WEBINAR

ODS / API Change Queries Deep Dive

Tuesday, Oct. 30 | 1 pm CST

Presenters:

Vinaya Mayya, *Ed-Fi Software Development Lead*

Sayee Srinivasan, *Ed-Fi Solutions Architect*

Change queries allow client systems to narrow requests for data. Join Ed-Fi Technical Leads for a lab demonstration on how you can use change queries.

Welcome!

- 30 minute presentation, 15 minutes of Q&A
- Use the Webex Q&A feature to submit a question at anytime

What We'll Cover

- What are change queries?
- Change Tracking and Change Data Capture
- Understanding Ed-Fi's implementation of change queries
- How to enable this feature
- How to use this feature

Hello!

Caroline Kazmierski
Director of Communications & Mktg
- Three years @ Ed-Fi
- Tech PR expert
- Best Summer Vacation Spot: Greece
- Contact: Caroline.Kazmierski@ed-fi.org
- Twitter: @edfialliance

Sayee Srinivasan
Solutions Architect
- Two years @ Ed-Fi
- Solutions Architect – SEA Support
- Best Summer Vacation Spot: India
- Contact: Sayee.Srinivasan@ed-fi.org

Vinaya Mayya
Software Development Lead
- Two years @ Ed-Fi
- 15 + years in software development
- Best Summer Vacation Spot: Andaman
- Contact: vinaya.mayya@ed-fi.org

Why do we need the change queries feature?

- No longer need to complete a full data sync
- Inefficient
- Time consuming
- Performance issues

Change queries will fix these problems!

When to use the change queries?

- Change queries is a feature that allows client systems to narrow requests for data to only data that has changed since a specified point in time.
- It allows clients systems to stay in sync with the ODS/API without having to pull a complete data set.
- Released with ODS/API v3.1 (Dec 2018). It is an optional feature.
  - Analytics Systems Vendors can use this feature to stay in sync
  - Datawarehouse product vendors can do the ODS-to-ODS sync
  - SIS vendors can use this for their reconciliation process

Change Tracking

Change tracking is a lightweight solution that help answer -What rows have changed for a table?

- Tracks only the fact that a row has changed, not how many times the row has changed or the values of any intermediate changes.
- Tracks the primary key of changed data as part of ongoing transactions to indicate when a row in a tracked table has been changed.
- The change tracking feature first needs to be enabled at the database level and then for each table where you want to track the changes.
- No changes to the table schema are required, but existing application code needs to be updated to take advantage of CT.

Change Data Capture

Change data capture (CDC) records insert, update, and delete activity that is applied to a SQL Server table asynchronously based on the transaction log.

- The CDC feature first needs to be enabled at the database level and then for each table where you want to track the changes.
- Once a table in a database is enabled for change data capture all changes to that table are tracked by storing changes in a change table. The change table will contain:
  - One record for every INSERT that can be used to identify column values for the inserted records.
  - One record for each DELETE that will show the values in each column prior to the DELETE.
  - Two records for UPDATES, one with the updated column values and one with the original column values.

Comparing CT and CDC

Feature	Change Tracking	Change Data Capture
Supported Editions	Express, Workgroup, Web, Standard, Enterprise, DataCenter	DataCenter, Enterprise
Summary	Tracks when a row/column has changed.	Tracks when data has changed and includes the values as well. Entire table or subset of columns can be captured.
Methodology	Synchronous with DML, records change tracking info as part of transaction.	Asynchronous from Transaction Log and requires SQL Serer Agent.
Recommendations	CT is well suited for applications that require notice of a database change, but which don't need a change history. E.g. Analytics applications that only require current data, or an application that keeps separate databases in sync.	CDC is suited for large scale data warehouse applications where maintaining historical data is important. E.g. Data warehouse load process that need to identify changes, so they can correctly apply updates to track historical changes over time.

Does Ed-Fi Change Query Feature use any SQL Server provided change tracking features?

No

- CT would have better suited the use cases we were targeting.
- For the existing API, CT based solution still would have led to complex design. So we leveraged a simple designed based on already tracked last modified date.
- The community demand for open source database informed us to **avoid the dependency on SQL Server specific features**.

Change Query Implementation

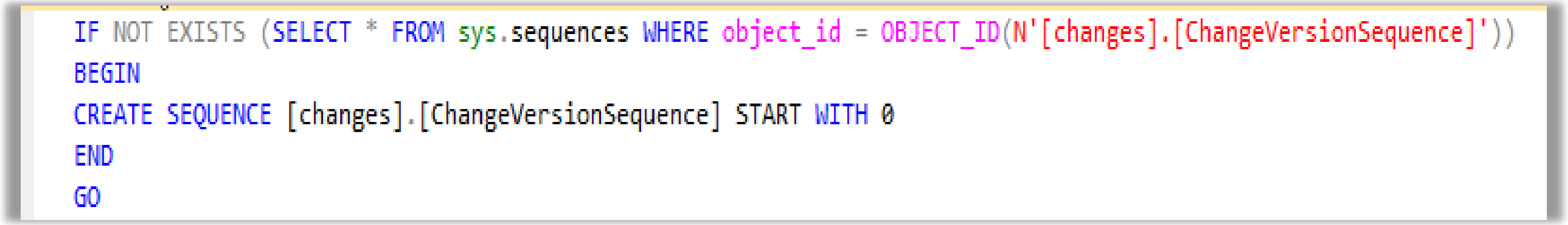
- Custom solution based on already tracked LastModifiedDate.
- Database Sequence Object is used for ease of API usage.
- Adds a change version column and update/delete triggers to top level entities/resources for change tracking
- Uses existing authorization models used for data management APIs

Database Artifacts

- Sequence
- Change Version Column
- Tracked Delete Tables
- Delete and Update Triggers

Change Query Sequence Object

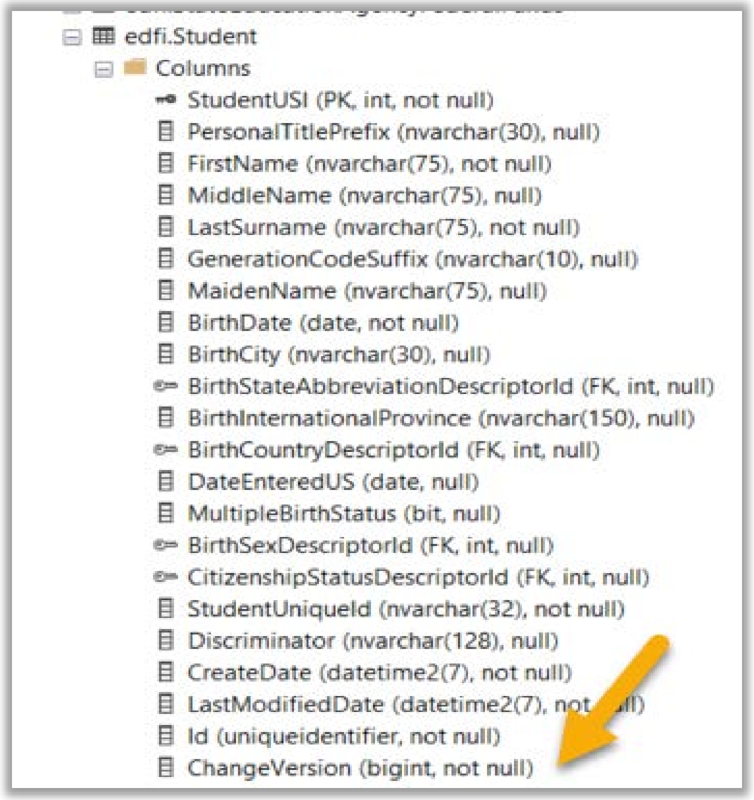
SQL Sequence, is a database object that could be used for generating number sequence. It is supported by many database management systems.
We use it for tracking update version in place of LastModifiedDate



```
IF NOT EXISTS (SELECT * FROM sys.sequences WHERE object_id = OBJECT_ID(N'[changes].[ChangeVersionSequence]'))
BEGIN
CREATE SEQUENCE [changes].[ChangeVersionSequence] START WITH 0
END
GO
```

Change Version Column

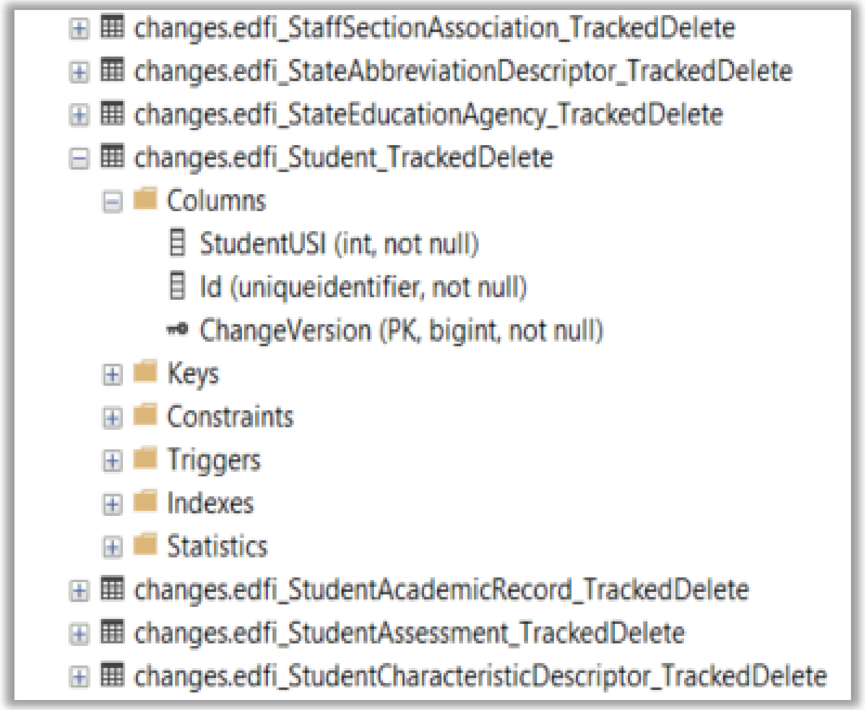
Added to top level entities/resources for change tracking



edfi.Student
Columns
StudentUSI (PK, int, not null)
PersonalTitlePrefix (nvarchar(30), null)
FirstName (nvarchar(75), not null)
MiddleName (nvarchar(75), null)
LastSurname (nvarchar(75), not null)
GenerationCodeSuffix (nvarchar(10), null)
MaidenName (nvarchar(75), null)
BirthDate (date, not null)
BirthCity (nvarchar(30), null)
BirthStateAbbreviationDescriptorId (FK, int, null)
BirthInternationalProvince (nvarchar(150), null)
BirthCountryDescriptorId (FK, int, null)
DateEnteredUS (date, null)
MultipleBirthStatus (bit, null)
BirthSexDescriptorId (FK, int, null)
CitizenshipStatusDescriptorId (FK, int, null)
StudentUniqueId (nvarchar(32), not null)
Discriminator (nvarchar(128), null)
CreateDate (datetime2(7), not null)
LastModifiedDate (datetime2(7), not null)
Id (uniqueidentifier, not null)
ChangeVersion (bigint, not null)

Tracked Delete Tables

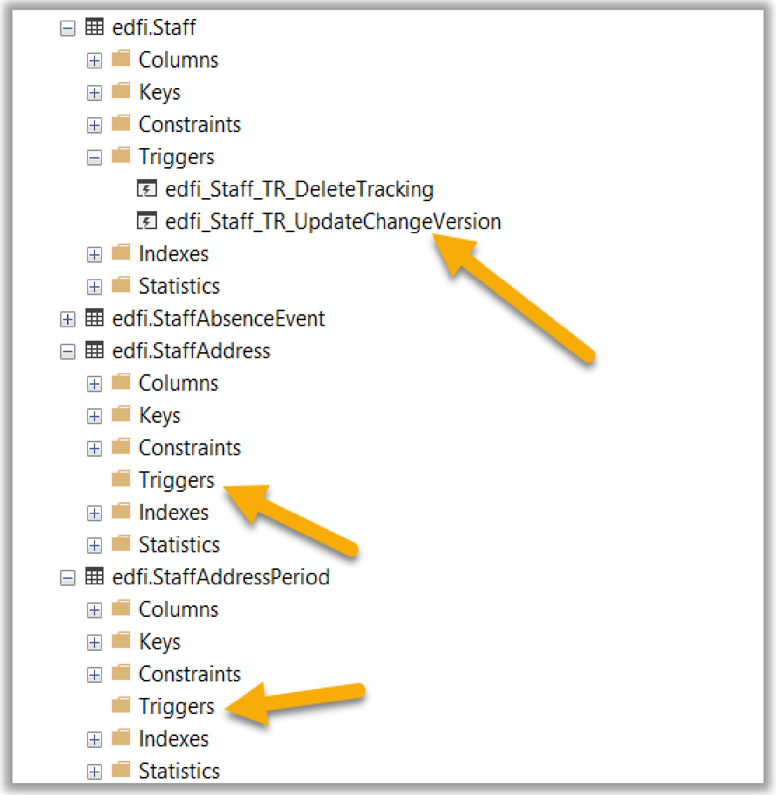
Added to top level entities/resources for tracking deletes



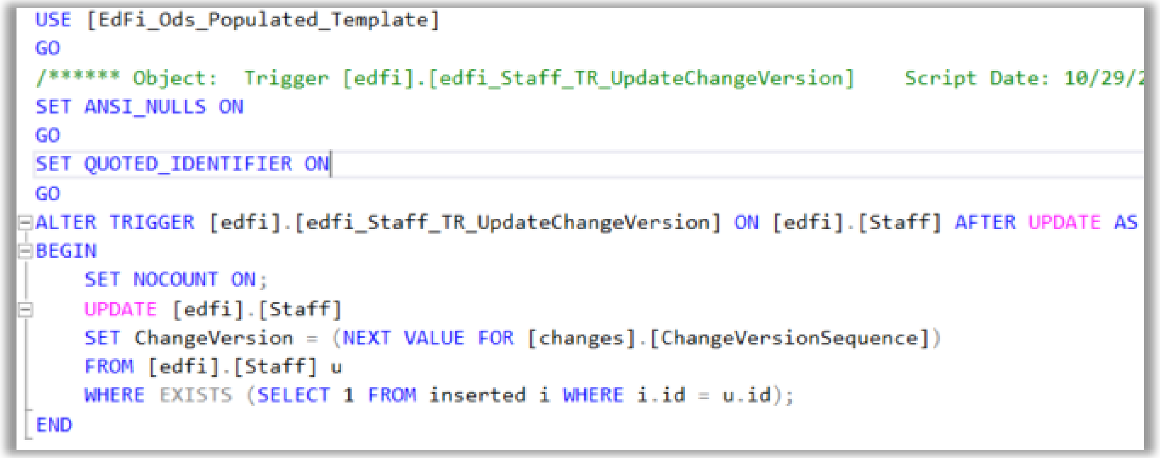
```
⊞ changes.edfi_StaffSectionAssociation_TrackedDelete
⊞ changes.edfi_StateAbbreviationDescriptor_TrackedDelete
⊞ changes.edfi_StateEducationAgency_TrackedDelete
⊟ changes.edfi_Student_TrackedDelete
    ⊟ Columns
        StudentUSI (int, not null)
        Id (uniqueidentifier, not null)
        ChangeVersion (PK, bigint, not null)
    ⊞ Keys
    ⊞ Constraints
    ⊞ Triggers
    ⊞ Indexes
    ⊞ Statistics
⊞ changes.edfi_StudentAcademicRecord_TrackedDelete
⊞ changes.edfi_StudentAssessment_TrackedDelete
⊞ changes.edfi_StudentCharacteristicDescriptor_TrackedDelete
```

Delete and Update Triggers

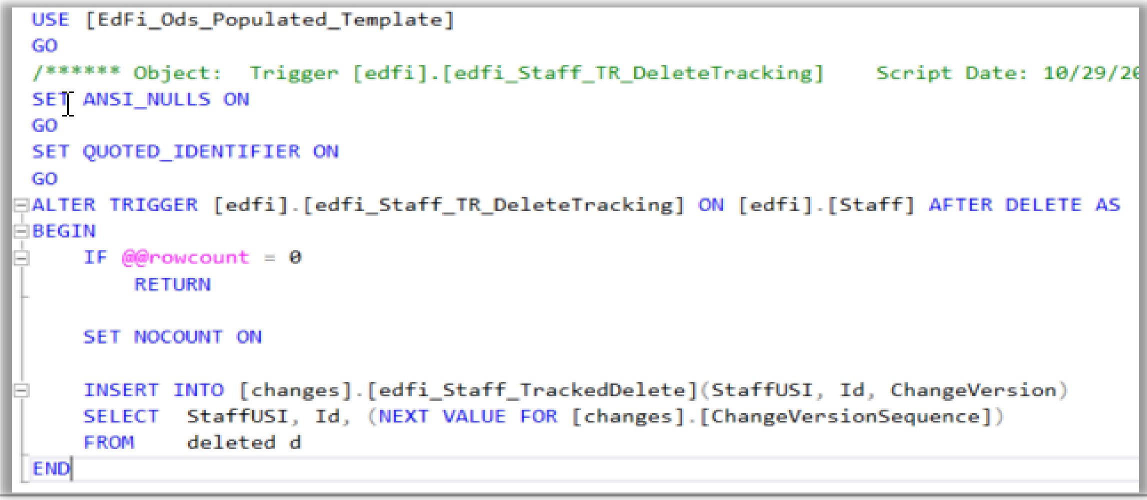
Added to top level entities/resources for tracking deletes



```
⊟ ▦ edfi.Staff
    ⊞ Columns
    ⊞ Keys
    ⊞ Constraints
    ⊟ Triggers
        edfi_Staff_TR_DeleteTracking
        edfi_Staff_TR_UpdateChangeVersion
    ⊞ Indexes
    ⊞ Statistics
⊞ ▦ edfi.StaffAbsenceEvent
⊟ ▦ edfi.StaffAddress
    ⊞ Columns
    ⊞ Keys
    ⊞ Constraints
      Triggers
    ⊞ Indexes
    ⊞ Statistics
⊟ ▦ edfi.StaffAddressPeriod
    ⊞ Columns
    ⊞ Keys
    ⊞ Constraints
      Triggers
    ⊞ Indexes
    ⊞ Statistics
```



```
USE [EdFi_Ods_Populated_Template]
GO
/****** Object:  Trigger [edfi].[edfi_Staff_TR_UpdateChangeVersion]    Script Date: 10/29/2
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [edfi].[edfi_Staff_TR_UpdateChangeVersion] ON [edfi].[Staff] AFTER UPDATE AS
BEGIN
    SET NOCOUNT ON;
    UPDATE [edfi].[Staff]
    SET ChangeVersion = (NEXT VALUE FOR [changes].[ChangeVersionSequence])
    FROM [edfi].[Staff] u
    WHERE EXISTS (SELECT 1 FROM inserted i WHERE i.id = u.id);
END
```



```
USE [EdFi_Ods_Populated_Template]
GO
/****** Object:  Trigger [edfi].[edfi_Staff_TR_DeleteTracking]    Script Date: 10/29/20
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [edfi].[edfi_Staff_TR_DeleteTracking] ON [edfi].[Staff] AFTER DELETE AS
BEGIN
    IF @@rowcount = 0
        RETURN

    SET NOCOUNT ON

    INSERT INTO [changes].[edfi_Staff_TrackedDelete](StaffUSI, Id, ChangeVersion)
    SELECT  StaffUSI, Id, (NEXT VALUE FOR [changes].[ChangeVersionSequence])
    FROM    deleted d
END
```

Enabling Change Queries Feature

- The feature is enabled on the deployed code by changing the Web.config file of the EdFi.Ods.WebApi project.

```
<add key="changeQueries:featureIsEnabled" value="true" />
```

Deploying Database Artifacts for Change Queries Feature

In Development Environment

initdev process automatically deploys the change query schema based on the "changeQueries:featureIsEnabled" flag in the Web.config

Deploying Database Artifacts for Change Queries Feature Cont.

In non-development environments,

Download database artifacts from: https://www.myget.org/feed/ed-fi/package/nuget/EdFi.RestApi.Databases.EFA/3.2.0

Update the Databases.config to set the "changeQueries:featureIsEnabled" flag to "true".

Run Powershell database deployment scripts:

```
Import-Module .\Deployment.psm1
Initialize-DeploymentEnvironment -InstallType SharedInstance -PathResolverRepositoryOverride “Ed-Fi-Common;Ed-Fi-ODS;Ed-Fi-ODS-Implementation”
```

API Details

There is a (global) AvailableChangeVersions API resource that provides information on current change version. This resource allows clients to request changes they have not already requested and/or processed.

```
GET /ChangeQueries/v1/availableChangeVersions
```

API clients can then use a query syntax to request newer changes on any resource

```
GET /data/v3/ed-fi/students?minChangeVersion=234378&maxChangeVersion=234974
```

API clients can get deletes on any resource via new route

```
GET /data/v3/ed-fi/students/deletes?minChangeVersion=234378&maxChangeVersion=234974
```

Client Sync Process

Obtained from the availableChangeVersions endpoint

```
GET /ChangeQueries/v1/availableChangeVersions
```

Client does a Full Sync

First on each resource according to resource dependency order (for creations and edits)

```
GET /data/v3/ed-fi/students?maxChangeVersion=234738
```

Then on each resource in reverse dependency order (for deletes)

```
GET /data/v3/ed-fi/students/deletes?maxChangeVersion=234738
```

Subsequent sync queries would use the previous change version value, and retrieve a new one at the start of processing

```
GET /data/v3/ed-fi/students?minChangeVersion=234378&maxChangeVersion=234974
GET /data/v3/ed-fi/students/deletes?minChangeVersion=234378&maxChangeVersion=234974
```

Snapshot Isolation

- Change Queries solution does not aim at achieving immediate consistency but best attempts at achieving eventual consistency.
- For 100% consistency it is highly recommended to use a second static database to serve the changes.

Performance Consideration

- Additional Triggers tracked data will have some impact on database performance so do the SAL server features like CDC, CT —the change data must be stored somewhere.
- Our performance analysis of the API when the change query feature was enabled didn’t show significant impact. https://techdocs.ed-fi.org/pages/viewpage.action?pageId=64687244

Resources

- Changed Record Queries Technical Article
- Using the Changed Record Queries
- PowerShell modules: https://techdocs.ed-fi.org/display/EFTD/Initialize-DeploymentEnvironment

Feedback

- More questions? Submit a tracker ticket
- Slack
- Would love to hear about your production use case!

ed-fi®

ALLIANCE