GitHub versjonskontroll + SQL subqueries

Modul 9 (uke 7) innhold

Dagens agenda

08:30 | **Check-in** + **GitHub versjonskontroll** & **SQL subqueries**

| Sette opp GitHub, så løse SQL subquery oppgaver via Git

| (Oppgavene ligger på Canvas)

11:30 | *45 min. lunch*

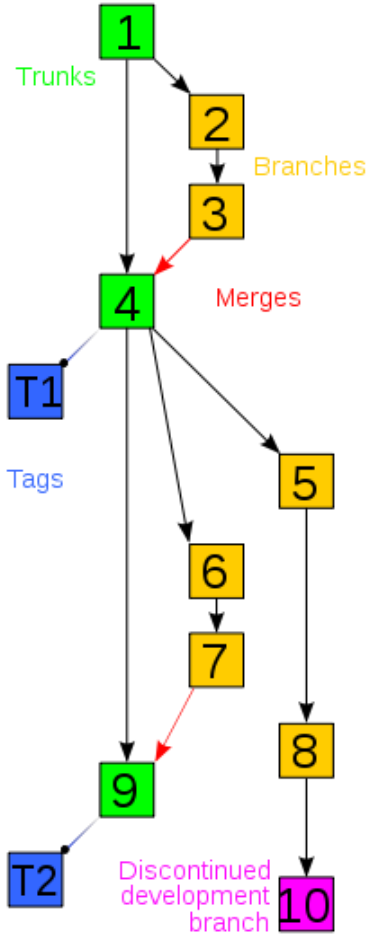
12:15 | **GitHub: Demo av merge conflicts++**

| Repetisjon: Forberedelse mot checkpoint, gjennom selvvalgte oppgaver

15:30 | **Små, utestående modultemaer** + **Check-out** (til 16:05'ish)

Versjonskontroll

- Så ... hva er versjonskontroll?
- Ref. Wikipedia: *"[…] version control […] is a class of systems responsible for managing changes to computer programs, documents, […] or other collections of information."*
- Figur: Noen muligheter med versjonskontroll.



Versjonskontroll – forts.

- Før i tiden var versjonskontrollsystemer alltid klient-tjener ("client-server") oppsett. Noen client-server systemer er:
  - Concurrent Versions System, CVS (1990)
  - Visual SourceSafe (1994)
  - Apache Subversion, SVN (2000)
  - MKS Integrity (2001)

- Etter det har en annen struktur blitt veldig populær: Distribuerte systemer ("distributed systems"). Noen distributed systems er:
  - Mercurial (2005)
  - Git (2005) <-- Det vanligste systemet i industrien(?).

Fra oversikten: List of Revision Control Software - Wikipedia

Få trening med GitHub

- For å komme i gang med GitHub, se denne videoen og følg beskrivelsen:
  - https://www.youtube.com/watch?v=AW_U0q5BtFI

- For å integrere GitHub i VS Code, se denne artikkelen: https://code.visualstudio.com/docs/sourcecontrol/github
  - Merk: Med Github integrert i VS Code slipper dere (stort sett) å benytte Git console! ☺

- Noen andre populære Git verktøy:
  - git for windows, SourceTree

Ikke alle filer i et prosjekt skal deles

- Når vi lager et kodeprosjekt kan det inneholde filer vi ikke ønsker delt på Git.
  - F.eks. filer med programmererens personlige oppsett for prosjektet eller liknende.

- Dette tar Git systemet høyde for:
  - Vi kan innkludere en "ignore file". Denne spesifiserer filtyper som *ikke* skal synkroniseres via Git.

- Mange IDE-er lar det legge et nyopprettet prosjekt på Git dirkete og lager en git ignore fil for deg.
  - Eller så kan man alltids google det: Ligger maler for slike filer på nett.

Git kommandoene vi bruker oftest

Noen vanlige Git kommandoer:

- git clone: For å opprette en lokal kopi/kopling til et (nytt) repo på GitHub.
- git status: For å se (git-)tilstand på lokale filer.
- git add: For å kople en (ny) lokal fil i git-mappa opp mot git systemet.
- git **commit:** Ofte! Gjør en commit etter hver utførte oppgave som *virker* (altså ikke om du har bugs).
  - Husk å legge inn en beskrivende kommentar! (Hvis console: `git commit -m "`*Beskrivende tekst her da*`"`.)
- git **pull**, etterfulgt av git **push** (i noen verktøy er disse kombinert som en git **sync** feature)
  - Gjør pull så push (evt. sync) før du begynner på noe nytt (så du er sikker på å ha siste versjon av kodebasen).
  - Når du er ferdig med en del endringer / ferdig for dagen (om koden kompilerer og ikke har "uferdige" deler).

Inspirert av: docs.github.com/en/get-started/using-git/about-git

Hvor viktig er det å kunne Git da?

- Git-kunnskap er VELDIG viktig!

- Absolutt ALLE seriøse softwareutviklingsfirmaer benytter versjonskontrollsoftware.

- ALLE oppdragsgivere ønsker at du på jobbintervju kan svare:
  - *"Jeg brukte Git på mye av koden jeg utviklet på JavaScript-programmet."*

- Fin øvelse fremover:
  - Bruk GitHub på de nye prosjektene dere lager.
  - Gjør commits ofte! Sync (pull og push) innimellom. ☺

In case of fire…

SQL subqueries

Subqueries

- Å putte en SELECT inne i en annen SQL statement kalles en subquery.

- Resultatet fra en SELECT er på samme format som en ny tabell:
  - Det danner kolonner og rader på samme måte som databasens eksisterende tabeller.
  - Derfor er det ikke noe problem å bruke resultatet av en SELECT som et element i en SQL statement!

Subqueries – fortsetter

- Hvorfor benytte subqueries?
  - Noen ganger trenger vi svaret fra en spørring før vi kan begynne på en annen SQL statement.

- Eksempel: Ønsker å finne hvor mange byer som har innbyggertall over eller likt gjennomsnittet.
  - Før vi kan fullføre denne spørringen, må vi vite hva gjennomsnittet er!

- NB: Vi har faktisk allerede brukt subqueries:
  - Når vi gjorde insert i en tabell (email) basert på uttrekket fra en annen (person), benyttet vi subquery. :-)

SELECT med subquery, eksempel #1

- Oppgave: *"Hvor mange byer har et innbyggertall over eller likt gjennomsnittet?"*
  - Kan løse det med en subquery, slik:

```
SELECT COUNT(*)
FROM city
WHERE Population >=
 (SELECT AVG(Population)
  FROM city);
```

SELECT med subquery, eksempel #2

- En oppgave til: *"Hent navnet på alle europeiske byer."*
  - Kan løses med subquery på denne måten:

```
SELECT Name
FROM city
WHERE CountryCode IN
 (SELECT Code
  FROM country
  WHERE Continent = 'Europe');
```

  - Forklaring av `IN`: Lar oss sammenlikne innholdet i en kolonne med et utvalg verdier.
  - NB: Denne oppgaven kan også løses med JOIN.

SELECT med subquery, eksempel #3

- Tredje (og siste) eksempelside: *"Hvor mange prosent utgjør hvert land av det største landets størrelse?"*
  - Kan løses med en subquery i SELECT delen, på denne måten:

```
SELECT Name, SurfaceArea / (SELECT MAX(SurfaceArea) FROM country) * 100
FROM country
ORDER BY SurfaceArea DESC;
```

- Bonusspm.: "*Ønsker å se prosentandelen innbyggerne i et land utgjør av jordas totale befolkning.*"
  - Denne og andre er nærmere spesifisert i øvingsoppgavene. ☺

Oppgavetid!

- Øvingsoppgavene ligger på Canvas.

- HUSK! Det er 2 temaer for denne øvingen:
  1. GitHub versjonskontroll.
  2. SQL subqueries.

- Gi dere selv tid til å sette dere inn i GitHub først! ☺


Academy