6.98 Schema Changes

Last Modified on 09/08/2024 11:25 am ACST

Schema changes between CareRight V6.97 and 6.98

New Tables/Views

- None

New Columns

- case_categories.checklist_id
- case_categories.entitlement_category_id
- cases.entitlement_id
- data_collection_elements.primary_meteor_code
- data_collection_elements.meteor_registration_status
- statutory_report_elements.meteor_code
- statutory_report_elements.meteor_registration_status
- case_assessment_snapshots.checklist_instance_id
- cases.checklist_instance_id
- entitlement_usages.appointment_id
- signatures.invoice_id
- signatures.signer_name
- appointment_types.checklist_id
- person_detail_setups.external_platform_used
- person_detail_setups.external_platform_name
- person_detail_setups.external_platform_attr_name
- external_patient_tokens.appointment_id
- calendar_settings.default_confirmed_appointment_status_id

New Indexes

- None

Changed Views

- None

Changed Columns/Indexes

- report_results.message - Now limit 1000

Deleted Tables/Views

- None

Deleted Columns/Indexes

- None

Schema changes between CareRight 6.98 and 6.98.3

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 6.98.3 and 6.98.7

New Columns

- entitlements.notes
- appointment_types.show_cancellation_reason
- appointments.cancellation_reason_id
- providers.mo_routing_id
- current_assessments.submitted_for_approval_by
- current_assessments.submitted_for_approval_at
- entitlement_usages.archived_at
- people.current_information_classification_level

New Indexes

- correspondences.created_at
- documents.created_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
```