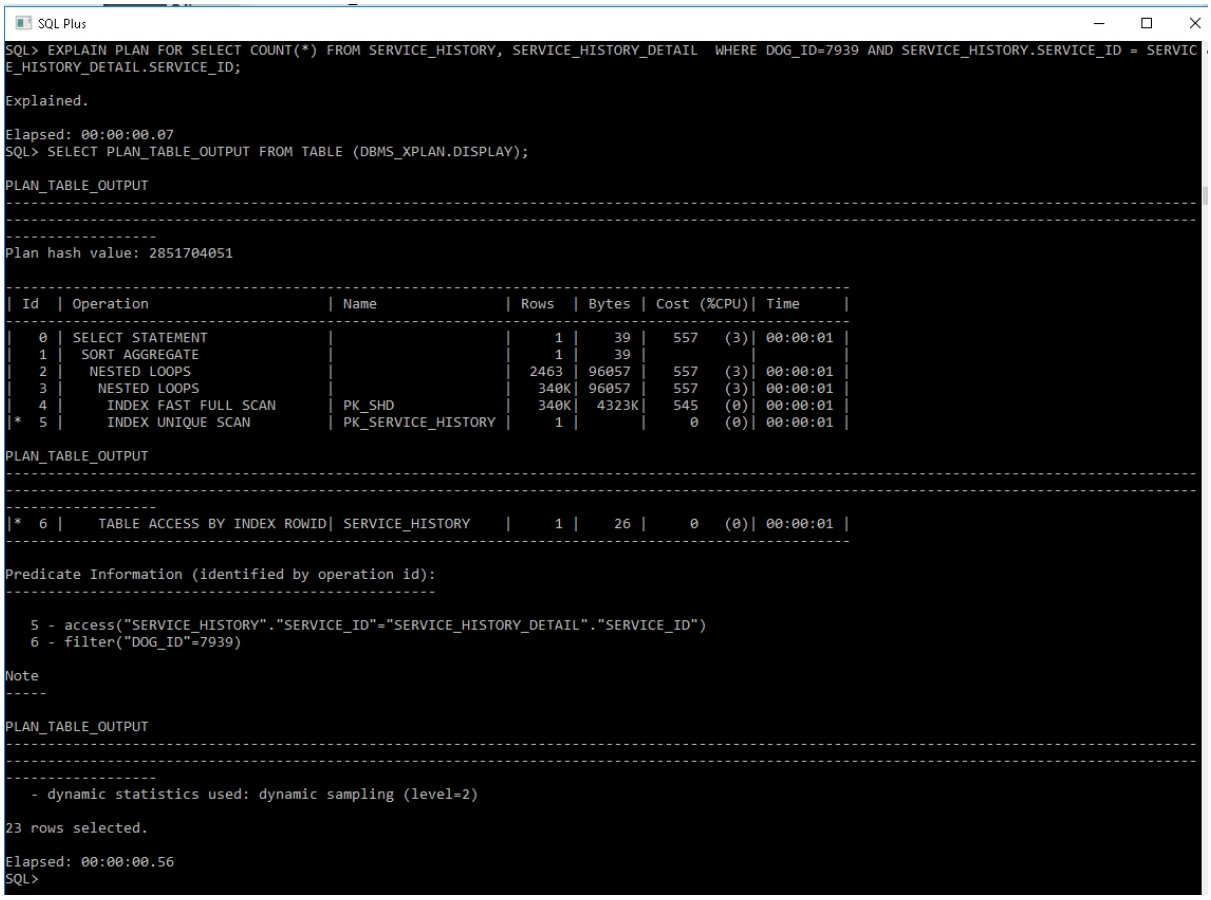
B: FIND B+ TREE INDICES

```
SELECT *
FROM (SELECT A.INDEX_NAME, A.TABLE_NAME, A.COLUMN_NAME, B.INDEX_TYPE
 FROM USER_IND_COLUMNS A, USER_INDEXES B
 WHERE A.TABLE_NAME='STORES' AND A.INDEX_NAME=B.INDEX_NAME AND
 B.INDEX_TYPE='NORMAL'),

 (SELECT A.INDEX_NAME, A.TABLE_NAME, A.COLUMN_NAME, B.INDEX_TYPE
 FROM USER_IND_COLUMNS A, USER_INDEXES B
 WHERE A.TABLE_NAME='SERVICE_HISTORY' AND A.INDEX_NAME=B.INDEX_NAME AND
 B.INDEX_TYPE='NORMAL'),

 (SELECT A.INDEX_NAME, A.TABLE_NAME, A.COLUMN_NAME, B.INDEX_TYPE
 FROM USER_IND_COLUMNS A, USER_INDEXES B
 WHERE A.TABLE_NAME='SERVICE_HISTORY_DETAIL' AND A.INDEX_NAME=B.INDEX_NAME AND
 B.INDEX_TYPE='NORMAL');
```

Ali Nawaz Maan
Student ID: 44688813

C: ALL VISITS TO DOGGO PARADISE

```
EXPLAIN PLAN FOR SELECT COUNT(*) FROM SERVICE_HISTORY, SERVICE_HISTORY_DETAIL
WHERE DOG_ID=7939 AND SERVICE_HISTORY.SERVICE_ID = SERVICE_HISTORY_DETAIL.SERVICE_ID;


SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);
```

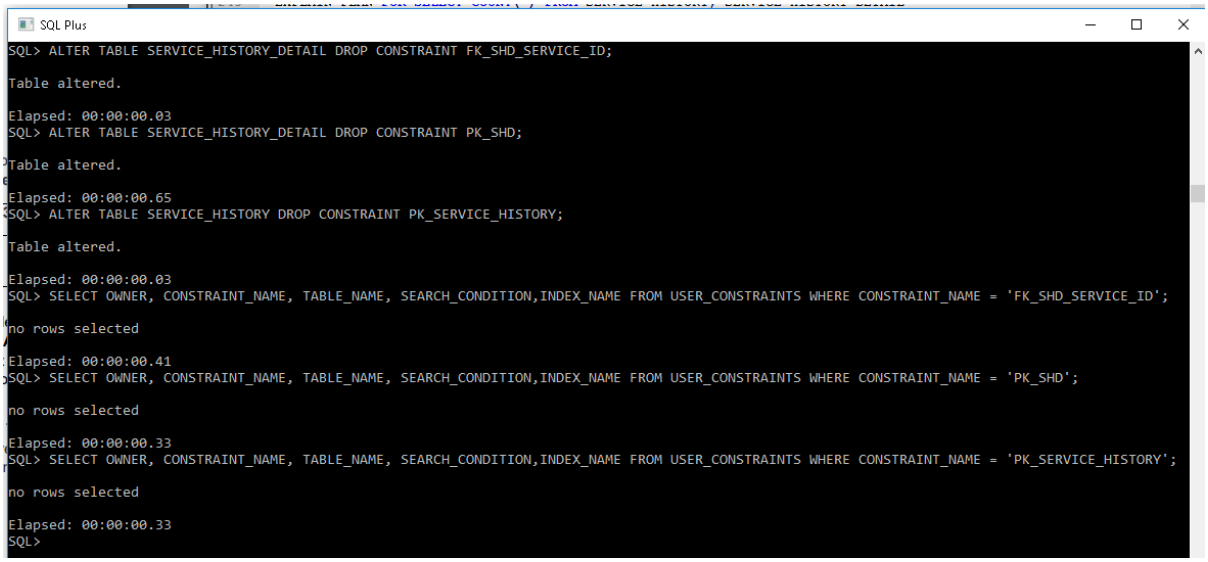


```
SQL Plus
SQL> EXPLAIN PLAN FOR SELECT COUNT(*) FROM SERVICE_HISTORY, SERVICE_HISTORY_DETAIL  WHERE DOG_ID=7939 AND SERVICE_HISTORY.SERVICE_ID = SERVIC
E_HISTORY_DETAIL.SERVICE_ID;

Explained.

Elapsed: 00:00:00.07
SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
------------------
Plan hash value: 2851704051

-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                    |     1 |    39 |   557   (3)| 00:00:01 |
|   1 |  SORT AGGREGATE               |                    |     1 |    39 |            |          |
|   2 |   NESTED LOOPS                |                    |  2463 | 96057 |   557   (3)| 00:00:01 |
|   3 |    NESTED LOOPS               |                    |   340K| 96057 |   557   (3)| 00:00:01 |
|   4 |     INDEX FAST FULL SCAN      | PK_SHD             |   340K|  4323K|   545   (0)| 00:00:01 |
|*  5 |     INDEX UNIQUE SCAN         | PK_SERVICE_HISTORY |     1 |       |     0   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
------------------
|*  6 |    TABLE ACCESS BY INDEX ROWID| SERVICE_HISTORY    |     1 |    26 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   5 - access("SERVICE_HISTORY"."SERVICE_ID"="SERVICE_HISTORY_DETAIL"."SERVICE_ID")
   6 - filter("DOG_ID"=7939)

Note
-----

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
------------------
   - dynamic statistics used: dynamic sampling (level=2)

23 rows selected.

Elapsed: 00:00:00.56
SQL>
```

D: DIFFERENCE IN QUERY EXECUTION

```
ALTER TABLE SERVICE_HISTORY_DETAIL DROP CONSTRAINT FK_SHD_SERVICE_ID;

ALTER TABLE SERVICE_HISTORY_DETAIL DROP CONSTRAINT PK_SHD;

ALTER TABLE SERVICE_HISTORY DROP CONSTRAINT PK_SERVICE_HISTORY;

SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME, SEARCH_CONDITION,
INDEX_NAME FROM USER_CONSTRAINTS WHERE CONSTRAINT_NAME = 'FK_SHD_SERVICE_ID';

SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME, SEARCH_CONDITION,
INDEX_NAME FROM USER_CONSTRAINTS WHERE CONSTRAINT_NAME = 'PK_SHD';

SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME, SEARCH_CONDITION,
INDEX_NAME FROM USER_CONSTRAINTS WHERE CONSTRAINT_NAME = 'PK_SERVICE_HISTORY';
```

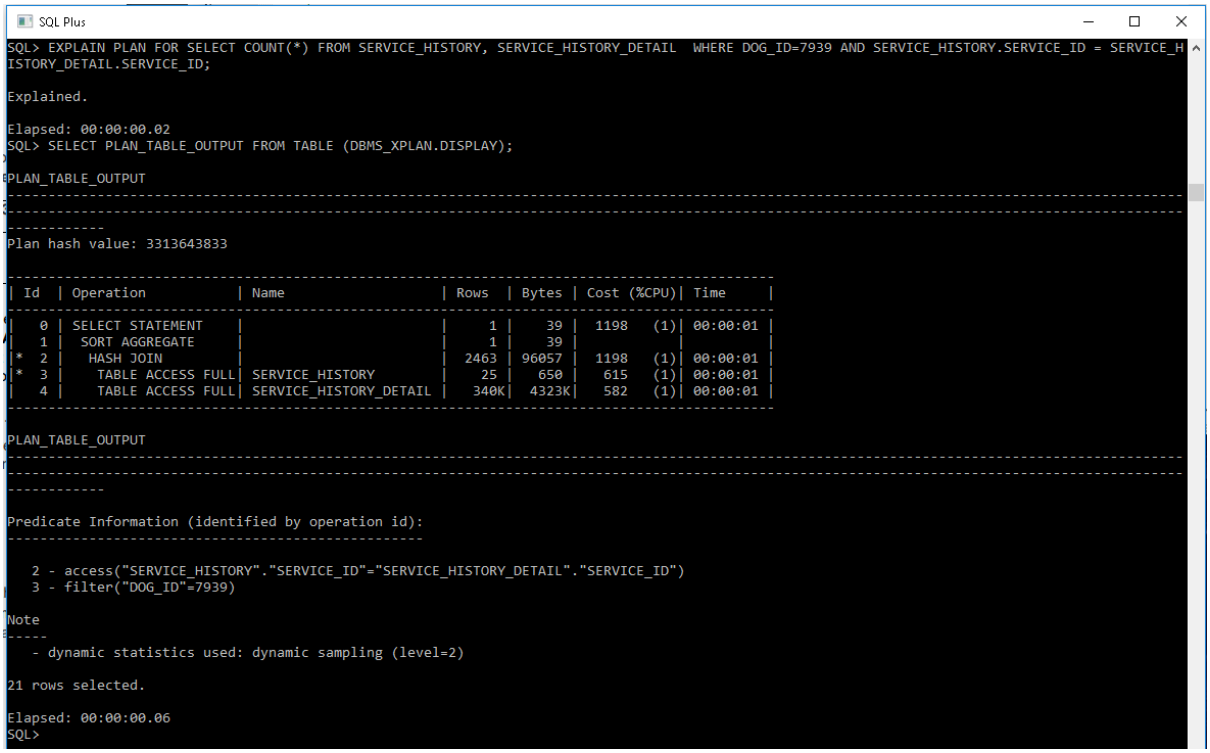
Ali Nawaz Maan
Student ID: 44688813



```
SQL Plus
SQL> ALTER TABLE SERVICE_HISTORY_DETAIL DROP CONSTRAINT FK_SHD_SERVICE_ID;

Table altered.

Elapsed: 00:00:00.03
SQL> ALTER TABLE SERVICE_HISTORY_DETAIL DROP CONSTRAINT PK_SHD;

Table altered.

Elapsed: 00:00:00.65
SQL> ALTER TABLE SERVICE_HISTORY DROP CONSTRAINT PK_SERVICE_HISTORY;

Table altered.

Elapsed: 00:00:00.03
SQL> SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME, SEARCH_CONDITION,INDEX_NAME FROM USER_CONSTRAINTS WHERE CONSTRAINT_NAME = 'FK_SHD_SERVICE_ID';

no rows selected

Elapsed: 00:00:00.41
SQL> SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME, SEARCH_CONDITION,INDEX_NAME FROM USER_CONSTRAINTS WHERE CONSTRAINT_NAME = 'PK_SHD';

no rows selected

Elapsed: 00:00:00.33
SQL> SELECT OWNER, CONSTRAINT_NAME, TABLE_NAME, SEARCH_CONDITION,INDEX_NAME FROM USER_CONSTRAINTS WHERE CONSTRAINT_NAME = 'PK_SERVICE_HISTORY';

no rows selected

Elapsed: 00:00:00.33
SQL>
```

Execution Plan Difference

```
EXPLAIN PLAN FOR SELECT COUNT(*) FROM SERVICE_HISTORY, SERVICE_HISTORY_DETAIL
WHERE DOG_ID=7939 AND SERVICE_HISTORY.SERVICE_ID = SERVICE_HISTORY_DETAIL.SERVICE_ID;


SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);
```



```
SQL Plus
SQL> EXPLAIN PLAN FOR SELECT COUNT(*) FROM SERVICE_HISTORY, SERVICE_HISTORY_DETAIL  WHERE DOG_ID=7939 AND SERVICE_HISTORY.SERVICE_ID = SERVICE_H
ISTORY_DETAIL.SERVICE_ID;

Explained.

Elapsed: 00:00:00.02
SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------
------------
Plan hash value: 3313643833

-------------------------------------------------------------------------------------------
| Id  | Operation           | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                     |     1 |    39 |  1198   (1)| 00:00:01 |
|   1 |  SORT AGGREGATE     |                     |     1 |    39 |            |          |
|*  2 |   HASH JOIN         |                     |  2463 | 96057 |  1198   (1)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| SERVICE_HISTORY     |    25 |   650 |   615   (1)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| SERVICE_HISTORY_DETAIL |  340K|  4323K|   582   (1)| 00:00:01 |
-------------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------
------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("SERVICE_HISTORY"."SERVICE_ID"="SERVICE_HISTORY_DETAIL"."SERVICE_ID")
   3 - filter("DOG_ID"=7939)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

21 rows selected.

Elapsed: 00:00:00.06
SQL>
```

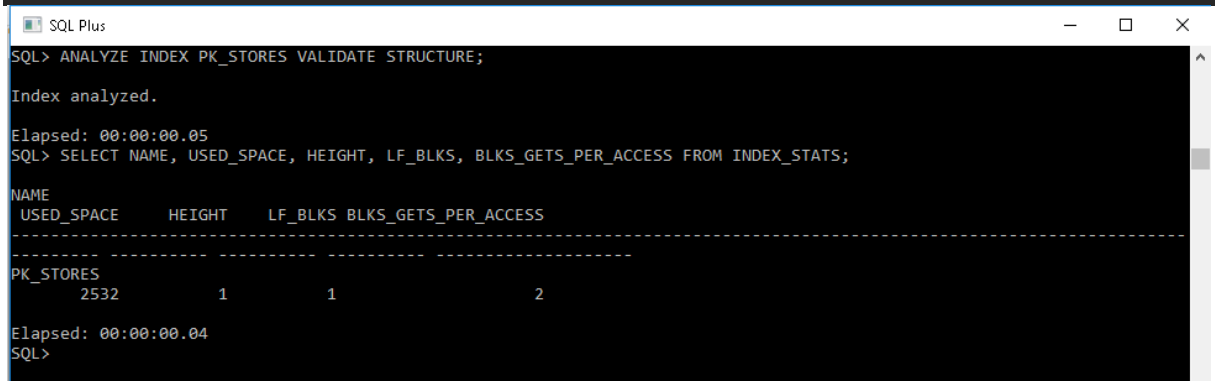
Ali Nawaz Maan
Student ID: 44688813

The query execution plans for the query with B-tree indexes on primary keys and without that have a significant difference. If we look at the operations column for both execution plans, we can clearly see that the plan with B-tree indexes on primary keys is scanning the indexes on Service_History and Serive_History_Detail tables. As searching through index is faster, we can clearly see the magnitude of bytes scanned and cost (% CPU time) that it is pretty low. That's a good thing about indexing - fast access to data without scanning the entire table for matching records.

On the other hand, if we look at the execution plan after dropping the B-tree indexes of primary keys from both tables, it is significantly different. Again, if we look at the operations column, the system is canning both tables entirely for matching records. This is clearly a disadvantage in terms of bytes scanned and cpu cost. The bytes scanned and their relative cpu cost is much greater in this plan than the execution plan with B-tree indexes.

E: INDEX ANALYSIS

```
ANALYZE INDEX PK_STORES VALIDATE STRUCTURE;

SELECT NAME, USED_SPACE, HEIGHT, LF_BLKS, BLKS_GETS_PER_ACCESS FROM INDEX_STATS;
```



Height of Index = 1

Used Space = 2532

Leaf Blocks = 1

Blocks per access = 2