# 1. Primary key definition and any other constraint or index suggestion

The primary keys should be created for unique values that identify one record

alter table ITEM add constraint ITEM\_PK PRIMARY KEY (ITEM, DEPT);

Because the user has only access to one tablespace I did not add the tablespace clause to the objects creation.

alter table LOC add constraint LOC\_PK PRIMARY KEY (LOC);

alter table ITEM\_LOC\_SOH add constraint ITEM\_LOC\_SOH\_PK PRIMARY KEY (ITEM, LOC, DEPT);

Foreign keys to enforce integrity between data

alter table ITEM\_LOC\_SOH add constraint ILS\_IT\_FK FOREIGN KEY (ITEM, DEPT) references ITEM (ITEM, DEPT);

alter table ITEM\_LOC\_SOH add constraint ILS\_LOC\_FK FOREIGN KEY (LOC) references LOC (LOC);

Also indexes for the foreign keys to avoid contention on delete operations on parent tables for instance

create index ILS\_IT\_FK\_IDX on ITEM\_LOC\_SOH (ITEM, DEPT);

create index ILS\_LOC\_FK\_IDX on ITEM\_LOC\_SOH (LOC);

# 2. Your suggestion for table data management and data access considering the application usage, for example, partition...

If the number of stores was small (like 10 for example) and a large number of items and dept per store, the table could be partitioned by store but with 1000 stores having 1000 partitions in the table and creating a new one for each store does not seem the best option for this scenario.

With the primary keys and indexes created, the application behavior should be acceptable. Just by doing queries before and after the indexes creation, the difference is noticeable.

The database should be monitored in several periods during the normal usage and in those where the load was higher a new analysis should be done but as mentioned above with the indexes created the access to the tables should be faster and with that the performance better.

# 3. Your suggestion to avoid row contention at table level parameter because of high level of concurrency

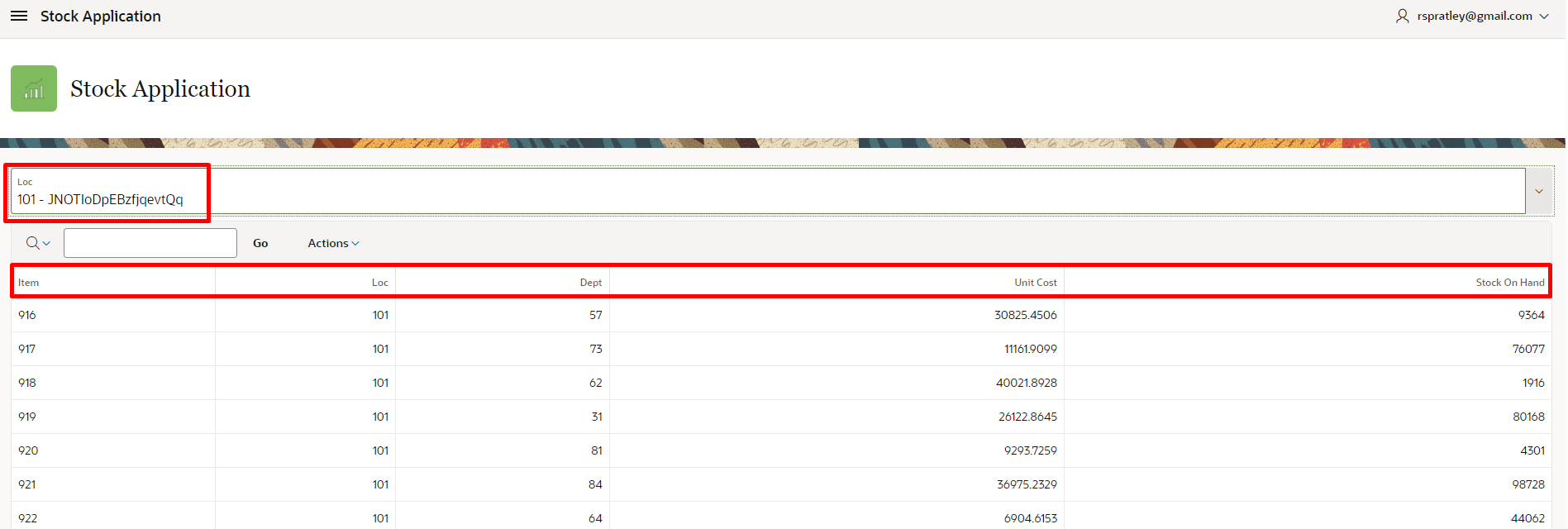
Minimize number of DML operations not committed

For large DML operations try to use BULK commands like FORALL

If the row contention still exists analyze V$SESSION and V$ACTIVE\_SESSION\_HISTORY tables to see the behavior at runtime and if any query is a bottleneck analyze it and see actions to improve it by changing the query or adding any missing constraintor index, among others

# 4. Create a view that can be used at screen level to show only the required fields

Based on the fields visible in the application:



CREATE OR REPLACE FORCE VIEW V\_LOC\_IT (ITEM, LOC, LOC\_DESC, DEPT, UNIT\_COST, STOCK\_ON\_HAND)

as SELECT ILS.ITEM, ILS.LOC, L.LOC\_DESC, ILS.DEPT, ILS.UNIT\_COST, ILS.STOCK\_ON\_HAND

FROM ITEM\_LOC\_SOH ILS

JOIN LOC L

ON L.LOC = ILS.LOC;

# 5. Create a new table that associates user to existing dept(s)

First create the USERS table:

create table USERS(

user\_id varchar2(25) not null,

user\_desc varchar2(25) not null,

constraint USERS\_PK PRIMARY KEY (USER\_ID)

);

Then create the DEPT table:

create table DEPTS(

DEPT number(24) not null,

DEPT\_DESC varchar2(25) not null,

constraint DEPTS\_PK PRIMARY KEY (DEPT)

);

Then create the association table

create table USERS\_DEPTS(

USER\_ID varchar2(25) not null,

DEPT number(24) not null,

constraint USERS\_DEPTS\_PK PRIMARY KEY (USER\_ID, DEPT)

);

Also as above the FKs and IDX should be created

alter table USERS\_DEPTS add constraint UD\_US\_FK FOREIGN KEY (USER\_ID) references USERS (USER\_ID);

alter table USERS\_DEPTS add constraint UD\_DPT\_FK FOREIGN KEY (DEPT) references DEPTS (DEPT);

create index UD\_US\_FK\_IDX on USERS\_DEPTS (USER\_ID);

create index UD\_DPT\_FK\_IDX on USERS\_DEPTS (DEPT);

# 6. Create a package with procedure or function that can be invoked by store or all stores to save the item\_loc\_soh to a new table that will contain the same information plus the stock value per item/loc (unit\_cost\*stock\_on\_hand)

create table item\_loc\_soh\_cost(

item varchar2(25) not null,

loc number(10) not null,

dept number(4) not null,

unit\_cost number(20,4) not null,

stock\_on\_hand number(12,4) not null,

total\_cost number(24,4) not null,

constraint ITEM\_LOC\_SOH\_COST\_PK PRIMARY KEY (ITEM, LOC, DEPT)

);

CREATE OR REPLACE PACKAGE pk\_store\_info AS

PROCEDURE migrate\_info (loc\_id NUMBER);

END pk\_store\_info;

--The reason to have different versions of the body is explained in the migration section

--Version 1

--Using forall by record

CREATE OR REPLACE PACKAGE BODY pk\_store\_info AS

PROCEDURE migrate\_info (loc\_id NUMBER) IS

TYPE ARRAY IS TABLE OF item\_loc\_soh\_cost%ROWTYPE;

l\_data ARRAY;

CURSOR c IS select item, loc, dept, unit\_cost, stock\_on\_hand, unit\_cost \* stock\_on\_hand

from item\_loc\_soh

where loc = loc\_id

and loc\_id is not null

union all

select item, loc, dept, unit\_cost, stock\_on\_hand, unit\_cost \* stock\_on\_hand

from item\_loc\_soh

where loc\_id is null;

BEGIN

OPEN c;

LOOP

FETCH c BULK COLLECT INTO l\_data LIMIT 5000;

FORALL i IN 1..l\_data.COUNT

INSERT INTO item\_loc\_soh\_cost VALUES l\_data(i);

EXIT WHEN c%NOTFOUND;

END LOOP;

CLOSE c;

COMMIT;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: SQLCODE=' || SQLCODE || ' SQLERRM=' || SQLERRM);

ROLLBACK;

END;

END pk\_store\_info;

/

--Version 2

--Using insert as select

CREATE OR REPLACE PACKAGE BODY pk\_store\_info AS

PROCEDURE migrate\_info (loc\_id NUMBER) IS

BEGIN

insert /\*+ APPEND NOLOGGING \*/ into item\_loc\_soh\_cost (item, loc, dept, unit\_cost, stock\_on\_hand, total\_cost)

select item, loc, dept, unit\_cost, stock\_on\_hand, unit\_cost \* stock\_on\_hand

from item\_loc\_soh

where loc = loc\_id

and loc\_id is not null

union all

select item, loc, dept, unit\_cost, stock\_on\_hand, unit\_cost \* stock\_on\_hand

from item\_loc\_soh

where loc\_id is null;

COMMIT;

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: SQLCODE=' || SQLCODE || ' SQLERRM=' || SQLERRM);

ROLLBACK;

END;

END pk\_store\_info;

/

# 7. Create a data filter mechanism that can be used at screen level to filter out the data that user can see accordingly to dept association (created previously)

With just one query:

select \*

from item\_loc\_soh ils

join user\_dept ud

on ud.dept = ils.dept

where ud.user\_id = :user;

--:user – Bind variable to filter the username

By the information on the screen another option could be:

For the loc filter on the top of the page:

Select l.loc, l.loc\_desc

From loc l

Join item\_loc\_soh ils

On ils.loc = l.loc

Join users\_depts ud

On ud.dept = ils.dept

where ud.user\_id = :user;

--:user – Bind variable to filter the username

Then the detail screen could have the same behavior.

# 8. Create a pipeline function to be used in the location list of values (drop down)

CREATE OR REPLACE TYPE location\_list\_t IS TABLE OF VARCHAR2 (25);

CREATE OR REPLACE FUNCTION location\_list

RETURN location\_list\_t PIPELINED AUTHID DEFINER IS

cursor c\_loc is

select loc

from loc;

BEGIN

for r\_loc in c\_loc

loop

PIPE ROW (r\_loc.loc);

end loop;

RETURN;

END;

/

SELECT COLUMN\_VALUE FROM TABLE (location\_list ());

# 9. Looking into the following explain plan what should be your recommendation and implementation to improve the existing data model. Please share your solution in sql and the corresponding explain plan of that solution. Please take in consideration the way that user will use the app.

*```sql*

*Plan Hash Value : 1697218418*

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

*| Id | Operation | Name | Rows | Bytes | Cost | Time |*

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

*| 0 | SELECT STATEMENT | | 100019 | 40760 | 10840 | 00:00:03 |*

*| \* 1 | TABLE ACCESS FULL |* ***ITEM\_LOC\_SOH*** *| 100019 | 40760 | 10840 | 00:00:03 |*

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

*Predicate Information (identified by operation id):*

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

*\* 1 - filter("****LOC****"=652 AND "****DEPT****"=68)*

*Notes*

*-----*

*- Dynamic sampling used for this statement ( level = 2 )*

*```*

First action would be to create a index for the columns being filtered:

create index ILS\_LOC\_IDX1 on ITEM\_LOC\_SOH (LOC, DEPT);

Running the explain plan for the same query:

explain plan for select \* from **ITEM\_LOC\_SOH** where **loc** = 652 and **dept** = 68;

select \* from table(dbms\_xplan.display);

Result:

Plan hash value: 509953501

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

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

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

| 0 | SELECT STATEMENT | | 1224 | 80784 | 163 (0)| 00:00:01 |

| 1 | TABLE ACCESS BY INDEX ROWID BATCHED| ITEM\_LOC\_SOH | 1224 | 80784 | 163 (0)| 00:00:01 |

|\* 2 | INDEX RANGE SCAN | ILS\_LOC\_IDX1 | 196 | | 3 (0)| 00:00:01 |

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

Predicate Information (identified by operation id):

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

2 - access("LOC"=652 AND "DEPT"=68)

Note

-----

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

# 10. Run the previous method that was created on 6. for all the stores from item\_loc\_soh to the history table. The entire migration should not take more than 10s to run (don't use parallel hint to solve it :))

BEGIN

pk\_store\_info.migrate\_info(null);

END;

/

Because I was not able to insert the records in the APEX account due to the fact that I had only a small quota in tablespace and did not had dba privilege to change I had to create a XE database in my computer to test.

Because we are speaking of a full migration and assuming the table is empty I used the following approach:

--Disable the constraint, although I have created it being an historic table as mention the constraint could not be created assuming that is validated in the origin table:

ALTER TABLE item\_loc\_soh\_cost DISABLE CONSTRAINT ITEM\_LOC\_SOH\_COST\_PK ;

BEGIN

pk\_store\_info.migrate\_info(null);

END;

/

ALTER TABLE item\_loc\_soh\_cost ENABLE NOVALIDATE CONSTRAINT ITEM\_LOC\_SOH\_COST\_PK ;

Version #1 – 15 seconds

Version 2 – 6 seconds

# 11. Please have a look into the AWR report (AWR.html) in attachment and let us know what is the problem that the AWR is highlighting and potential solution.

The first thing that is noticeable in the AWR is the high number of logical reads

Most of the reads are from 2 JOBS (DBMS\_SCHEDULER) and for automatic indexing SYS\_AI\_MODULE

Assuming that the storage access is not the bottleneck than the information access is (queries)

Create a new AWR after the indexes creation and monitor again to see if the values are the same or improved.