# 6.75 Schema Changes

Last Modified on 28/10/2021 2:16 pm ACDT

Schema changes between CareRight V6.74.1 and 6.75

### New Tables/Views

In order to provide more stability when creating Datasets, we have now provided new views or replacements as views for:

- v_receipts
- v_refunds
- v_transactions
- proda_settings
- proda_access_tokens

This will allow us moving forward to change internal schema, without affecting DataSets or Reporting.

### Substantial changes

Combined Payments, Batch Payments should now be accessed via the Receipts view (v_receipts)

Direct use of the Statements table for this information is **deprecated.**

Direct use of underlying tables, where a corresponding view exists, is now **deprecated.**

Unallocated Amount > statement_account_type previously "Patient Account" is now reported as "Combined Payment"

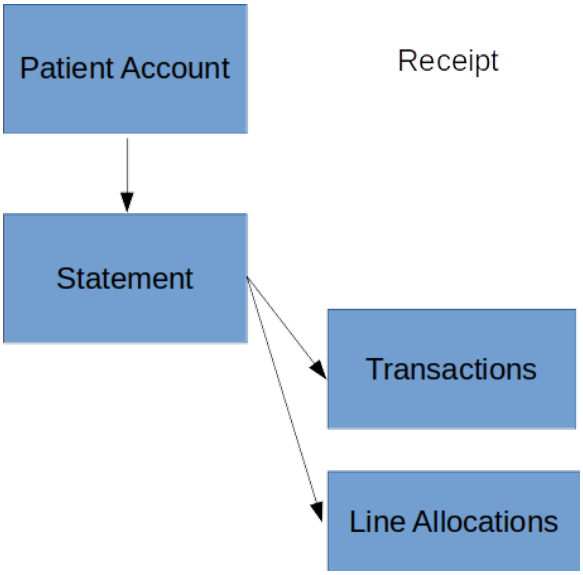
Unallocated Amount > type when previously "Combined" is now reported as "Credit"

Dataset table "Credits and debits" previously incorrectly reported "type" as "Credit" for refunds. They are now reported as "Debit".

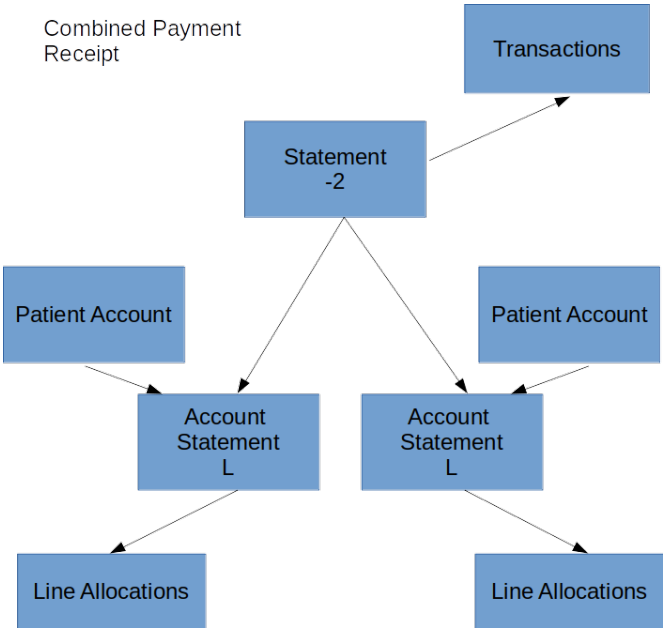
Unallocated amount with type "Combined" is now reported as type "Credit".

### Previous State - Receipts & Refunds

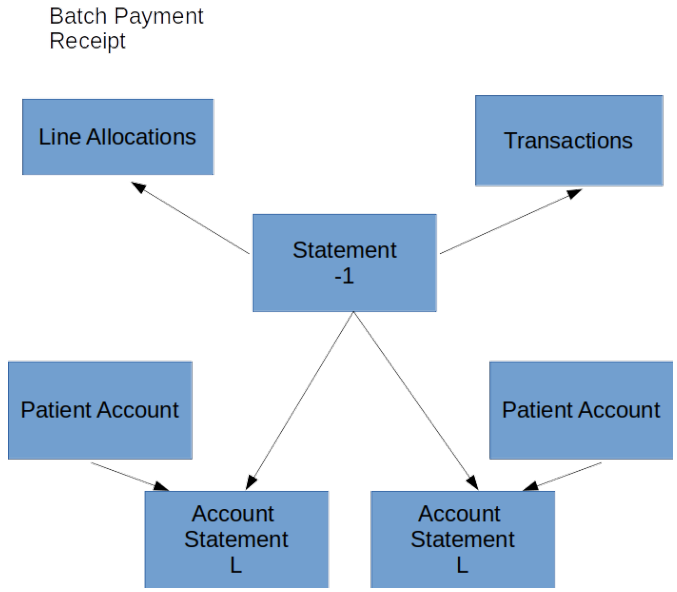
A receipt for a single account was represented in this structure.




A Combined Payment was modelled as below:

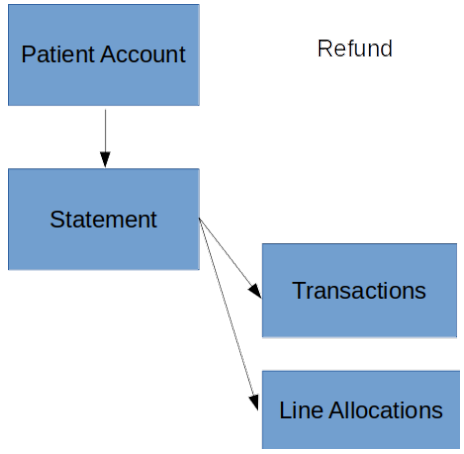



And a Batch Payment Receipt was a structure like this:




Problematically, this lead to a large number of 'statement' records with fundamentally different behaviours based on flags.

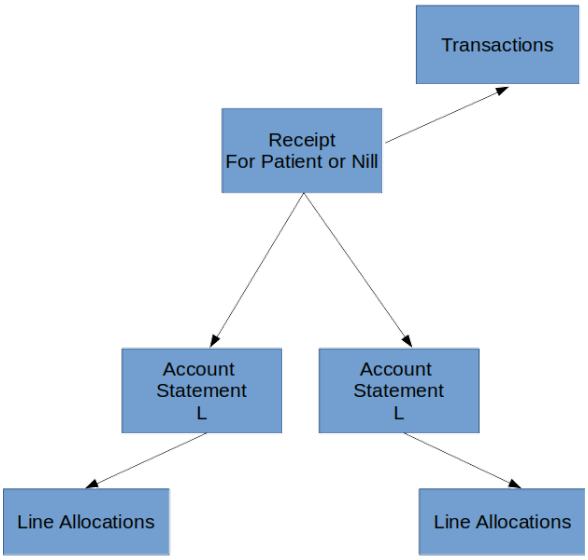
Refunds were similar:




### Current State - Receipt & Refund models

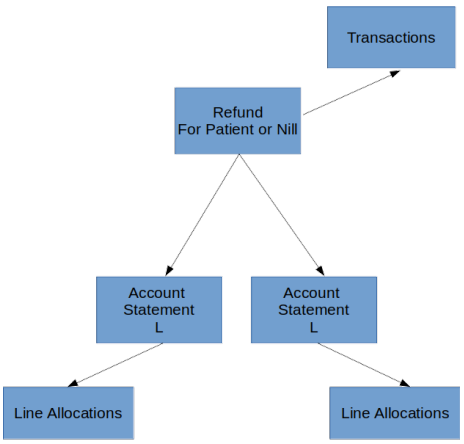
To simplify this, statement data was migrated to a cleaner Receipt structure.

Receipt Model




Location derived from transactions
Patient accounts on Account Statement Lines

Refund Model




Location derived from transactions
Patient accounts on Account Statement Lines

### Data Migration

For reference the following migration SQL was used; in this sequence. The SQL Server variation is used; with Postgres migrations being logically equivalent.

### Medicare Payment Reports

```
update a set
          receipt_id = b.stat_no
        from medicare_payment_reports a
          join f_statement b on a.statement_id = b.stat_no
        where b.stat_type = 'C' and b.stat_debit = 0 and b.stat_credit > 0
```

Convert Combined Payments

```
INSERT INTO receipts(
    id,
    [number],
    patient_id,
    guarantor_id,
    amount,
    [date],
    reversed,
    batch_account_id,
    note,
    detail,
    original_receipt_id,
    gst_deposit,
    medical_provider_id,
    created_at,
    updated_at
  )
select
 a.stat_no id,
 a.stat_ref [number],
 null patient_id,
 null guarantor_id,
 a.stat_credit amount,
 a.stat_date [date],
 case when a.stat_display = 0 and a.stat_status = 'A' then true else false end  reversed,
 nullif(a.stat_batch_number, '') AS batch_account_id,
 a.stat_note AS note,
 a.stat_detail detail,
 a.original_statement_id original_receipt_id,
 case when a.stat_gst_deposit = 1 then true else false end gst_deposit,
 a.stat_ass_numb medical_provider_id,
 coalesce(a.created_at, a.stat_date_created, a.stat_date) created_at,
 coalesce(a.updated_at, a.stat_date_created, a.stat_date) updated_at
from F_STATEMENT a
where a.stat_p_acc_ref = -2
```

1) Link Transactions to Receipts - SQL Server

```
        update a set
           receipt_id = c.id
        from f_transaction a
          join f_statement b on a.trans_stat_no = b.stat_no
          join receipts c on c.id = a.trans_stat_no
        where b.stat_p_acc_ref = -2
```

2) Link L Records to Receipts - SQL server

```
update a set
           receipt_id = c.id
        from f_statement a
          join F_STATEMENT b on a.stat_ref = b.stat_ref and b.stat_p_acc_ref = -2
          join receipts c on c.id = b.stat_no
        where a.stat_type = 'L' and a.stat_p_acc_ref not in (-1,-2,-3,-4,-5)
```

3) Add Reversal Reason to L record  - SQL server

```
        update a set
          reversal_reason = case when a.reversal_reason is null or a.reversal_reason = '' then b.reversal_reason else a.rev
ersal_reason end
        from f_statement a
          join F_STATEMENT b on a.stat_ref = b.stat_ref and b.stat_p_acc_ref in (-1, -2)
        where a.stat_type = 'L' and a.stat_p_acc_ref not in (-1,-2,-3,-4,-5) and a.stat_display = 0 and a.stat_status = 'A'
```

4) Delete old combined payments

```
delete from f_statement where stat_p_acc_ref = -2
```

### Convert Legecy receipts - Create

```
INSERT INTO receipts(
    id,
    [number],
    patient_id,
    guarantor_id,
    amount,
    [date],
    reversed,
    batch_account_id,
    note,
    detail,
    original_receipt_id,
    gst_deposit,
    medical_provider_id,
    created_at,
    updated_at
  )
select
 a.stat_no id,
 a.stat_ref [number],
 c.id patient_id,
 null guarantor_id,
 a.stat_credit amount,
 a.stat_date [date],
 case when a.stat_display = 0 and a.stat_status = 'A' then true else false end  reversed,
 nullif(a.stat_batch_number, '') AS batch_account_id,
 a.stat_note AS note,
 a.stat_detail detail,
 a.original_statement_id original_receipt_id,
 case when a.stat_gst_deposit = 1 then true else false end gst_deposit,
 a.stat_ass_numb medical_provider_id,
 coalesce(a.created_at, a.stat_date_created, a.stat_date) created_at,
 coalesce(a.updated_at, a.stat_date_created, a.stat_date) updated_at
from f_statement a
 join people b on a.stat_pt_numb = b.unique_id
 join patients c on c.person_id = b.id
where a.stat_credit > 0 and a.stat_type in ('C') and a.stat_p_acc_ref > 0 and a.stat_p_acc_ref not in(-1,-2,-3,-4,-5)
```

1) Link transactions to receipts

```
update a set
  receipt_id = c.id
from f_transaction a
 join f_statement b on a.trans_stat_no = b.stat_no
 join receipts c on c.id = a.trans_stat_no
where b.stat_credit > 0 and b.stat_type in ('C') and b.stat_p_acc_ref > 0 and b.stat_p_acc_ref not in(-1,-2,-3,-4,-5)
```

2) Modify statement records into Line records

```
update a set
  receipt_id = c.id,
  stat_type = 'L',
  was_receipt = #{Healthsolve::SQLHelper.sql_true}
from f_statement a
 join receipts c on c.id = a.stat_no
where a.stat_credit > 0 and a.stat_type in ('C') and a.stat_p_acc_ref > 0 and a.stat_p_acc_ref not in(-1,-2,-3,-4,-5)
```

### Convert Batch Payments

```
INSERT INTO receipts(
   id,
   [number],
   patient_id,
   guarantor_id,
   amount,
   [date],
   reversed,
   batch_account_id,
   note,
   detail,
   original_receipt_id,
   gst_deposit,
   medical_provider_id,
   created_at,
   updated_at
 )
select
 a.stat_no id,
 a.stat_ref [number],
 null patient_id,
 cast(b.batch_to as int) guarantor_id,
 a.stat_credit amount,
 a.stat_date [date],
case when a.stat_display = 0 and a.stat_status = 'A' then true else false end reversed,
 nullif(a.stat_batch_number, '') AS batch_account_id,
 a.stat_note AS note,
 a.stat_detail detail,
 a.original_statement_id original_receipt_id,
 case when a.stat_gst_deposit = 1 then true else false end gst_deposit,
 a.stat_ass_numb medical_provider_id,
 coalesce(a.created_at, a.stat_date_created, a.stat_date) created_at,
 coalesce(a.updated_at, a.stat_date_created, a.stat_date) updated_at
from f_statement a
left join F_BATCH_ACCOUNT b on b.batch_number = a.stat_batch_number
where a.stat_p_acc_ref = -1
```

1) Link transactions to receipts - SQL Server

```
update a set
  receipt_id = c.id
from f_transaction a
 join f_statement b on a.trans_stat_no = b.stat_no
 join receipts c on c.id = a.trans_stat_no
where b.stat_p_acc_ref = -1
```

2) Line allocations must be linked to the corresponding 'L' statement records  - SQL Server

```
update a set
  was_stat_no = b.stat_no,
  la_stat_no = c.stat_no
from f_line_alloc a
 join f_line_items f on a.la_line_no = f.line_no
 join f_statement b on a.la_stat_no = b.stat_no
 join (
      select a.la_stat_no, b.line_inv_no, sum(a.la_payment) payment  from f_line_alloc a
      join f_line_items b on a.la_line_no = b.line_no
      group by a.la_stat_no, b.line_inv_no
    ) d on d.la_stat_no = a.la_stat_no and f.line_inv_no = d.line_inv_no
 -- this does not take into account 2 statements for different invoices with the same payment amount
 join f_statement c on c.stat_type = 'L' and c.stat_p_acc_ref not in (-1,-2,-3,-4,-5) and b.stat_ref = c.stat_ref and c.
stat_pt_numb = a.la_pt_numb and c.stat_credit = d.payment
where b. stat_p_acc_ref < 0
```

3) Line allocation logs must be linked to the corresponding 'L' statement records.

3.1) Firstly update the currently valid ones

```
update a set
  was_stat_no = a.la_stat_no,
  la_stat_no = b.la_stat_no
from line_allocation_logs a
 join F_LINE_ALLOC b on a.la_no = b.la_no
where b.was_stat_no is not null
```

3.2) Now try and deal with the historical insert, deletes linked to the batch statement and not the corresponding 'L' statement record

If cant match patient_account, match patient.

```
update a set
  was_stat_no = a.la_stat_no,
  la_stat_no = coalesce(d.stat_no, f.stat_no)
from line_allocation_logs a
join f_statement b on a.la_stat_no = b.stat_no
join f_line_items c on c.line_no = a.la_line_no
join (select b.inv_pt_numb, b.INV_P_ACC_REF, a.*
      from (
           select a.la_stat_no, b.line_inv_no, sum(a.la_payment) payment  from line_allocation_logs a
           join f_line_items b on a.la_line_no = b.line_no
           group by a.la_stat_no, b.line_inv_no
          ) a
      join F_INVOICE b on a.LINE_INV_NO = b.inv_no
     ) e on a.la_stat_no = e.la_stat_no and c.LINE_INV_NO = e.LINE_INV_NO
left join f_statement d on d.stat_type = 'L' and d.stat_p_acc_ref not in (-1,-2,-3,-4,-5) and d.stat_ref = b.stat_ref an
d d.STAT_P_ACC_REF = e.INV_P_ACC_REF
left join f_statement f on f.stat_type = 'L' and f.stat_p_acc_ref not in (-1,-2,-3,-4,-5) and f.stat_ref = b.stat_ref and f.
stat_pt_numb = e.inv_pt_numb
where a.was_stat_no is null and b.stat_p_acc_ref < 0
```

4) Link L records to receipts

```
update a set
  receipt_id = c.id
from f_statement a
  join F_STATEMENT b on a.stat_ref = b.stat_ref and b.stat_p_acc_ref = -1
  join receipts c on c.id = b.stat_no
where a.stat_type = 'L' and a.stat_p_acc_ref not in (-1,-2,-3,-4,-5)
```

5)Update allocations that are over allocated

```
update a set
   was_stat_no = coalesce(a.was_stat_no, a.la_stat_no),
   la_stat_no = b.new_stat_no
from  f_line_alloc a
join (
     -- these are the line allocations that need to change
     select x. la_no, x.LA_STAT_NO bad_stat_no, f.STAT_NO new_stat_no from (
     select row_number() over (partition by w.receipt_id order by w.stat_pt_numb desc ) rn, v.*
     from F_LINE_ALLOC v
     join f_statement w on v.la_stat_no = w.stat_no
     where v.la_stat_no in (select a.stat_no from F_STATEMENT a
     join receipts b on a.receipt_id = b.id
     left join (select sum(la_payment) payment, la_stat_no from F_LINE_ALLOC group by la_stat_no) c on c.LA_STAT_
NO = a.stat_no
     where b.batch_account_id is not null and b.batch_account_id <> '' and a.stat_display <> 0 and a.stat_status <
> 'A'
     and (c.payment <> a.stat_credit))
     ) x
     join f_statement b on x.la_stat_no = b.stat_no
     join (select row_number() over (partition by a.receipt_id order by a.stat_pt_numb desc ) + 1 rn, a.receipt_id, a.st
at_no, a.stat_credit, a.stat_pt_numb from F_STATEMENT a
     join receipts b on a.receipt_id = b.id
     left join (select sum(la_payment) payment, la_stat_no from F_LINE_ALLOC group by la_stat_no) c on c.LA_STAT_
NO = a.stat_no
     where b.batch_account_id is not null and b.batch_account_id <> '' and a.stat_display <> 0 and a.stat_status <
> 'A'
     and c.payment is null
     and b.id in (select b.id from F_STATEMENT a
     join receipts b on a.receipt_id = b.id
     left join (select sum(la_payment) payment, la_stat_no from F_LINE_ALLOC group by la_stat_no) c on c.LA_STAT_
NO = a.stat_no
     where b.batch_account_id is not null and b.batch_account_id <> '' and a.stat_display <> 0 and a.stat_status <
> 'A'
     and (c.payment <> a.stat_credit))
     ) f on f.receipt_id = b.receipt_id and f.rn = x.rn and f.STAT_CREDIT = b.STAT_CREDIT and f.stat_pt_numb = b.ST
AT_PT_NUMB
     where x.rn > 1
    ) b on a.la_no = b.la_no and a.la_stat_no = b.bad_stat_no
```

6) Lastly ensure line_allocation_logs match f_line_alloc

```
update a set
  was_stat_no = a.la_stat_no,
  la_stat_no = b.la_stat_no
from line_allocation_logs a
  join F_LINE_ALLOC b on a.la_no = b.la_no
where b.was_stat_no is not null
```

7) Delete old batch payments

```
delete from f_statement where stat_p_acc_ref = -1
```

Refunds

Medicare Payment Reports

```
update a set
           refund_id = b.stat_no
         from medicare_payment_reports a
           join f_statement b on a.statement_id = b.stat_no
         where b.stat_type = 'C' and b.stat_debit > 0 and b.stat_credit = 0
```

Convert Refunds - Create

```
INSERT INTO refunds(
    id,
    [number],
    patient_id,
    guarantor_id,
    amount,
    [date],
    reversed,
    note,
    detail,
    receipt_id,
    medical_provider_id,
    created_at,
    updated_at
  )
select
 a.stat_no id,
 a.stat_ref [number],
 c.id patient_id,
 null guarantor_id,
 a.stat_debit amount,
 a.stat_date [date],
 case when a.stat_display = 0 and a.stat_status = 'A' then true else false end  reversed,
 a.stat_note AS note,
 a.stat_detail detail,
 d.id receipt_id,
 a.stat_ass_numb medical_provider_id,
 coalesce(a.created_at, a.stat_date_created, a.stat_date) created_at,
 coalesce(a.updated_at, a.stat_date_created, a.stat_date) updated_at
from F_STATEMENT a
join people b on a.stat_pt_numb = b.unique_id
join patients c on c.person_id = b.id
left join (select *,
           row_number() over (partition by [number]} order by updated_at desc) rn
           from receipts
         ) d on d.[number] = a.stat_xref_rec_no and d.rn=1
where a.stat_type = 'C' and a.stat_debit > 0 and a.stat_credit = 0
```

1) Link Transactions to Refunds

```
update a set
           refund_id = c.id
         from f_transaction a
           join f_statement b on a.trans_stat_no = b.stat_no
           join refunds c on c.id = a.trans_stat_no
         where b.stat_type = 'C' and b.stat_debit > 0 and b.stat_credit = 0
```

2) Modify statement records into Line records

```
update a set
  refund_id = c.id,
  stat_type = 'L',
  was_refund = true
from f_statement a
 join refunds c on c.id = a.stat_no
where a.stat_type = 'C' and a.stat_debit > 0 and a.stat_credit = 0
```

### New Tables

- Receipts - do not access directly, use v_receipts
- Refunds - do not access directly, use v_refunds

### New Columns

| Table | Column |
|---|---|
| medicare_payment_reports | medicare_processing_report_id - Added index |
| medicare_payment_reports | claim_id - Added index |
| medicare_payment_reports | claim_transaction_id - Added index |
| service_configuration_mappings | service_configuration_id - Added index |
| service_configuration_mappings | referral_id - Added index |
| service_configuration_mappings | health_fund_id - Added index |
| service_configuration_mappings | professional_category_id - Added index |

### Changed Views

| Table | Change |
|---|---|
| v_age_accounts | Altered source of data |
| v_ledger | Altered source of data |
| v_transaction_links | Altered source of data |
| v_reconcillations | Altered source of data |

### Changed Columns

- Service Configuration Mappings - health_fund_id, service_configuration_id, referral_id, professional_category_id all have indexes now

### Deleted Tables

- None

### Deleted Columns

- None