6.97 Schema Changes

Last Modified on 09/08/2024 11:26 am ACST

Schema changes between CareRight V6.96.3 and 6.97

Healthlink Messages

All Healthlink NZ messages are migrated to the Secure Message schema.

```
insert into secure_messages(
                                vendor,
                                message_type,
                                message_control_id,
                                data,
                                created_at,
                                updated_at,
                                directory_id,
                                md5
                                )
           SELECT
                    'healthlink-nz' AS vendor,
                    'REF^I12' AS message_type,
                    a.control_id AS message_control_id,
                    a.message AS data,
                    a.created_at,
                    a.updated_at,
                    a.directory_id,
                    MD5(a.message)
                  FROM
                    healthlink_messages a
                    join correspondences b on a.correspondence_id = b.id
                    where b.received_at is not null ;

           update correspondences s set
              message_type = 'SecureMessage',
              message_id = c.id
          from correspondences a
            join healthlink_messages b on a.message_id = b.id
            join secure_messages c on b.control_id = c.message_control_id and c.vendor = 'healthlink-nz' and c.md5 =
MD5(b.message)
            where a.message_type = 'HealthlinkMessage'
              and a.id = s.id;

          update correspondences s SET
            message_control_id = b.control_id -- control_id is the id that ACK will come back on
          from correspondences a
            join healthlink_messages b on a.id= b.correspondence_id
          where a.received_at is null -- outgoing messages only
            and a.id = s.id;
```

```
UPDATE healthlink_import_dirs SET vendor = 'healthlink-nz' WHERE vendor IS NULL
```

```
update clinical_specialties set uri = 'https://healthterminologies.gov.au/fhir/ValueSet/clinical-specialty-1' where coding
_system = 'http://snomed.info/sct'
```

Change of Roles

All users with admin_role will be granted calendar_admin, accounting_admin, correspondence_admin.

```
SecureApplicationSchema.create_roles


  admin_role = Role.find_by!(name: "admin_role")
  %w[
    calendar_admin
    accounting_admin
    correspondence_admin
  ].each do |new_role_name|
    new_role = Role.find_by!(name: new_role_name)
    GroupRole.where(role_id: admin_role.id).each do |group_role|
      next unless group_role.group


      unless group_role.group.roles.where(name: new_role_name).any?
        group_role.group.roles << new_role
        group_role.group.save!(validate: false)
      end
    end
  end
```

New Tables/Views

- ihc_misc_codes
- relationship_assessment_snapshots
- information_classifications

New Columns

- healthlink_import_dir..vendor
- provider_directories.api_key
- clinical_specialties.uri
- patient_measurements.source_object_type
- patient_measurements.source_object_id
- letters.abnormal
- letters.correction

New Indexes

- None

Changed Views

- None

Changed Columns/Indexes

- None

Deleted Tables/Views

- None

Deleted Columns/Indexes

- None

Schema changes between CareRight 6.97 and 6.97.18

New tables

- medicare_era_links

Other

Medicare data migrated from a singular to one to many relationship:

update a set medicare_online_request_id = m.medicare_online_request_id

from medicare_era_reports a

join (select * from (

select a.id, b.medicare_online_request_id, ROW_NUMBER() OVER (partition BY b.medicare_era_report_id order by b.id) AS rn

from medicare_era_reports a

join medicare_era_links b on a.id = b.medicare_era_report_id

) x where x.rn = 1) m on a.id = m.id

Schema changes between CareRight 6.97.18 and 6.97.21

New view - v_line_allocation_logs

Schema changes between CareRight 6.97.21 and 6.97.23

New Columns

- current_assessments.submitted_for_approval_by
- current_assessments.submitted_for_approval_at

- entitlement_usages.archived_at

Other

Sql server

```
update a set current_information_classification_level = m.current_classification
        from people a
        join (
              select a.id,
                    coalesce(b.highest_classification_level, 0) current_classification#{" "}
              from people a#{" "}
                left join (select b.id, max(classification_level) highest_classification_level#{" "}
                          from information_classifications a#{" "}
                            join people b on b.id = a.classificationable_id and a.classificationable_type = 'Person'
                          where (a.archived is null or a.archived = 0)
                            and a.classified_from < CURRENT_TIMESTAMP
                            and (a.classified_until IS NULL OR a.classified_until > CURRENT_TIMESTAMP)
                          group by b.id
                         ) b on a.id = b.id
 ) m on a.id = m.id
```

Postgres

```
UPDATE people a
        SET current_information_classification_level = m.current_classification
        from people b
        join (
              select a.id,
                    coalesce(b.highest_classification_level, 0) current_classification#{" "}
              from people a#{" "}
                left join (select b.id, max(classification_level) highest_classification_level#{" "}
                          from information_classifications a#{" "}
                            join people b on b.id = a.classificationable_id and a.classificationable_type = 'Person'
                          where (a.archived is null or a.archived = 0)
                            and a.classified_from < CURRENT_TIMESTAMP
                            and (a.classified_until IS NULL OR a.classified_until > CURRENT_TIMESTAMP)
                          group by b.id
                         ) b on a.id = b.id

             ) m on m.id = b.id
        WHERE a.id = b.id
```