Universitetet i Sørøst-Norge

Leksjon 7: Triggere

Jarle Håvik

DAT2000 – Database 2

Kunnskaps- og ferdighetsmål

- Kunnskap:
  - *Bruksområder og virkemåte for lagrede prosedyrer og triggere*
    - ❑ Forskjellen på rad- og setningstrigger
    - ❑ Forklare forskjellen på BEFORE og AFTER

- Ferdigheter:
  - Opprette triggere i MySQL
  - Programmere triggerfunksjoner og tilordne disse til triggere i PostgreSQL
  - Slette, endre, aktivere og deaktivere triggere i PostgreSQL

Litteratur

- Silberschatz, Abraham et.al: Database System Concepts
  - Kapittel 5.3 Advanced SQL - Triggers s 206-213
- Kristoffersen, Bjørn: Databasesystemer
  - Kapittel 13.3 Triggere (s. 393 – 400)
- Dubois, Paul: MySQL
  - kapittel 4.2.3 Triggers
- PostgreSQL manualen: https://www.postgresql.org/docs/12/sql-createtrigger.html
- EDB: Everything you need to know about PostgreSQL triggers

Agenda

- Kort repetisjon om funksjoner og prosedyrer
- CRUD operasjoner
- Definisjonen og formålet med triggere
- Strukturen til en trigger
  - MySQL
  - PostgreSQL
- Ulike trigger typer
- Eksempel
- Fordeler og ulemper med triggere
- I økt 2 blir det fokus på eksempler og da først og fremst postgreSQL

Tilbakeblikk til forrige leksjon om lagrede rutiner

- Vi såg på lagrede prosedyrer og funksjoner:
  - Prosedyrekoder for å kapsle inn oppgaver eller beregninger og utføre forretningslogikk
  - Blir lagra trygt på databaseserveren
  - De utføres eksplisitt og når som helst av brukere av systemet
  - Funksjoner returnerer noe
  - Prosedyrer returnerer ingenting

Triggere

Mens triggere er:

- Også prosedyrekoder for å utføre handlinger relatert til noen referansemessige integritetsbegrensninger, andre komplekse begrensninger eller revisjonsendringer i data
- De lagres trygt på databaseserveren
- De kjøres implisitt og automatisk når en hendelse inntreffer på databaseserveren

En utløser er en prosedyre som påkalles automatisk av DBMS som svar på spesifiserte endringer i databasen, det er DBA som spesifiserer disse.

Definisjon av trigger

- Triggere er en spesiell type lagrede prosedyrer som avfyres før (BEFORE), etter (AFTER)eller i steden for (INSTEAD OF) en SQL-setning
  - Når et DML utsagn blir utført på en tabell → hendelsesdrevet
  - Påkaller en trigger som avfyres for INSERT, UPDATE, or DELETE
- Skiller mellom 2 typer
  - Radtrigger
  - Setningstrigger (mangler i både mySQL og MariaDB)

Rad-trigger

Blir avfyrt for hver rad som blir berørt av en oppdatering.

```
CREATE TRIGGER tr_ansatt_1
  BEFORE UPDATE ON ansatt
  FOR EACH ROW
BEGIN
  -- Behandle raden
END
```

- Kan benytte OLD og NEW
- Blir ikke avfyrt om ingen rader er påvirket av SQL-setningen

En SQL-setninger kan oppdatere mange rader. For eksempel vil en DELETE som påvirker 10 rader føre til at ON-DELETE-triggere på måltabellen blir avfyrt ti separate ganger, en gang for hver slettet rad.

Eller om en velger å oppdatere lønnen til alle ansatte i en avdeling med 10%

```
UPDATE ansatt SET timeloenn = timeloenn * 1.1 WHERE avdnr = 2;
```

Setningstriggere

Blir avfyrt én gang for hver SQL-setning

- Setnings-triggere er ikke støttet i MySQL

- Kan ikke benytte OLD og NEW
- Blir uansett avfyrt om ingen rader er påvirket av SQL-setningen

```
CREATE TRIGGER transfer_insert
    AFTER INSERT ON transfer
    REFERENCING NEW TABLE AS inserted
    FOR EACH STATEMENT
    EXECUTE FUNCTION check_transfer_balances_to_zero();
```

- Triggere som er spesifisert for å å bli avfyrt ved INSTEAD OF hendelse må merkes FOR EACH ROW, og kan bare defineres på VIEWS.
- BEFORE og AFTER triggere på et VIEW må merkes som FOR EACH STATEMENT.
- I tillegg kan triggere defineres til å avfyres for TRUNCATE, men bare FOR EACH STATEMENT.

Formålet med triggere

- Triggere kan nyttes til å:
  - Opprettholde dataintegriteten i database
    - For eksempel når en ny post (ny tilsett) blir lagt inn i ansatt-tabellen, så bør også nye poster bli opprettet i løns-, skatt- og ferietabellene.
    - Når en ordre blir effektuert skal det lages bade pakkseddel, faktura og eventuelt restordre.
  - Flytt overvåkingslogikk fra applikasjoner til DBMS, for eksempel å sjekke oppdateringer i databasen.
    - Eksempelvis når lønna til en ansatt blir justert fra 1500 til 2000 så fyres det av en trigger og sjekker om oppdateringen faktisk har gått igjennom.
  - Logg oppdateringer som er gjort på databasetabeller.
    - Eksempelvis kan en ha en trigger som fyres av og legger gammel og ny lønn, samt tidspunkt og hvem som har oppdatert lønn til en eller flere ansatte

Generell trigger struktur

- Trigger kroppen eller trigger handlingen består av:
  - Hendelsen – event (Påkrevd): Fyrer av triggeren
  - Betingelse (valgfri): Tester hvorvidt triggeren skal utføres. WHEN
  - Handlingen (Påkrevd): Det som skal skje om triggerfunksjonen/blokken skal utføres(SQL-setning og kode)

Trigger Format

CREATE TRIGGER TriggerName

BEFORE | AFTER | INSTEAD OF

INSERT | DELETE | UPDATE [OF TriggerColumnList] → INSERT, UPDATE, or DELETE

ON TableName

[REFERENCING {OLD | NEW} AS {OldName | NewName}] → Old row as var / New row as var

[FOR EACH {ROW | STATEMENT}] → Mandatory in row-level triggers only

[WHEN Condition] → Similar to SQL's WHERE clause

<trigger action> → SQL statements

Syntaksen for triggere varierer mellom de ulike DBMS!

Forskjellige triggersyntakser

PostgreSQL

```
CREATE TRIGGER check_update
    BEFORE UPDATE ON accounts
    FOR EACH ROW
    EXECUTE PROCEDURE check_account_update();
```

MySql

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
       FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.01 sec)
```

MariaDB

```
CREATE TRIGGER increment_animal
AFTER INSERT ON animals
FOR EACH ROW
UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
```

Cassandra

```
CREATE TRIGGER myTrigger ON myTable USING 'org.apache.cassandra.triggers.InvertedIndex';
```

Oracle

```
CREATE TRIGGER bt BEFORE UPDATE OR DELETE OR INSERT ON sal
BEGIN
 stat.rowcnt := 0;
END;
```

MS SQL Server

```
CREATE TRIGGER reminder1
ON Sales.Customer
AFTER INSERT, UPDATE
AS RAISERROR ('Notify Customer Relations', 16, 10);
GO
```

Eksempel 1

- Anta at du har en trigger som sørger for at alle high school seniors er satt til «graduate». Dette betyr at etter en hendelse med oppdatering av en seniors grade fra 12 til 13, så bør triggeren sette karakteren til NULL, noe som indikerer at eleven er uteksaminert.

- Statement-level trigger:

```
create trigger stmt_level_trigger
after update on Highschooler
begin
   update Highschooler
   set grade = NULL
   where grade = 13;
end;
```

Sjekker hele tabellen for oppdateringer samtidig. Triggeren vil fortsatt avfyres selv om ingen rader er oppdatert.

- Row-level trigger:

```
create trigger row_level_trigger
after update on Highschooler
for each row
when New.grade = 13
begin
   update Highschooler
   set grade = NULL
   where New.grade = Highschooler.grade;
end;
```

Kontroll av oppdateringer vil bare bli utført rad for rad for de radene som er oppdatert. Triggeren aktiveres ikke hvis ingen rader er oppdatert.

Et annet eksempel

- Anta at vi har en trigger som brukes til å opprettholde statistikk over hvor mange studenter som er yngre enn 18 og som oppdateres på en gang med et typisk UPDATE utsagn.

- Statement-level trigger:

```
CREATE TRIGGER init_count_trigger
BEFORE UPDATE ON Students
DECLARE
        count:= INTEGER;
BEGIN
   count:=0;
END
```

Sjekker hele tabellen for oppdateringer.
Triggeren blir avfyrt selv om ingen rader blir oppdatert.

- Row-level trigger:

```
CREATE TRIGGER incr_count_trigger
AFTER UPDATE ON Students
WHEN (new.age < 18)
FOR EACH ROW
BEGIN
   count:=count+1 ;
END
```

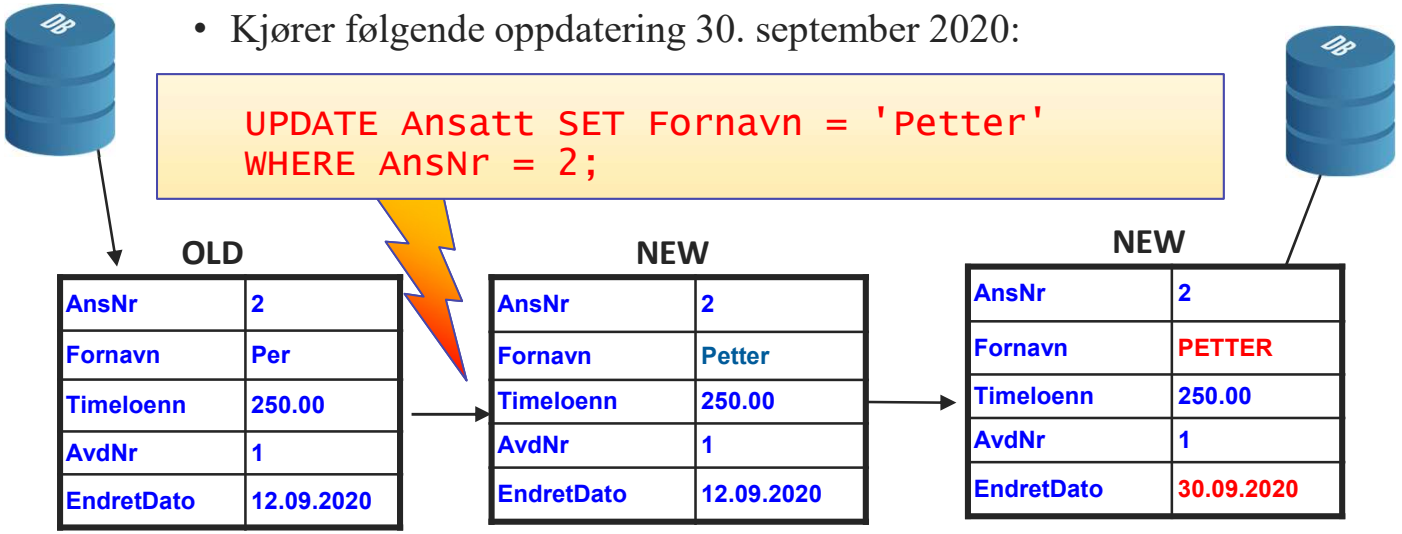
Sjeking på oppdatering vil bli utført rad for rad for de radene som faktisk blir oppdatert. Triggeren vil ikke utløses dersom ingen rader er oppdatert.

Automatisk formatering – OLD og NEW

```
CREATE TRIGGER tr_ansatt
  BEFORE UPDATE ON Ansatt
  FOR EACH ROW
BEGIN
  SET NEW.Fornavn = UPPER(NEW.Fornavn);
  SET NEW.EndretDato = CURDATE();
END
```

- Kjører følgende oppdatering 30. september 2020:

```
UPDATE Ansatt SET Fornavn = 'Petter'
WHERE AnsNr = 2;
```



OLD og NEW

- To innebygde «postvariabler» OLD og NEW
  - OLD → opprinnelige verdier før endring
  - NEW → nye verdier slik de vil bli etter oppdatering i databasen
  - Ved INSERT er OLD = NULL, ved DELETE er NEW = NULL

- Hvordan referere til data i raden som oppdateres
  - OLD.<kolonnenavn> gir gammel verdi
  - NEW.<kolonnenavn> gir ny verdi

- BEFORE triggere → kjøres før databasen oppdateres
  - NEW-verdier er endret og tilgjengelige selv om databasen ikke er oppdatert ennå!
  - Triggeren kan endre NEW-verdier før oppdatering i basen
  - Automatisk konvertering/formatering av data som settes inn
  - Automatisk oppdaterte dataverdier, f.eks. datostempel som viser når raden sist ble endret

- AFTER triggere → kjøres etter at databasen er oppdatert
  - Da er det for seint å endre NEW-verdiene.
  - OLD og NEW-verdier kan fremdeles refereres, f.eks. for å oppdatere andre tabeller

OLD og NEW

- Hva inneholder bufferne?

	OLD	NEW
INSERT	NULL i alle kolonner	Verdier i den nye raden som skal settes inn
UPDATE	Verdier slik de er/var i databasen før oppdatering	Verdier slik de vil bli i databasen etter oppdatering
DELETE	Verdier slik de er/var i databasen før sletting	NULL i alle kolonner

- OLD og NEW kan bare brukes i rad-triggere!

BEFORE - AFTER

- Rad-triggere

	BEFORE	AFTER
INSERT	Ny rad er ikke satt inn ennå	Ny rad er satt inn i tabellen
UPDATE	Raden er ikke oppdatert i tabellen	NEW-verdier er oppdatert i tabellen
DELETE	Raden er ikke slettet ennå	Raden er slettet

- (Setnings-triggere – ikke støttet i MySQL, men i PostgreSQL)

	BEFORE	AFTER
INSERT	Ingen rader er satt inn ennå	Alle nye rader er satt inn i tabellen
UPDATE	Ingen rader er oppdatert ennå	Alle berørte rader er oppdatert i tabellen
DELETE	Ingen rader er slettet ennå	Alle rader som berøres av DELETE er slettet fra tabellen

Loggføring og AFTER-triggere

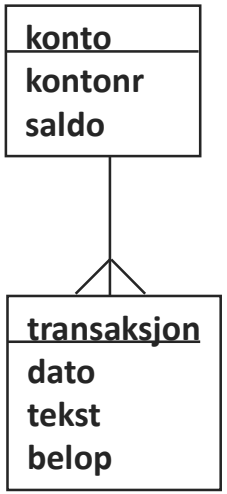
- Hver gang noen endrer lønnen til en ansatt
  - Sett inn en rad i en egen loggtabell AnsattLogg
  - (Forutsetter at tabellen er opprettet)

```
CREATE TRIGGER ansatt_a_upd_trg
  AFTER UPDATE ON ansatt
  FOR EACH ROW
BEGIN
  INSERT INTO
    AnsattLogg(AnsNr, Dato, GammelLønn, NyLønn)
  VALUES
    (NEW.AnsNr, CURDATE(), OLD.Lønn, NEW.Lønn);
END
```

Automatiske følgeoppdateringer

- Oppdatere saldo på konto etter en ny transaksjon

```
CREATE OR REPLACE TRIGGER tr_trans_ins
  AFTER INSERT ON transaksjon
  FOR EACH ROW
BEGIN
  UPDATE konto
    SET konto.saldo =
      konto.saldo + NEW.belop
    WHERE konto.kontonr = NEW.kontonr;
END;
```



- Bør gjøres i AFTER-triggere når NEW-verdier er endelige
- Ikke bruk COMMIT / ROLLBACK i triggere!
- Oppdateringene bekreftes av klientprogrammet
- Ved feil skal alle endringer rulles tilbake. Det vil ikke skje hvis triggeren har gjort COMMIT!

Forretningsregler

CHECK-regler kan legges på i CREATE TABLE eller ALTER TABLE:

```
ALTER TABLE Vare
ADD CHECK (Pris < 10000);
```

Men kan også bruke trigger i stedet:

```
CREATE TRIGGER vare_b_upd_trg
  BEFORE UPDATE ON Vare
  FOR EACH ROW
BEGIN
  IF NOT (NEW.Pris < 10000) THEN
    SIGNAL SQLSTATE '80000'
    SET MESSAGE_TEXT = 'For dyr vare!';
  END IF;
END
```

Beskrivelse av triggere i systemkatalogen

Metadata-kommando:

```
SHOW TRIGGERS;
```

- MySQL: Eller spør mot metatabellen information_schema.triggers:

```
SELECT TRIGGER_SCHEMA, TRIGGER_NAME,
       EVENT_MANIPULATION, EVENT_OBJECT_TABLE,
       ACTION_STATEMENT, ACTION_TIMING
 FROM INFORMATION_SCHEMA.triggers;
```

- Lag en SQL-spørring som lager et SQL-skript som sletter alle triggere:

```
SELECT CONCAT('DROP TRIGGER IF EXISTS ',
  TRIGGER_SCHEMA, '.', TRIGGER_NAME, ';') AS "SQL"
FROM INFORMATION_SCHEMA.TRIGGERS
```

Bruksområder for triggere

- BEFORE triggere – fyres før oppdatering i basen
  - Automatisk formatering av verdier
  - Automatisk datostempling av rader
  - Kontroll med forretningsregler
  - Finmasket tilgangskontroll

- AFTER triggere - fyres etter oppdatering i basen
  - Automatiske oppdateringer i andre tabeller
  - Loggføring av oppdateringer
  - Oppdatere avledete data i andre tabeller

- Brukes for operasjoner som skal gjøres «uten unntak»!

Fordeler med Triggere

- Reuserer redundant kode
- Økt sikkerhet
- Forbedra integritet
- Passar godt inn i en client-server arkitektur

Sjå også Silberschatz 5.3.1 Need for Triggers (s. 206)

Ulemper med triggere

- Ytelse overhead
- Kaskade effekter (rekursive triggere)
- Kan ikke planlegges
- Mindre portable

Sjå også Silberschatz 5.3.3 When Not to Use Triggers (s. 210)

Triggere oppsummert

- Trigger
  - Definerer en handling som databasen skal utføre når en hendelse inntreffer i applikasjonen
  - Basert på Event-Condition-Action (ECA) model
- Typer
  - Row-level
  - Statement-level
- Event: INSERT, UPDATE or DELETE – i PostgreSQL også TRUNCATE
- Timing: BEFORE, AFTER or INSTEAD OF

Praktisk øving 1

Etter første økt – sjå på hvordan en kan opprette følgende trigger på *ansatt*-tabell*

- Når en legger inn en ny tilsett i ansatt tabellen så skal både fornavn og etternavn konverteres til å ha stor forbokstav.

*Du kan finne en tutorial for dette her: http://www.postgresqltutorial.com/creating-first-trigger-postgresql/

Praktisk øving 2

Vi har gitt følgende relasjoner

```
Highschooler(ID int, name text, grade int);
Friend(ID1 int, ID2 int);
Likes(ID1 int, ID2 int);
```

Skriv en eller flere triggere for å opprettholde symmetri i venneforhold. Spesielt hvis (A, B) slettes fra Friend, bør (B, A) også slettes. Hvis (A, B) settes inn i Friends, så bør (B, A) også settes inn.