Database Migrations

Because of all the deleting that has to be done I find it illogical to clean up their HQ database and then have to also fix issues with missing data that now occurs in their HQ database. There are orphan records and empty values in some tables that might affect the running of RMH. Phillyp has gotten some errors just trying to convert the HQ database to Central and I suspect some of these issues are causing the problems. I propose to clean up both store databases and from one store generate the Central database, then sync the tables and at the end do an upload of the historical data. These are all available in RMH. Below I will detail all the steps to achieve this. I also include detailed scripts to run on each database to achieve the clean up requested by the customer. You can run this whole process in your lab and right before going live repeat the process with the customer's live data. Since the customer is live and continues to sell you must wait until the last moment to run this process. The complete process should take 1 hour or 2 it depends on the hardware and internet speeds between the Central location and the remote store. We always leave a copy of RMS running at the stores on at least 1 computer so if needed they can go and look at the data. RMS and RMH journals (digital copy of the receipt) are stored differently so in RMH you will not be able to view the journals from RMS. Hence leaving access to the RMS database and Store Manager.

The steps below assume that RMH store and Central software has been installed and ports are configured so Central clients connect to Central server. Also that a backup of the RMS databases is performed and then restored into a new set of databases so those will be the RMH databases and not lose the RMS databases.

Plan of action

1. Delete department and categories to match the list provided by the customer.
2. Run provided scripts at both stores to delete
    a. Historical data before 2018
    b. Items created before 2018
3. Compact and reindex databases after cleaning up.
4. Convert databases to RMH.
5. Create Central DB and import 1 of the store databases.
6. Run in RMH Central Client the initial store sync for each store.
7. Run the consistency checker to upload store historical data for each store.

1.Delete department and categories to match the list provided by the customer.

Since Phillyp stated he already was able to perform this he can replicate this in the live environment. I am not providing scripts for this.

2.Deleting data from store databases.

Some background on the process. For some tables I am running a truncate rather than deleting each record. It is faster and will not affect the running of RMH. Before each delete or update of a table I disable all triggers on that table. This will make the process run faster and since we are deleting there is no need to have triggers run. I have separated each delete for easier understanding. I would run one at a time just to validate that there was no issues running it before moving to the next. I have also included a short comment explaining each one. Yoo can copy and paste directly into a SSMS query. Comments will be ignored. Some of the deletes might return no rows deleted.

```
truncate table RecordDeletedLog; /* Holds just the id and the table or deleted records. No very useful for lack of info, just used to know if a record was deleted */
truncate table timecard; /* Holds each login a user makes. Clean start for RMH*/

disable trigger all on Journal;
Delete from Journal;      /*RMS Journal is not compatible with RMH no need to keep it */
enable trigger all on Journal;

disable trigger all on DailySales;
delete from DailySales where date < '1/1/2018'; /* Daily Sales totals for department, category, supplier, cashier, salesrep, register*/
delete from DailySales where type=2 and typeid not in (Select id from category); /* Daily Sales totals for categories that do not exist*/
delete from DailySales where type=3 and typeid not in (Select id from department); /* Daily Sales totals for departments that do not exist*/
enable trigger all on DailySales;

disable trigger all on [Order];
delete from [Order] where time < '1/1/2018'; /* Delete Quotes, Work Orders and layaways */
enable trigger all on [Order];

disable trigger all on [Transaction];
delete from [Transaction] where time < '1/1/2018'; /*Delete sales */
enable trigger all on [Transaction];

disable trigger all on [NonTenderTransaction];
delete from [NonTenderTransaction] where time < '1/1/2018'; /* Records for No Sales, Aborted Transactions, X Reports, Z Reports, ZZ Reports, and Blind Close outs*/
enable trigger all on [NonTenderTransaction];

disable trigger all on Batch;
delete from Batch where ClosingTime< '1/1/2018'; /* Sale Batches */
enable trigger all on Batch;

disable trigger all on PurchaseOrder;
delete from PurchaseOrder where DateCreated < '1/1/2018'; /*Purchases */
enable trigger all on PurchaseOrder;

disable trigger all on InventoryTransferLog;
delete from InventoryTransferLog where DateTransferred < '1/1/2018'; /*Item receptions, transfers, adjustments */
enable trigger all on InventoryTransferLog;

disable trigger all on ItemValueLog;
delete from ItemValueLog where lastupdated < '1/1/2018'; /*Holds each time an item's cost or price is changed */
enable trigger all on ItemValueLog;
```

```
disable trigger all on InventoryOffline;
delete from InventoryOffline where date < '1/1/2018'; /* Items moved from main inventory*/
enable trigger all on InventoryOffline;

disable trigger all on OrderEntry;
delete from OrderEntry where orderid not in (select id from [order]); /*Items in Quotes, Work Orders and layaways */
enable trigger all on OrderEntry;

disable trigger all on OrderHistory;
delete from OrderHistory where orderid not in (select id from [order]); /*Holds each time there is an update to Quotes,
Work Orders and layaways */
enable trigger all on OrderHistory;

disable trigger all on PhysicalInventory;
delete from PhysicalInventory where OpenTime < '1/1/2018'; /*Holds item counts */
enable trigger all on PhysicalInventory;

disable trigger all on PhysicalInventoryEntry;
delete from PhysicalInventoryEntry where PhysicalInventoryid not in (select id from PhysicalInventory); /*Items counted
where the count was deleted in the previous step */
enable trigger all on PhysicalInventoryentry;

disable trigger all on PurchaseOrderEntry;
delete from PurchaseOrderEntry where PurchaseOrderid not in (select id from PurchaseOrder); /* Items in a purchase
order or transfer that where the order was deleted in a previous step*/
enable trigger all on PurchaseOrderEntry;

disable trigger all on TransactionEntry;
delete from TransactionEntry where transactionnumber not in (select transactionnumber from [transaction]); /*Items in a
sale that where the sale was deleted in a previous step */
enable trigger all on TransactionEntry;

disable trigger all on TaxEntry;
delete from TaxEntry where transactionnumber not in (select transactionnumber from [transaction]); /*Items taxes in a
sale that where the sale was deleted in a previous step*/
enable trigger all on TaxEntry;

disable trigger all on TaxTotals;
delete from TaxTotals where Batchnumber not in (select Batchnumber from Batch); /* tax batch totals for deleted
batches*/
enable trigger all on TaxTotals;

disable trigger all on Tenderentry;
delete from Tenderentry where Batchnumber not in (select Batchnumber from Batch); /*tender details for deleted
batches */
enable trigger all on Tenderentry;

disable trigger all on TenderTotals;
delete from TenderTotals where Batchnumber not in (select Batchnumber from Batch); /* tender batch totals for deleted
batches*/
enable trigger all on TenderTotals;
```

```
disable trigger all on item; /* This will delete all items older that 2018 and not in the remaining sales, quotes, layaways, purchaseorders, transfers, counts or has movement after 1/1/2018*/
delete from item where datecreated < '1/1/2028' and itemlookupcode not in ('SGC', 'SGC1', 'SGC2') and id not in
(select distinct itemid from TransactionEntry
union all
select distinct itemid from OrderEntry
union all
select distinct itemid from InventoryTransferLog
union all
select distinct itemid from PurchaseOrderEntry
union all
select distinct itemid from PhysicalInventoryEntry);
enable trigger all on item;
```

After deleting items older than 2018 then we will delete from all tables that might have items that have been deleted.

```
disable trigger all on serial;
delete from serial where itemID not in (select id from Item); /* Holds serial numbers for serializes items and gift cards*/
delete from serial where TransactionEntryID <>0 and TransactionEntryID not in (select id from TransactionEntry); /* delete serial number entries for deleted transaction lines*/
disable trigger all on serial;

disable trigger all on OrderEntry;
delete from OrderEntry where itemID not in (select id from Item); /* delete lines that point to deleted items*/
enable trigger all on OrderEntry;

disable trigger all on PhysicalInventoryEntry;
delete from PhysicalInventoryEntry where itemID not in (select id from Item); /* delete lines that point to deleted items*/
enable trigger all on PhysicalInventoryEntry;

disable trigger all on InventoryOffline;
delete from InventoryOffline where itemID not in (select id from Item); /* delete lines that point to deleted items*/
enable trigger all on InventoryOffline;

disable trigger all on PurchaseOrderEntry;
delete from PurchaseOrderEntry where itemID not in (select id from Item); /* delete lines that point to deleted items*/
enable trigger all on PurchaseOrderEntry;

disable trigger all on ItemClassComponent;
delete from ItemClassComponent where itemID not in (select id from Item); /* delete matrix component that point to deleted items*/
enable trigger all on ItemClassComponent;

disable trigger all on itemclass;
delete from ItemClass where ID not in (select ItemClassid from ItemClassComponent); /* Delete Matrices that have no components*/
enable trigger all on ItemClass;

disable trigger all on MatrixAttributeDisplayOrder;
delete from MatrixAttributeDisplayOrder where ItemClassid not in (select id from ItemClass); /* holds Matrix Attribute display order with no attributes*/
```

```
enable trigger all on MatrixAttributeDisplayOrder;
```

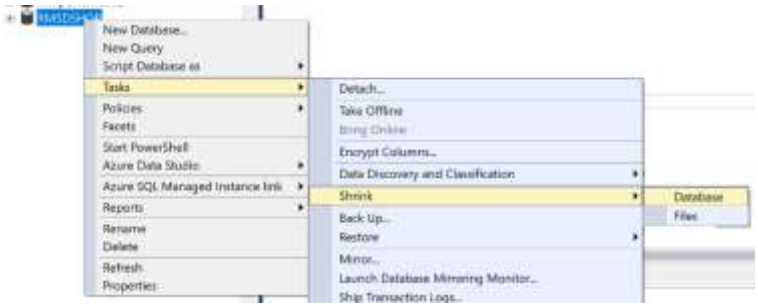
After all deletions we need to address items that are pointing to deleted departments, categories, suppliers, and quantitydiscounts. Lastly, we will set the last sold and last received dates based on the existing data of sales and purchases. RMS did not always these fields.

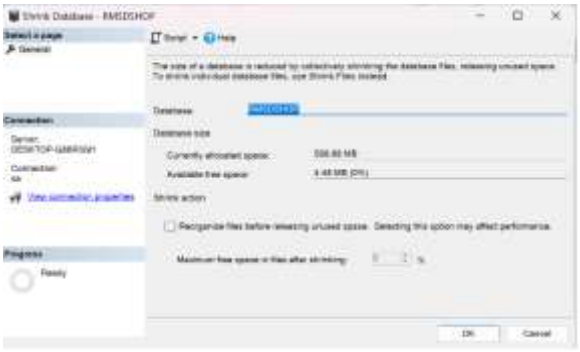
```
disable trigger all on item;
update item set DepartmentID=0 where departmentid <>0 and departmentid not in (select id from department); /*Zero out item's department that point to non-existent departments*/
update item set categoryID=0 where categoryid <>0 and categoryid not in (select id from category); /*Zero out item's category  that point to non-existent categories */
update item set supplierID=0 where supplierid <>0 and supplierid not in (select id from supplier); /*Zero out item's supplier that point to non-existent suppliers */
update item set QuantityDiscountid=0 where QuantityDiscountid <>0 and QuantityDiscountid not in (select id from QuantityDiscount); /*Zero out item's quantitydiscount that point to non-existent discounts */
update item set LastReceived=lr.lastreceiveddate from item,   /* set the item's last received date. RMS did not always update it*/
(select itemid, max(lastreceiveddate) as lastreceiveddate from PurchaseOrderEntry where QuantityReceivedToDate<>0 group by itemid) lr where item.id=lr.itemid;
update item set LastSold=ls.lastsold from item,   /* set the item's last sold date. RMS did not always update it*/
(select itemid, max(transactiontime) as lastsold from TransactionEntry group by itemid) ls where item.id=ls.itemid;
enable trigger all on item;
```

Compact and reindex databases after cleaning up.

After deleting all this data, you should compact the database and then reindex it. Here are the steps in case you are not sure.

In Microsoft SQL Management Studio right click on the database name and then select tasks, shrink and database





Press OK on the next window. This process might take some time. Once completed the window will close. At this point we are ready to reindex the database. Replace YOUR DATABASE NAME HERE with the database name. Again, this script might take some time to run. When finished it will respond with Commands completed successfully.

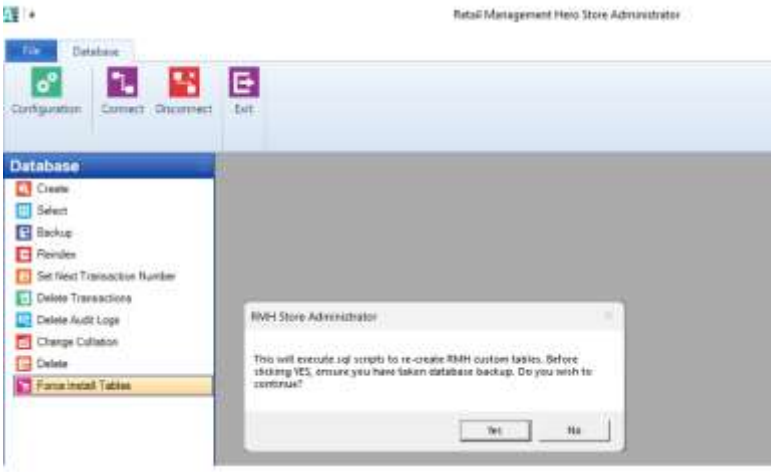
```
DECLARE @Database NVARCHAR(255)
DECLARE @Table NVARCHAR(255)
DECLARE @cmd NVARCHAR(1000)

SET @Database='YOUR DATABASE NAME HERE'
SET @cmd = 'DECLARE TableCursor CURSOR READ_ONLY FOR SELECT ''['' + table_catalog + ''].['' + table_schema + ''].['' +
  table_name + '']'' as tableName FROM [' + @Database + '].INFORMATION_SCHEMA.TABLES'
  EXEC (@cmd)
  OPEN TableCursor
  FETCH NEXT FROM TableCursor INTO @Table
  WHILE @@FETCH_STATUS = 0
  BEGIN
    BEGIN TRY
      SET @cmd = 'ALTER INDEX ALL ON ' + @Table + ' REBUILD'
      EXEC (@cmd)
    END TRY
    BEGIN CATCH
      PRINT '---'
      PRINT @cmd
      PRINT ERROR_MESSAGE()
      PRINT '---'
    END CATCH

    FETCH NEXT FROM TableCursor INTO @Table
  END
  CLOSE TableCursor
  DEALLOCATE TableCursor
```

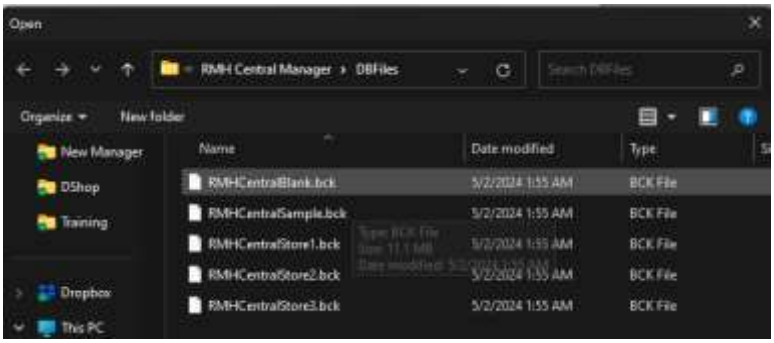
Convert databases to RMH.

In the RMH Store Administrator connect to the database and then run the Force install tables. Depending on the database size and computer speed it might take some time.



Create Central DB and import 1 of the store databases.

In the Central Administrator connect and then Create a new Central Database, select the RMHCentralBlank backup file to create the Central Database.

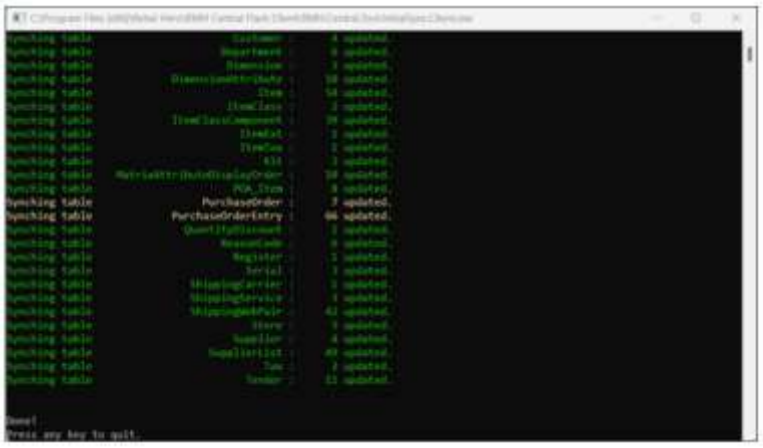


Once created run the force install tables.
Select import store database, connect to the newly converted RMH database from the previous section.
Open Central server and configure the settings to point to the new Central DB
Save Changes
Press **Prepare database**

When completed make sure no errors are reported.
Start the Central Server services.

Run in RMH Central Client the initial store sync for each store.

Open the Central Client
Configure the store database settings and then save changes
Press **Prepare database**
Once completed press **Synchronize Store Database with Server**



Since the Central DB was created from this store there will be no sync error or warnings

Run the consistency checker to upload store historical data for each store.

Next upload all historical data to Central.
In the settings tabs change the **From date** to 1/1/2018. Make sure the **To date** is set to today.
No checkboxes are selected so all data uploads
Press **Run**

A new window will appear and show all records uploading. It will close when all is completed. Repeat this whole section for the other database