hub_duplicate_errors

Column	**Description**	**Example**
Tablename	Hub table name	hub_account
Source_tablename	Staged table, although the staged table is not involved in the duplicate check it is informative to know which load initiated the check, can occur multiple times if the source loads to the same hub more than once, i.e. staged file contains a relationship to the same hub.	staged.card_masterfile
Loaddate	Batch load date	2021-06-10 00:00:00
Rundate	When test was executed	2021-06-22 18:24:13
HUB_SKEY_DUPE_err	Duplicate surrogate key error count	0
HUB_SKEY_DUPE_tgt_columns	Array of surrogate hash key, this must be unique	["dv_hashkey_hub_account"]
HUB_BKEY_DUPE_err	Duplicate business key(s) error count	0
HUB_BKEY_DUPE_tgt_columns	Array of columns included for hash key generation; this must be unique	["dv_tenantid", "dv_bkeycolcode", "account_id"]

Code:

```
alter session set query_tag = 'Recon: utilities.reconcile_hub_duplicate_errors';
insert into utilities.reconcile_hub_duplicate_errors(tablename, source_tablename, loaddate, rundate, HUB_SKEY_DUPE_err,
HUB_SKEY_DUPE_tgt_columns, HUB_BKEY_DUPE_err, HUB_BKEY_DUPE_tgt_columns)
with HUB_SKEY_DUPE as (
select count(e) HUB_SKEY_DUPE_err
    , array_construct('dv_hashkey_hub_account') as HUB_SKEY_DUPE_tgt_columns
from (select count(*) e
      from rawvault.hub_account
      group by dv_hashkey_hub_account
having count(*) > 1) sq)
```

```
, HUB_BKEY_DUPE as (
select count(e) HUB_BKEY_DUPE_err
    , array_construct('dv_tenantid', 'dv_bkeycolcode', 'account_id') as HUB_BKEY_DUPE_tgt_columns
from (select count(*) e
      from rawvault.hub_account
      group by dv_tenantid, dv_bkeycolcode, account_id
      having count(*)>1) sq)
 select 'hub_account' as tablename
 , 'staged.card_masterfile' as source_tablename
 , to_timestamp($my_loaddate) as loaddate
 , current_timestamp() as rundate
 , HUB_SKEY_DUPE_err
 , HUB_SKEY_DUPE_tgt_columns
 , HUB_BKEY_DUPE_err
 , HUB_BKEY_DUPE_tgt_columns
from HUB_SKEY_DUPE
, HUB_BKEY_DUPE
;
```

No Snowflake Streams are used here since the effort to check uniqueness must be on the whole hub table, and besides, it's a small and thin table

hub_reconciliation_errors

Column	Description	Example
Tablename	Hub table name	hub_account
Source_tablename	Staged table, required because a hub can have many staged sources	staged.card_masterfile
Loaddate	Batch load date	2021-06-20 00:00:00
Rundate	When test was executed	2021-06-22 18:46:29
HUB_SKEY_SGTG_ncnt	**New** hash key count after load – base this on hub recon stream,	123
HUB_SKEY_SGTG_scnt	**Staged** hash key count – staged hash key, uses Snowflake Cache	94858

Column	Description	Example
HUB_SKEY_SGTG_dcnt	**Staged Distinct** hash key count – staged hash key, (optional) does not use Snowflake Cache	88927
HUB_SKEY_SGTG_total	**Total** hub table count after load, uses Snowflake Cache	147359
HUB_SKEY_SGTG_err	**Missing** surrogate key error count	0
HUB_SKEY_SGTG_src_columns	Array of hash key, same staged file can have multiple hash columns (i.e., same-as, hierarchy link tables)	["dv_hashkey_hub_account"]
HUB_SKEY_SGTG_tgt_columns	Array of hash key, this for the most part, will be the same as the source column	["dv_hashkey_hub_account"]
HUB_BKEY_SGTG_err	Missing business key(s) error count	0
HUB_BKEY_SGTG_src_columns	Array of columns included for hash key in the source; uniqueness should match HUB_SKEY_SGTG_src_columns. If this were a same as link for example the other array of columns would be [“dv_tenantid”, “dv_bkeycolcode_hub_other_account”, “other_account_id”]	["dv_tenantid", "dv_bkeycolcode_hub_account", "account_id"]
HUB_BKEY_SGTG_tgt_columns	Array of columns included for hash key in the target, for a given hub this will always be the same	["dv_tenantid", "dv_bkeycolcode", "account_id"]

Code:

```
alter session set query_tag = 'Recon: utilities.reconcile_hub_reconciliation_errors';
insert into utilities.reconcile_hub_reconciliation_errors(tablename, source_tablename, loaddate, rundate, HUB_SKEY_SGTG_ncnt,
HUB_SKEY_SGTG_scnt, HUB_SKEY_SGTG_dcnt, HUB_SKEY_SGTG_total, HUB_SKEY_SGTG_err,
HUB_SKEY_SGTG_src_columns, HUB_SKEY_SGTG_tgt_columns, HUB_BKEY_SGTG_err, HUB_BKEY_SGTG_src_columns,
HUB_BKEY_SGTG_tgt_columns)
with HUB_SKEY_SGTG as (
select count(*) HUB_SKEY_SGTG_err
     , array_construct('dv_hashkey_hub_account') as HUB_SKEY_SGTG_src_columns
     , array_construct('dv_hashkey_hub_account') as HUB_SKEY_SGTG_tgt_columns
from staged.card_masterfile sg
where not exists
(select 1
 from rawvault.hub_account h
 where sg.dv_hashkey_hub_account = h.dv_hashkey_hub_account ))
, HUB_BKEY_SGTG as (
select count(*) HUB_BKEY_SGTG_err
     , array_construct('dv_tenantid', ' dv_bkeycolcode_hub_account ', 'account_id') as HUB_BKEY_SGTG_src_columns
     , array_construct('dv_tenantid', 'dv_bkeycolcode', 'account_id') as HUB_BKEY_SGTG_tgt_columns
from staged.card_masterfile sg
where NOT EXISTS (select *
from rawvault.hub_account h
where sg.account_id = h.account_id
and sg.dv_tenantid = h.dv_tenantid
and sg.dv_bkeycolcode_hub_account = h.dv_bkeycolcode))
, Fetch_Hub_Stats as (
  select count(dv_hashkey_hub_account) HUB_SKEY_SGTG_ncnt
  from utilities.reconcile_hub_account)
, Fetch_Hub_Stats1 as (
  select count(dv_hashkey_hub_account) HUB_SKEY_SGTG_scnt
  , count(distinct dv_hashkey_hub_account) HUB_SKEY_SGTG_dcnt
  from staged.card_masterfile)
, Fetch_Hub_Stats2 as (
  select count(dv_hashkey_hub_account) HUB_SKEY_SGTG_total
  from rawvault.hub_account)
 select 'hub_account' as tablename
 , 'staged.card_masterfile' as source_tablename
 , to_timestamp($my_loaddate) as loaddate
 , current_timestamp() as rundate
 , HUB_SKEY_SGTG_ncnt
 , HUB_SKEY_SGTG_scnt
 , HUB_SKEY_SGTG_dcnt
 , HUB_SKEY_SGTG_total
 , HUB_SKEY_SGTG_err
 , HUB_SKEY_SGTG_src_columns
 , HUB_SKEY_SGTG_tgt_columns
 , HUB_BKEY_SGTG_err
 , HUB_BKEY_SGTG_src_columns
 , HUB_BKEY_SGTG_tgt_columns
from HUB_SKEY_SGTG
, HUB_BKEY_SGTG
, Fetch_Hub_Stats
, Fetch_Hub_Stats1
, Fetch_Hub_Stats2
;
```

This uses Streams to detect new keys loaded, Stream: utilities.reconcile_hub_account

sat_duplicate_errors

Column	**Description**	**Example**
Tablename	Satellite table name	sat_card_masterfile
Source_tablename	Staged table, although the staged table is not involved in the duplicate check it is informative to know which load initiated the check	staged.card_masterfile
Loaddate	Batch load date	2021-07-01 00:00:00
Rundate	When test was executed	2021-06-22 19:21:55
SAT_SKEY_DUPE_err	Duplicate surrogate key + load date error count	0

Column	Description	Example
SAT_SKEY_DUPE_tgt_columns	Array of hash key, - including the tenant id deals with the odd configuration of having multiple tenants load to the same satellite - including the hashdiff deals with satellites with dependent-child keys and multi-active satellite configurations	["dv_hashkey_hub_account", “dv_loaddate”, “dv_tenantid”, “dv_hashdiff”]

Code:

```
alter session set query_tag = 'Recon: utilities.reconcile_sat_duplicate_errors';
insert into utilities.reconcile_sat_duplicate_errors(tablename, source_tablename, loaddate, rundate, SAT_SKEY_DUPE_err,
SAT_SKEY_DUPE_tgt_columns)
 select 'sat_card_masterfile' as tablename
    , 'staged.card_masterfile' as source_tablename
    , to_timestamp($my_loaddate) as loaddate
    , current_timestamp() as rundate
    , count(e) SAT_SKEY_DUPE_err
    , array_construct('dv_hashkey_hub_account', 'dv_loaddate', 'dv_tenantid', 'dv_hashdiff') as SAT_SKEY_DUPE_tgt_columns
from (select count(*) e
          , dv_loaddate
      from rawvault.vc_sat_card_masterfile
      group by dv_hashkey_hub_account, dv_loaddate, dv_tenantid, dv_hashdiff
      having count(*) > 1) sq
      ;
```

This will use the current record to check against, rawvault.vc_sat_card_masterfile
No Snowflake Streams are used here since the effort to check uniqueness must be on current satellite record

sat_reconciliation_errors

Column	**Description**	**Example**
Tablename	Satellite table name	sat_card_masterfile
Source_tablename	Staged table	staged.card_masterfile
Loaddate	Batch load date	2021-06-27 00:00:00
Rundate	When test was executed	2021-06-22 19:12:30
SAT_SKEY_SGTG_ncnt	**New** hash key + load date count after load – base this on satellite recon stream	555
SAT_SKEY_SGTG_scnt	**Staged** hash key + load date count – staged records, uses Snowflake Cache	94725
SAT_SKEY_SGTG_dcnt	**Staged Distinct** hash key + load date count – staged records, it may differ due to record condensing, (optional) does not use	94700
SAT_SKEY_SGTG_total	**Total** satellite table count after load, uses Snowflake Cache	147360
SAT_SKEY_SGTG_err	**Missing** surrogate key error count based on satellite current view	0
SAT_SKEY_SGTG_src_columns	Array of columns included for satellite table uniqueness, hash key, source column can differ from target column if the source includes multiple business keys going to the same hub, such as same-as or hierarchy link, we do not bother with dependent child keys because this key is included in the hashdiff calculation and thus checked in another stat	["dv_hashkey_hub_account",]
SAT_SKEY_SGTG_tgt_columns	Array of columns included for satellite table uniqueness, hash key	["dv_hashkey_hub_account"]
SAT_HDIF_SGTG_err	Missing record hash and surrogate hash key error count based on satellite current view	0
SAT_HDIF_SGTG_src_columns	Array of record hash column, important to note for satellite splitting, should always include the hash key because the same hashdiff could occur for a different hash key	[“'dv_hashkey_hub_account'”, "dv_hashdiff_sat_card_masterfile", “dv_tenantid”]

Column	Description	Example
SAT_HDIF_SGTG_src_hdiff_column	Array of columns included in record hash column, includes dependent-child key	["card_type", "card_balance", "card_status", "credit_limit"]
SAT_HDIF_SGTG_tgt_columns	Array of columns included for hash key	["dv_hashdiff]

Code:

```
alter session set query_tag = 'Recon: utilities.reconcile_sat_reconciliation_errors';
insert into utilities.reconcile_sat_reconciliation_errors(tablename, source_tablename, loaddate, rundate, SAT_SKEY_SGTG_ncnt,
SAT_SKEY_SGTG_scnt, SAT_SKEY_SGTG_dcnt, SAT_SKEY_SGTG_total, SAT_SKEY_SGTG_err, SAT_SKEY_SGTG_src_columns,
SAT_SKEY_SGTG_tgt_columns, SAT_HDIF_SGTG_err, SAT_HDIF_SGTG_src_columns, SAT_HDIF_SGTG_src_hdiff_columns,
SAT_HDIF_SGTG_tgt_columns)
with SAT_SKEY_SGTG as (
select count(*) SAT_SKEY_SGTG_err
     , array_construct('dv_hashkey_hub_account') as SAT_SKEY_SGTG_src_columns
     , array_construct('dv_hashkey_hub_account') as SAT_SKEY_SGTG_tgt_columns
from staged.card_masterfile sg
where not exists
(select 1
 from rawvault.vc_sat_card_masterfile s
 where sg.dv_hashkey_hub_account = s.dv_hashkey_hub_account ))
, SAT_HDIF_SGTG as (
select count(*) SAT_HDIF_SGTG_err
     , array_construct('dv_hashkey_hub_account', 'dv_hashdiff_sat_card_masterfile', 'dv_tenantid') as SAT_HDIF_SGTG_src_columns
     , array_construct('dv_hashkey_hub_account', 'dv_hashdiff', 'dv_tenantid') as SAT_HDIF_SGTG_tgt_columns
from staged.card_masterfile sg
where NOT EXISTS (select *
from rawvault.vc_sat_card_masterfile s -- uses current record for the satellite
where sg.dv_hashkey_hub_account = s.dv_hashkey_hub_account
and sg.dv_hashdiff_sat_card_masterfile = s.dv_hashdiff
and sg.dv_tenantid = s.dv_tenantid))
, Fetch_Sat_Stats as (
  select count(dv_hashkey_hub_account, dv_loaddate) SAT_SKEY_SGTG_ncnt
  from utilities.reconcile_sat_card_masterfile)
, Fetch_Sat_Stats1 as (
  select count(dv_hashkey_hub_account, dv_loaddate) SAT_SKEY_SGTG_scnt
    , count(distinct dv_hashkey_hub_account, dv_loaddate) SAT_SKEY_SGTG_dcnt
  from staged.card_masterfile)
, Fetch_Sat_Stats2 as (
  select count(*) SAT_SKEY_SGTG_total
  from rawvault.sat_card_masterfile)
 select 'sat_card_masterfile' as tablename
 , 'staged.card_masterfile' as source_tablename
 , to_timestamp($my_loaddate) as loaddate
 , current_timestamp() as rundate
 , SAT_SKEY_SGTG_ncnt
 , SAT_SKEY_SGTG_scnt
 , SAT_SKEY_SGTG_dcnt
 , SAT_SKEY_SGTG_total
 , SAT_SKEY_SGTG_err
 , SAT_SKEY_SGTG_src_columns
 , SAT_SKEY_SGTG_tgt_columns
 , SAT_HDIF_SGTG_err
 , SAT_HDIF_SGTG_src_columns
 , array_construct('card_type', 'card_balance', 'card_status', 'credit_limit') as SAT_HDIF_SGTG_src_hdiff_column
 , SAT_HDIF_SGTG_tgt_columns
from SAT_SKEY_SGTG
, SAT_HDIF_SGTG
, Fetch_Sat_Stats
, Fetch_Sat_Stats1
, Fetch_Sat_Stats2
;
```

This will use the current record to check against, rawvault.vc_sat_card_masterfile
This uses Streams to detect new keys loaded, Stream: utilities.reconcile_sat_card_masterfile

sat_referential_errors

Column	**Desc**	**Example**
Tablename	Satellite table name	sat_card_masterfile
Parent_tablename	Parent table, hub or link, satellite table only has one parent	hub_account
Loaddate	Batch load date	2021-06-30 00:00:00
Rundate	When test was executed	2021-06-22 19:19:53
SAT_SKEY_ORPH_err	Missing surrogate key error count, base this on satellite orphan stream	0

Code:

```
alter session set query_tag = 'Recon: utilities.reconcile_sat_referential_errors';
insert into utilities.reconcile_sat_referential_errors(tablename, parent_tablename, loaddate, rundate, SAT_SKEY_ORPH_err)
select 'sat_card_masterfile' as tablename
  , 'hub_account' as parent_tablename
  , to_timestamp($my_loaddate) as loaddate
  , current_timestamp() as rundate
  , count(*) SAT_SKEY_ORPH_err
from utilities.orphancheck_sat_card_masterfile s
where not exists
(select 1
 from rawvault.hub_account p
 where s.dv_hashkey_hub_account = p.dv_hashkey_hub_account )
and s.dv_recsource <> 'GHOST'
;
```

This uses Streams to detect new keys loaded, Stream: utilities.orphancheck_sat_card_masterfile

lnk_duplicate_errors

Column	**Description**	**Example**
Tablename	Link table name	lnk_account_other_account
Source_tablename	Staged table, although the staged table is not involved in the duplicate check it is informative to know which load initiated the check, can occur multiple times if the source loads to the same hub more than once.	staged.card_relatedcard
Loaddate	Batch load date	2021-06-30 00:00:00
Rundate	When test was executed	2021-06-22 19:19:53
LNK_SKEY_DUPE_err	Duplicate link surrogate key error count	0
LNK_SKEY_DUPE_tgt_columns	Array of link hash key, this can include dependent-child keys or degenerative dimensions	["dv_hashkey_lnk_account_other_account"]
LNK_HKEY_DUPE_err	Duplicate hub surrogate key error count	0
LNK_HKEY_DUPE_tgt_columns	Array of hub hash keys, this can include dependent-child keys or degenerative dimensions	["dv_hashkey_hub_account", "dv_hashkey_hub_other_account"]

An addition of new records could cause duplicates; thus, the whole link must be checked, it should be a small table anyway!

Code:

```
alter session set query_tag = 'Recon: utilities.reconcile_lnk_duplicate_errors';
 -- link tests
insert into utilities.reconcile_lnk_duplicate_errors(tablename, source_tablename, loaddate, rundate, LNK_SKEY_DUPE_err,
LNK_SKEY_DUPE_tgt_columns, LNK_HKEY_DUPE_err, LNK_HKEY_DUPE_tgt_columns)
```

```
with LNK_SKEY_DUPE as (
select count(e) LNK_SKEY_DUPE_err
     , array_construct('dv_hashkey_lnk_account_customer') as LNK_SKEY_DUPE_tgt_columns
from (select count(*) e
      from rawvault.lnk_account_customer
      group by dv_hashkey_lnk_account_customer
having count(*) > 1) sq)
, LNK_HKEY_DUPE as (
select count(e) LNK_HKEY_DUPE_err
     , array_construct('dv_hashkey_hub_account', 'dv_hashkey_hub_customer') as LNK_HKEY_DUPE_tgt_columns
from (select count(*) e
      from rawvault.lnk_account_customer
      group by dv_hashkey_hub_account, dv_hashkey_hub_customer
      having count(*)>1) sq)
 select 'lnk_account_customer' as tablename
 , 'staged.card_masterfile' as source_tablename
 , to_timestamp($my_loaddate) as loaddate
 , current_timestamp() as rundate
 , LNK_SKEY_DUPE_err
 , LNK_SKEY_DUPE_tgt_columns
 , LNK_HKEY_DUPE_err
 , LNK_HKEY_DUPE_tgt_columns
from LNK_SKEY_DUPE
, LNK_HKEY_DUPE
;
```

No Snowflake Streams are used here since the effort to check uniqueness must be on the whole link table, and besides, it’s a small and thin table

lnk_reconciliation_errors

Column	**Description**	**Example**
Tablename	Link table name	lnk_account_other_account
Source_tablename	Staged table	staged.card_relatedcard

Column	**Description**	**Example**
Loaddate	Batch load date	2021-06-30 00:00:00
Rundate	When test was executed	2021-06-22 19:19:53
LNK_SKEY_SGTG_ncnt	**New** hash key count after load – base this on link recon stream	433
LNK_SKEY_SGTG_scnt	**Staged** hash key count – staged link hash key, uses Snowflake Cache	43242
LNK_SKEY_SGTG_dcnt	**Staged Distinct** hash key count – staged link hash key, (optional) does not use Snowflake Cache	43239
LNK_SKEY_SGTG_total	**Total** link table count after load, uses Snowflake Cache	54232
LNK_SKEY_SGTG_err	**Missing** surrogate key error count	0
LNK_SKEY_SGTG_src_columns	Array of link hash key, same staged file can have multiple hash columns	["dv_hashkey_lnk_account_other_account"]
LNK_SKEY_SGTG_tgt_columns	Array of link hash key	["dv_hashkey_lnk_account_other_account"]
LNK_HKEY_SGTG_err	**Missing** hub hash keys error count	0
LNK_BKEY_SGTG_src_columns	Array of columns included for link hash key	["dv_tenantid", "dv_bkeycolcode_hub_account", "account_id", "dv_tenantid", "dv_bkeycolcode_hub_other_account", "other_account_id"]
LNK_HKEY_SGTG_tgt_columns	Array of hub hash key columns and tenanted, ensures that if the link structure is shared that the staged content is indeed in the link table structure	[“dv_tenantid”, "dv_hashkey_hub_account", "dv_hashkey_hub_other_account"

Column	Description	Example
		]

Code:

```
alter session set query_tag = 'Recon: utilities.reconcile_lnk_reconciliation_errors';
insert into utilities.reconcile_lnk_reconciliation_errors(tablename, source_tablename, loaddate, rundate, LNK_SKEY_SGTG_scnt,
LNK_SKEY_SGTG_dcnt, LNK_SKEY_SGTG_ncnt, LNK_SKEY_SGTG_total, LNK_SKEY_SGTG_err,
LNK_SKEY_SGTG_src_columns, LNK_SKEY_SGTG_tgt_columns, LNK_HKEY_SGTG_err, LNK_BKEY_SGTG_src_columns,
LNK_HKEY_SGTG_tgt_columns)
with LNK_SKEY_SGTG as (
select count(*) LNK_SKEY_SGTG_err
     , array_construct('dv_hashkey_lnk_account_customer') as LNK_SKEY_SGTG_src_columns
     , array_construct('dv_hashkey_lnk_account_customer') as LNK_SKEY_SGTG_tgt_columns
from staged.card_masterfile sg
where not exists
(select 1
 from rawvault.lnk_account_customer h
 where sg.dv_hashkey_lnk_account_customer = h.dv_hashkey_lnk_account_customer ))
, LNK_BKEY_SGTG as (
select count(*) LNK_HKEY_SGTG_err
     , array_construct('dv_tenantid', 'dv_bkeycolcode_hub_account', 'account_id', 'dv_tenantid', 'dv_bkeycolcode_hub_customer', 'customer_id')
as LNK_BKEY_SGTG_src_columns
     , array_construct('dv_tenantid', 'dv_hashkey_hub_account', 'dv_hashkey_hub_customer') as LNK_HKEY_SGTG_tgt_columns
from staged.card_masterfile sg
where NOT EXISTS (select *
from rawvault.lnk_account_customer h
where sg.dv_hashkey_hub_account = h.dv_hashkey_hub_account
and sg.dv_tenantid = h.dv_tenantid
and sg.dv_hashkey_hub_customer = h.dv_hashkey_hub_customer))
, Fetch_Lnk_Stats as (
  select count(dv_hashkey_lnk_account_customer) LNK_SKEY_SGTG_ncnt
  from utilities.reconcile_lnk_account_customer)
, Fetch_LNK_Stats1 as (
  select count(dv_hashkey_lnk_account_customer) LNK_SKEY_SGTG_scnt
  , count(distinct dv_hashkey_lnk_account_customer) LNK_SKEY_SGTG_dcnt
  from staged.card_masterfile)
, Fetch_LNK_Stats2 as (
  select count(dv_hashkey_lnk_account_customer) LNK_SKEY_SGTG_total
  from rawvault.lnk_account_customer)
 select 'lnk_account_customer' as tablename
 , 'staged.card_masterfile' as source_tablename
 , to_timestamp($my_loaddate) as loaddate
 , current_timestamp() as rundate
 , LNK_SKEY_SGTG_scnt
 , LNK_SKEY_SGTG_dcnt
 , LNK_SKEY_SGTG_ncnt
 , LNK_SKEY_SGTG_total
 , LNK_SKEY_SGTG_err
 , LNK_SKEY_SGTG_src_columns
 , LNK_SKEY_SGTG_tgt_columns
 , LNK_HKEY_SGTG_err
 , LNK_BKEY_SGTG_src_columns
 , LNK_HKEY_SGTG_tgt_columns
from LNK_SKEY_SGTG
, LNK_BKEY_SGTG
, Fetch_LNK_Stats
, Fetch_LNK_Stats1
, Fetch_LNK_Stats2
;
```

This uses Streams to detect new keys loaded, Stream: utilities.reconcile_lnk_account_customer lnk_referential_errors

Column	**Description**	**Example**
Tablename	Link table name	lnk_account_other_account
Parent_tablename	Hub tables, an entry for each participant	hub_account
Link_columnname	Array of hub table hash key, each participant of the link will have a record in this table. Will have the same loaddate but different rundate. This ensures accurate reporting of missing hub hash keys, especially true in a same-as link scenario (same hub used more than once requires both column’s content being loaded to the common hub)	["dv_hashkey_hub_account",]
Loaddate	Batch load date	2021-06-30 00:00:00
Rundate	Supports reloading	2021-06-22 19:19:53
LNK_SKEY_ORPH_err	Err count	0

Code:

```
alter session set query_tag = 'Recon: utilities.reconcile_lnk_referential_errors';
 -- link orph checks
insert into utilities.reconcile_lnk_referential_errors(tablename, parent_tablename, link_columnname, loaddate, rundate,
LNK_SKEY_ORPH_err)
select 'lnk_account_customer' as tablename
  , 'hub_account' as parent_tablename
  , array_construct('dv_hashkey_hub_account') as link_columnname
  , to_timestamp($my_loaddate) as loaddate
  , current_timestamp() as rundate
  , count(*) LNK_SKEY_ORPH_err
from utilities.orphancheck_lnk_account_customer_dv_hashkey_hub_account s
where not exists
(select 1
 from rawvault.hub_account p
 where s.dv_hashkey_hub_account = p.dv_hashkey_hub_account )
```

```
;
```

This uses Streams to detect new keys loaded, Stream utilities.orphancheck_lnk_account_customer_dv_hashkey_hub_account

Streams

Stream	Based on	Rationale
utilities.reconcile_hub_account	rawvault.hub_account	A single stream for the hub to check for new business objects loaded, not one per source. If a source unexpectantly loads records we would detect it, ex. BV loads new records to an RV hub
utilities.reconcile_sat_card_masterfile	rawvault.sat_card_masterfile	Count new records
utilities.orphancheck_sat_card_masterfile	rawvault.sat_card_masterfile	Check that the new loaded records have a parent in hub or link
utilities.reconcile_lnk_account_customer	rawvault.lnk_account_customer	Count new records
utilities.orphancheck_lnk_account_customer_dv_hashkey_hub_account	rawvault.lnk_account_customer	One for each hub-hash key to check against the hub table

Aggregate Views example for reporting

```


-- all hub summary, disregard source
create or replace view reconcile_aggregate_hub as
select h1.tablename
```

```
, h1.loaddate
, sum(h1.hub_skey_dupe_err) as hub_hash_key_duplicates
, sum(h1.hub_bkey_dupe_err) as hub_business_key_duplicates
, sum(h2.hub_skey_sgtg_ncnt) as hub_new_business_objects
, sum(h2.hub_skey_sgtg_scnt) as hub_staged_business_objects
, sum(h2.hub_skey_sgtg_dcnt) as hub_distinct_staged_business_objects
, max(h2.hub_skey_sgtg_total) as hub_business_object_count_after_load
, sum(h2.hub_skey_sgtg_err) as hub_missing_staged_business_objects
, sum(h2.hub_bkey_sgtg_err) as hub_missing_business_keys
from reconcile_hub_duplicate_errors h1
inner join reconcile_hub_reconciliation_errors h2
on h1.tablename = h2.tablename
and h1.loaddate = h2.loaddate
group by h1.tablename, h1.loaddate
;

-- all hub summary, by source
create or replace view reconcile_aggregate_hub_by_source as
select h1.tablename
, h1.loaddate
, h1.source_tablename
, sum(h1.hub_skey_dupe_err) as hub_hash_key_duplicates
, sum(h1.hub_bkey_dupe_err) as hub_business_key_duplicates
, sum(h2.hub_skey_sgtg_ncnt) as hub_new_business_objects
, sum(h2.hub_skey_sgtg_scnt) as hub_staged_business_objects
, sum(h2.hub_skey_sgtg_dcnt) as hub_distinct_staged_business_objects
, max(h2.hub_skey_sgtg_total) as hub_business_object_count_after_load
, sum(h2.hub_skey_sgtg_err) as hub_missing_staged_business_objects
, sum(h2.hub_bkey_sgtg_err) as hub_missing_business_keys
from reconcile_hub_duplicate_errors h1
inner join reconcile_hub_reconciliation_errors h2
on h1.tablename = h2.tablename
and h1.source_tablename = h2.source_tablename
and h1.loaddate = h2.loaddate
group by h1.tablename, h1.source_tablename, h1.loaddate
;

-- all links, disregard source
create or replace view reconcile_aggregate_link as
select l1.tablename
, l1.loaddate
, sum(l1.lnk_skey_dupe_err) as link_hash_key_duplicates
, sum(l1.lnk_hkey_dupe_err) as link_hub_key_duplicates
, sum(l2.lnk_skey_sgtg_ncnt) as link_new_relationships --- FIX
, sum(l2.lnk_skey_sgtg_scnt) as link_staged_relationships
, sum(l2.lnk_skey_sgtg_dcnt) as link_distinct_staged_relationships
, max(l2.lnk_skey_sgtg_total) as link_relationship_count_after_load
, sum(l2.lnk_skey_sgtg_err) as link_missing_staged_relationships
, sum(l2.lnk_hkey_sgtg_err) as link_missing_relationship_hubs
, sum(l3.lnk_skey_orph_err) as link_referential_errors
from reconcile_lnk_duplicate_errors l1
inner join reconcile_lnk_reconciliation_errors l2
on l1.tablename = l2.tablename
and l1.loaddate = l2.loaddate
inner join reconcile_lnk_referential_errors l3
on l1.tablename = l3.tablename
```

```
and l1.loaddate = l3.loaddate
group by l1.tablename, l1.loaddate
;

-- all links, by source
create or replace view reconcile_aggregate_link_by_source as
select l1.tablename
, l1.loaddate
, l1.source_tablename
, sum(l1.lnk_skey_dupe_err) as link_hash_key_duplicates
, sum(l1.lnk_hkey_dupe_err) as link_hub_key_duplicates
, sum(l2.lnk_skey_sgtg_ncnt) as link_new_relationships --- FIX
, sum(l2.lnk_skey_sgtg_scnt) as link_staged_relationships
, sum(l2.lnk_skey_sgtg_dcnt) as link_distinct_staged_relationships
, max(l2.lnk_skey_sgtg_total) as link_relationship_count_after_load
, sum(l2.lnk_skey_sgtg_err) as link_missing_staged_relationships
, sum(l2.lnk_hkey_sgtg_err) as link_missing_relationship_hubs
, sum(l3.lnk_skey_orph_err) as link_referential_errors
from reconcile_lnk_duplicate_errors l1
inner join reconcile_lnk_reconciliation_errors l2
on l1.tablename = l2.tablename
and l1.loaddate = l2.loaddate
inner join reconcile_lnk_referential_errors l3
on l1.tablename = l3.tablename
and l1.loaddate = l3.loaddate
group by l1.tablename, l1.source_tablename, l1.loaddate
;
```

```
-- all sats, disregard source
create or replace view reconcile_aggregate_satellite as
select s1.tablename
, s1.loaddate
, sum(s1.sat_skey_dupe_err) as satellite_load_duplicates
, sum(s2.sat_skey_sgtg_ncnt) as satellite_new_changes
, sum(s2.sat_skey_sgtg_scnt) as satellite_staged_changes
, sum(s2.sat_skey_sgtg_dcnt) as satellite_distinct_staged_changes
, max(s2.sat_skey_sgtg_total) as satellite_record_count_after_load
, sum(s2.sat_skey_sgtg_err) as satellite_missing_staged_objects
, sum(s2.sat_hdif_sgtg_err) as satellite_missing_staged_descriptive_changes
, sum(s3.sat_skey_orph_err) as satellite_referential_errors
from reconcile_sat_duplicate_errors s1
inner join reconcile_sat_reconciliation_errors s2
on s1.tablename = s2.tablename
and s1.loaddate = s2.loaddate
inner join reconcile_sat_referential_errors s3
on s1.tablename = s3.tablename
and s1.loaddate = s3.loaddate
group by s1.tablename, s1.loaddate
;

-- all sats, by source
create or replace view reconcile_aggregate_satellite_by_source as
select s1.tablename
, s1.loaddate
, s1.source_tablename
, sum(s1.sat_skey_dupe_err) as satellite_load_duplicates
, sum(s2.sat_skey_sgtg_ncnt) as satellite_new_changes
, sum(s2.sat_skey_sgtg_scnt) as satellite_staged_changes
, sum(s2.sat_skey_sgtg_dcnt) as satellite_distinct_staged_changes
, max(s2.sat_skey_sgtg_total) as satellite_record_count_after_load
, sum(s2.sat_skey_sgtg_err) as satellite_missing_staged_objects
, sum(s2.sat_hdif_sgtg_err) as satellite_missing_staged_descriptive_changes
, sum(s3.sat_skey_orph_err) as satellite_referential_errors
from reconcile_sat_duplicate_errors s1
inner join reconcile_sat_reconciliation_errors s2
on s1.tablename = s2.tablename
and s1.loaddate = s2.loaddate
inner join reconcile_sat_referential_errors s3
on s1.tablename = s3.tablename
and s1.loaddate = s3.loaddate
group by s1.tablename, s1.source_tablename, s1.loaddate
;

-- all objects, by loaddate
create or replace view reconcile_aggregate_datavault_by_loaddate as
select h.loaddate
, sum(h.hub_hash_key_duplicates) as hub_hash_key_duplicates
, sum(h.hub_business_key_duplicates) as hub_business_key_duplicates
, sum(h.hub_new_business_objects) as hub_new_business_objects
, sum(h.hub_staged_business_objects) as hub_staged_business_objects
, sum(h.hub_distinct_staged_business_objects) as hub_distinct_staged_business_objects
, sum(h.hub_business_object_count_after_load) as hub_business_object_count_after_load
, sum(h.hub_missing_staged_business_objects) as hub_missing_staged_business_objects
, sum(h.hub_missing_business_keys) as hub_missing_business_keys
, sum(l.link_hash_key_duplicates) as link_hash_key_duplicates
, sum(l.link_hub_key_duplicates) as link_hub_key_duplicates
, sum(l.link_new_relationships) as link_new_relationships --- FIX
, sum(l.link_staged_relationships) as link_staged_relationships
, sum(l.link_distinct_staged_relationships) as link_distinct_staged_relationships
, sum(l.link_relationship_count_after_load) as link_relationship_count_after_load
, sum(l.link_missing_staged_relationships) as link_missing_staged_relationships
, sum(l.link_missing_relationship_hubs) as link_missing_relationship_hubs
, sum(l.link_referential_errors) as link_referential_errors
, sum(s.satellite_load_duplicates) as satellite_load_duplicates
, sum(s.satellite_new_changes) as satellite_new_changes
, sum(s.satellite_staged_changes) as satellite_staged_changes
, sum(s.satellite_distinct_staged_changes) as satellite_distinct_staged_changes
, sum(s.satellite_record_count_after_load) as satellite_record_count_after_load
, sum(s.satellite_missing_staged_objects) as satellite_missing_staged_objects
, sum(s.satellite_missing_staged_descriptive_changes) as satellite_missing_staged_descriptive_changes
, sum(s.satellite_referential_errors) as satellite_referential_errors
from reconcile_aggregate_hub h
inner join reconcile_aggregate_link l
on h.loaddate=l.loaddate
inner join reconcile_aggregate_satellite s
on h.loaddate=s.loaddate
group by h.loaddate
;

--- which source contributes the most?
create or replace view reconcile_aggregate_datavault_by_loaddate_sourcetable as
select h.loaddate
```

```
, h.source_tablename
, sum(h.hub_hash_key_duplicates) as hub_hash_key_duplicates
, sum(h.hub_business_key_duplicates) as hub_business_key_duplicates
, sum(h.hub_new_business_objects) as hub_new_business_objects
, sum(h.hub_staged_business_objects) as hub_staged_business_objects
, sum(h.hub_distinct_staged_business_objects) as hub_distinct_staged_business_objects
, sum(h.hub_business_object_count_after_load) as hub_business_object_count_after_load
, sum(h.hub_missing_staged_business_objects) as hub_missing_staged_business_objects
, sum(h.hub_missing_business_keys) as hub_missing_business_keys
, sum(l.link_hash_key_duplicates) as link_hash_key_duplicates
, sum(l.link_hub_key_duplicates) as link_hub_key_duplicates
, sum(l.link_new_relationships) as link_new_relationships --- FIX
, sum(l.link_staged_relationships) as link_staged_relationships
, sum(l.link_distinct_staged_relationships) as link_distinct_staged_relationships
, sum(l.link_relationship_count_after_load) as link_relationship_count_after_load
, sum(l.link_missing_staged_relationships) as link_missing_staged_relationships
, sum(l.link_missing_relationship_hubs) as link_missing_relationship_hubs
, sum(l.link_referential_errors) as link_referential_errors
, sum(s.satellite_load_duplicates) as satellite_load_duplicates
, sum(s.satellite_new_changes) as satellite_new_changes
, sum(s.satellite_staged_changes) as satellite_staged_changes
, sum(s.satellite_distinct_staged_changes) as satellite_distinct_staged_changes
, sum(s.satellite_record_count_after_load) as satellite_record_count_after_load
, sum(s.satellite_missing_staged_objects) as satellite_missing_staged_objects
, sum(s.satellite_missing_staged_descriptive_changes) as satellite_missing_staged_descriptive_changes
, sum(s.satellite_referential_errors) as satellite_referential_errors
from reconcile_aggregate_hub_by_source h
left join reconcile_aggregate_link_by_source l
```

```
on h.loaddate=l.loaddate
and h.source_tablename = l.source_tablename
left join reconcile_aggregate_satellite_by_source s
on h.loaddate=s.loaddate
and h.source_tablename = s.source_tablename
group by h.loaddate, h.source_tablename
order by h.loaddate
;
```

The transaction isolation level is a state within databases that specifies the amount of data that is visible to a statement in a transaction, specifically when the same data source is accessed by multiple transactions simultaneously.

Transaction isolation level is part of the isolation state of a database management system. Isolation is one of the ACID (atomicity, consistency, isolation, durability) properties.

1. Read Uncommitted: One transaction can see the uncommitted changes made by the other transaction. **Azure Synapse (Default)**
2. Read Committed: Implements write locks until the transaction is completed but releases read locks when a SELECT operation is performed. **Snowflake, Azure Synapse**
3. Repeatable Reads: Implements read and write locks until the transaction is completed. Doesn't manage range locks. Reads and DMLs lock
4. Serializable: Implements read and writes locks until the transaction is finished. Also implements range locks. Concurrent executing transactions appears to be serially executing. A query in the current transaction cannot read data modified by another transaction that has not yet committed. No other transaction can modify data being read by the current transaction until it completes, and no other transaction can insert new rows that would match the search condition in the current transaction until it completes. **Redshift**
5. Snapshot, A statement can use data only if it will be in a consistent state throughout the transaction. If another transaction modifies data after the start of the current transaction, the data is not visible to the current transaction. The current transaction works with a snapshot of the data as it existed at the beginning

of that transaction. Snapshot transactions do not request locks when reading data, nor do they block other transactions from writing data. In addition, other transactions writing data do not block the current transaction for reading data **BigQuery**

Isolation Level	Dirty Reads	Non-repeatable Reads	Phantoms
Read Uncommitted	Y	Y	Y
Read Committed	N	Y	Y
Repeatable Read	N	N	Y
Serializable	N	N	N
Snapshot	N	N	N

Dirty Reads: A *dirty read* occurs when a transaction reads data that has not yet been committed

Non-repeatable reads: A *nonrepeatable read* occurs when a transaction reads the same row twice but gets different data each time

Phantom Reads: A *phantom* is a row that matches the search criteria but is not initially seen.

ACID

A - An **atomic transaction** is an *indivisible* and *irreducible* series of database operations such that either *all* occurs, or *nothing* occurs

C - **Consistency (or Correctness)** in database systems refers to the requirement that any given database transaction must change affected data only in allowed ways. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof