Triggern.
Change Data Capture

Triggers

- Was sind Triggers?
  - Eine bestimmte Art von gespeicherte Prozedur, die automatisch ausgeführt wird wenn eine DML oder DDL Anweisung ausgeführt wird
  - Eine Menge von Aktionen, die durch das Eintreten eines Ereignisses angestoßen werden
- Beispiel von ausgelösten Aktionen (Effekt des Triggers)
  - Änderungen an anderen Tabellen
  - Erzeugung bzw. Transformation von Werten für eingefügte bzw. Geänderte Werte
  - Aufrufen von Datenbankfunktionen

Triggers

- Wann werden Triggers benutzt?
  - Datenbankzusicherungen (constraints) sichern einen Zustand einer Datenbank
  - Datenbanktrigger werden benutzt, um Aktionen auszuführen, falls bestimmte Bedingungen bzgl. des Zustand einer Datenbank zutreffen
- Ereignisse, die Trigger feuern
  - DML Anweisungen: INSERT, UPDATE, DELETE
  - DDL Anweisungen: CREATE DATABASE, DROP LOGIN, UPDATE STATISTICS, DROP TRIGGER, ALTER TABLE

Triggers

- Trigger-Konzept
  - ECA-Regeln: Event-Condition-Action Rules
- Event
  - Auslösendes Ereignis einer Aktion
- Condition
  - Bedingung zum Auslösen der Aktion
- Action
  - Angabe der Aktion (Folge von Datenbankoperationen)

Anlegen von Triggern

```
CREATE TRIGGER <trigger-name>
ON {table | view}
[WITH <dml_trigger_option> [,…n] ]
{FOR | AFTER | INSTEAD OF}
{ [INSERT] [,] [UPDATE] [,] [DELETE] }
[ WITH APPEND ] [NOT FOR REPLICATION ]
AS
     {sql_statement [;] [,…n] |
EXTERNAL NAME <method specifier [;] > }
```

Auslösezeitpunkt

Auslöser

Aktion

Triggern

- Auslösezeitpunkt
  - `FOR`
  - `AFTER`
  - `INSTEAD OF`
- Wenn mehrere Triggers für dieselbe Operation definiert werden, dann werden diese in beliebige Reihenfolge ausgeführt
- Wenn ein Trigger ausgeführt wird, stehen 2 spezielle Tabellen zur verfügung: `inserted` und `deleted`

Trigger

```
CREATE TRIGGER [dbo].[On_Product_Insert]
   ON [dbo].[Products]
   AFTER INSERT
AS
BEGIN
     SET NOCOUNT ON;

     insert into LogBuys (Name, Date, Quantity)
     select Name, GETDATE(), Quantity
     from inserted
END
```

Trigger

```
CREATE TRIGGER [dbo].[On_Product_Delete]
   ON [dbo].[Products]
   AFTER DELETE
AS
BEGIN
      SET NOCOUNT ON;

      insert into LogSells (Name, Date, Quantity)
      select Name, GETDATE(), Quantity
      from deleted
END
```

Triggers

```
ALTER TRIGGER [dbo].[On_Product_Update]
   ON [dbo].[Products]
   AFTER UPDATE
AS
BEGIN
      SET NOCOUNT ON;
      insert into LogSells (Name, Date, Quantity)
      select d.Name, GETDATE(), d.Quantity - i.Quantity
      from deleted d inner join inserted i on d.ID=i.ID
      where i.Quantity < d.Quantity

      insert into LogBuys (Name, Date, Quantity)
      select i.Name, GETDATE(), i.Quantity - d.Quantity
      from deleted d inner join inserted i on d.ID=i.ID
      where i.Quantity > d.Quantity
END
```

Triggers

- `SET NOCOUNT ON/OFF`
  - `ON` – die Anzahl der betroffenen Zeilen wird nicht als Teil des Resultsets zurückgegeben
  - `OFF` – die Anzahl wird zurückgegeben
- @@ROWCOUNT
  - Anzahl der Zeilen auf die sich die letzte Anweisung ausgewirkt hat
  - wird immer aktualisiert (auch wenn `NOCOUNT ON` ist)

Triggers

- Entfernen von Triggern

```
DROP TRIGGER <trigger-name>
```

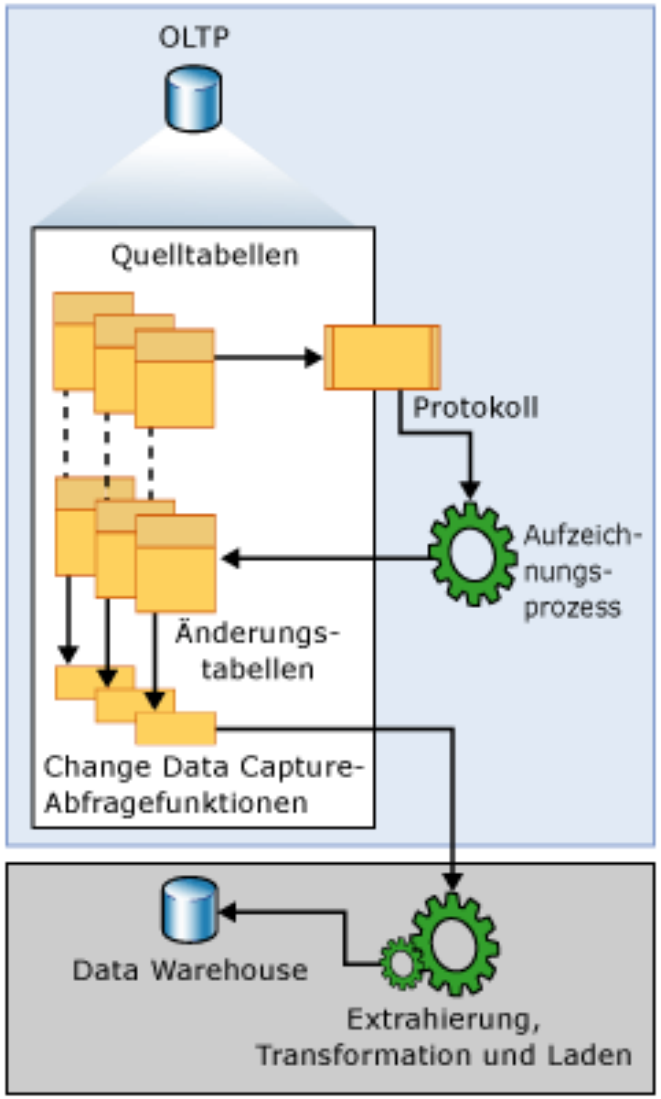
Anwendung von Triggern

- Definition und Umsetzung von Anwendungslogik („business rules“) → **`AFTER`**-Trigger
  - Bsp. Erhöhung der Gehälter um maximal 10%
  - Änderung der Anwendungslogik ohne Änderung der Anwendung
  - Wird dazu verwendet, Änderungen an der Datenbasis vorzunehmen bzw. Externe Funktionen (zB Fehlermeldungen) auszuführen
- Erweiterte Integritätsprüfung
  - Überprüfung der Eingabedaten auf Gültigkeit
  - Automatisches Generieren von Werten für neu eingefügte Tupel
  - Lesen von anderen Tabellen zur Evaluierung von Querverweisen

Aufgabe

- Für die Tabelle **Studenten(MatrNr, Name, Vorname, Geburtsdatum, Email)** wollen wir bei dem Einfügen Folgendes überprüfen:
  - '1990-1-1' <= Geburtsdatum <= '2002-1-1'
- Implementiere das einmal als **Integritätsregel** und einmal als **Trigger**, sodass Tupeln, welche die Regel nicht einhalten, nicht eingefügt werden

Change Data Capture



- Details zu Änderungen (Einfüge- , Aktualisierungs- und Löschaktivitäten auf die Tabellen) werden in einem leicht verwendbaren relationalen Format bereitgestellt
- Wurde in SQL Server 2008 eingeführt, ist aber nur in dem Enterprise Edition verfügbar

Change Data Capture (CDC)

- `sys.sp_cdc_enable_db`
  - Change Data Capture wird explizit für die Datenbank aktiviert (damit Änderungen an einzelnen Tabellen nachverfolgt werden können)
- `Sys.sp_cdc_enable_table`
  - Wenn die Datenbank für Change Data Capture aktiviert ist, dann können Quelltabellen als nachverfolgte Tabellen identifiziert werden
- Für mehrere Informationen: https://msdn.microsoft.com/de-de/library/cc645937.aspx

OUTPUT Klausel

- Gibt Informationen aus bzw. Ausdrücke basierend auf den einzelnen Zeilen zurück, auf die eine INSERT-, UPDATE-, DELETE- oder MERGE-Anweisung Auswirkungen hat
- Bietet Zugriff zu den `inserted` oder `deleted` Tabellen

OUTPUT Klausel

```
UPDATE Categories
SET CategoryName = 'Dried Produce'
OUTPUT inserted.CategoryId,
       deleted.CategoryName,
       inserted.CategoryName,
       SUSER_SNAME()
INTO CategoryChanges
WHERE CategoryId = 7
```

gibt den Username aus

MERGE Klausel

- Führt Einfüge-, Aktualisierungs- oder Löschvorgänge in einer Zieltabelle anhand der Ergebnisse eines Joins mit einer Quelltabelle
- Kann z.B. zwei Tabellen synchronisieren, indem Zeilen in einer Tabelle anhand von Unterschieden, die in der andere Tabelle gefunden wurden, eingefügt, aktualisiert oder gelöscht werden

MERGE KLAUSEL

```
MERGE Table_definition AS Target
USING (Table Source) AS Source
(
      Column Keys
)
ON (
      Search Terms
)
WHEN MATCHED THEN
      UPDATE SET
        or
      DELETE
WHEN NOT MATCHED BY TARGET/SOURCE THEN
      INSERT
```

MERGE Klausel

```
MERGE Books
USING
( select MAX(BookId) BookId, Title, MAX(Author)
Author, MAX(ISBN) ISBN, MAX(Pages) Pages
  from Books
  group by Title )MergeData
ON Books.BookId = MergeData.BookId
WHEN MATCHED THEN
     UPDATE SET Books.Title = MergeData.Title,
     Books.Author = MergeData.Author,
     Books.ISBN = MergeData.ISBN,
     Books.Pages = MergeData.Pages
WHEN NOT MATCHED BY SOURCE THEN DELETE;
```

MERGE Klausel

- Wie würde folgende Tabelle nach dem Merge aussehen?

BookId	Title	Author	ISBN	Pages
1	Microsoft SQL Server 2005 For Dummies	Andrew Watt	NULL	NULL
2	Microsoft SQL Server 2005 For Dummies	NULL	NULL	432
3	Microsoft SQL Server 2005 For Dummies	NULL	978-0-7645-7755-0	NULL

- Nach dem Merge:

BookId	Title	Author	ISBN	Pages
3	Microsoft SQL Server 2005 For Dummies	Andrew Watt	978-0-7645-7755-0	432

MERGE Klausel

```
MERGE Target AS T
USING Source AS S
ON (T.EmployeeID = S.EmployeeID)
WHEN NOT MATCHED BY TARGET AND S.EmployeeName LIKE 'S%'
    THEN INSERT(EmployeeID, EmployeeName)
     VALUES(S.EmployeeID, S.EmployeeName)
```

Angestellten aus der Tabelle S, deren Name mit S anfängt und die in der Tabelle T nicht existieren (nach Id), werden jetzt auch in T eingefügt mit Id und Name.

Aufgabe

- Gegeben seien folgende Tabellen:
  - Kunden (id ,name, vorname)
  - KundenNew (id ,name, vorname)
- In der Tabelle KundeNew hat man die aktuelle Daten über die Kunden einer Firma.
- Benutze die Merge Klausel um die Tabelle Kunden zu aktualisieren, sodass man auch hier die aktuelle Informationen über Kunden hat.
- Welche sind die mögliche Änderungen die vorkommen können?
  - Man nimmt an, dass die Ids der Kunden nicht geändert wurden
  - Ein Kunde kann sein Namen ändern
  - Ein Kunde kann gelöscht werden
  - Ein neuer Kunde kann eingefügt werden