TIETOKANNAT KUVIENJAKOSOVELLUKSESSA

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja vienstintätekniikka, insinööri (AMK)

Syksy 2021

Hanna Sinkkilä

Mira Hämäläinen

Nina Tulilahti

Sisällys

Liitteet

1 Projektin kuvaus ja ensimmäinen suunnitelma

Tässä projektissa on suunniteltu tietokantarakennne verkossa toimivalle kuvagallerialle. Web-sovellus on toteutettu Django-ohjelmistokehystä hyödyntäen. Web-sovelluksen ominaisuuksia ovat käyttäjätunnuksen luominen, albumeiden luominen ja kuvien lisääminen palveluun. Sovellus käyttää SQLite relaatiotietokantaa joka on luotu Django ORMin avulla.

Projektia aloittaessa ei kokonaiskuva tietokantarakenteesta ollut vielä selkeä, ja siksi websovelluksen luominen päätettiin aloittaa mahdollisimman yksinkertaisesta versiosta. Se pitäisi sisällään toiminnallisuuden kannalta vain välttämättömimmät elementit, mutta projektin edetessä voitaisiin tietokantarakennetta laajentaa. Ensimmäinen tietokantasuunnitelma piti sisällään kolme taulua: album, photo ja user.

Yksinkertaisesta taulurakenteesta huolimatta suunnitelmassa on pyritty huomioimaan taulujen normalisointi niin, että tietoja voitaisiin muokata ja poistaa ilman että taulujen väliset riippuvuussuhteet hankaloittavat sovelluksen käyttöä. Myös uusien tietojen lisäämisen täytyisi olla mahdollista ilman suurien muutosten tekemistä taulurakenteeseen. Taulujen tulisi olla myös mahdollisimman kuvaavia ja yksityiskohtaisia. Esimerkiksi taulu album ei voi pitää sisällään tietoa kaikista käyttäjistä, jotka ovat ladanneet kyseiseen albumiin kuvia, sillä sarakkeella on oltava yksi spesifi arvo. Sen sijaan photo-taulu pitää sisällään tiedon siitä, mihin albumiin kuva on ladattu (viittaus albumiin) ja kuka käyttäjistä on ladannut kyseisen kuvan (viittaus käyttäjään).

2 Tietokannan tekninen toteutus

Django ORM (An object-relational mapper) oli loogisen tuntuinen valinta Django-frameworkilla toteutettavaan sovellukseen. Tämä tarkoitti käytännössä sitä, että kaikki ohjelman taulut kirjoitettiin Pythonilla, ja Django ORM muuntaa koodin SQL-kielelle, jota tietokanta ymmärtää. Models.py tiedostossa olevat luokat kuvastavat tietokannan tauluja ja attribuutit luokassa edustavat tietokannan kolumneja.

```
class Photo(models.Model):

    title = models.CharField(max_length=20)
    description = models.CharField(max_length=50)
    created = models.DateTimeField(auto_now_add=True)
    image = models.ImageField(upload_to='photos/')
    submitter = models.ForeignKey(get_user_model(), on_delete=models.CASCADE)
    is_public = models.BooleanField(default=True)


    def __str__(self):
        return self.title
```

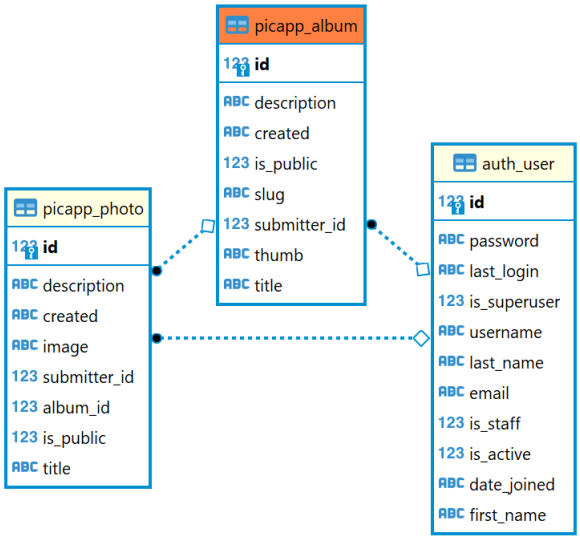
Yllä on esimerkki Photo-taulun luomisesta: jokainen kenttä edustaa taulun sarakkeita. Otsikko on tyyppiä `CharField` ja se on rajattu 20 merkkiin. Created-attribuutilla on `DateTimeField` tyyppi ja se saa arvokseen oletuksena sen hetken, jolloin kuva ladataan palveluun. `ImageField` määrittää minne varsinainen kuva ladataan ja varastoi osoitteen kyseiseen sijaintiin. Submitter viittaa viiteavaimena user-tauluun, ja lisäksi on määritelty, että poistettaessa tietoja poistetaan myös kaikki viittaukset lapsitauluihin. Viimeinen kenttä is_public on tyyppiä `BooleanField`, eli totuusarvo ja se määrittelee kuvan olevan oletusarvoisesti julkinen.

Funktio `__str__` määrittelee, kuinka kentät näkyvät admin-paneelissa. Jokaiselle taululle on määritelty myös pääavain, jonka tunniste on id. Valinnan koettiin olevan riittävän selkeä, kun tietokantojen luomista harjoitellaan ensimmäistä kertaa. Muutosten tekemiseksi täytyy vielä ajaa komenoto `python3 manage.py migrate` komentoriviä käyttäen.

3 Tietokantarakenne

Tässä osiossa käydään läpi projektin lopullista tietokantarakennetta. Projektissa päädyttiin käyttämään Djangon sisäänrakennettua autentikaatiosysteemiä, jolloin erillistä käyttäjätaulua ei tarvinut luoda. Alkuperäiseen suunnitelmaan lisättiin oma taulu hakusanoille, joka luotiin Djangon taggit-lisäosan avulla. Pääpaino on itse luoduissa tauluissa. Djangon luomia tauluja (kuten autentikaatioon liittyvät taulut) käydään läpi niiltä osin, kun ne koskettavat itse luotuja tauluja.

3.1 picapp_album



Kuva 1 picapp_album kaavio

Taulu picapp_album on albumeita varten luotu taulu ja se mukailee hyvin pitkälti suunnitelmaa, joka projektin alussa tehtiin. Taulun tarkoitus on säilöä yksityiskohtaisesti albumeihin viittaava tieto. Taulu auth_user on Djangon luoma taulu, joka kuvastaa käyttäjää. Picapp_album viittaa tähän viiteavaimella submitter_id, joka kuvaa tietoa siitä kuka on albumin luoja.

Ohjelmassa voi olla kahdenlaisia albumeita: julkisia ja yksityisiä. Tämä luo myös kahdenlaisia suhteita albumin ja käyttäjän välille. Julkisessa albumissa voi olla kuvia useilta eri käyttäjiltä, ja yksi käyttäjä voi luoda useita albumeita. Tällöin kyseessä on many-to-many suhde. Yksityisessä albumissa puolestaan voi olla ainoastaan albumin luojan lisäämiä kuvia. Käyttäjä voi edelleen luoda useita albumeita. Kyseessä on one-to-many-tyyppinen suhde.

Kolumnit:

- `id`: pääavain (PK)
- `description`: albumin kuvaus
- `created`: ajankohta, jolloin albumi on luotu

- is_public: määrittää, onko albumi julkinen vai yksityinen. Arvo 1 tarkoittaa julkista (true) ja arvo 0 yksityistä (false)
- submitter_id: viiteavain (FK) viittaa tauluun auth_user
- thumb: thumbnail-kuvakkeen sijainti
- title: albumin otsikko

3.2 picapp_photo

Taulun picapp_photo tarkoitus on säilöä yksittäisen kuvan tiedot. Taulussa on kaksi viiteavainta. Submitter_id viittaa tauluun auth_user, ja kuvastaa käyttäjää, joka kuvan on ladannut palveluun. Kuvalla voi olla vain yksi omistaja (user) mutta yksi käyttäjä (user) voi ladata palveluun useita kuvia. Näiden välillä on one-to-many -tyyppinen suhde. Viitevain album_id osoittaa picapp_album tauluun, ja kertoo, mihin kansioon kuva kuuluu. Yksi kuva voi kuulua vain yhteen albumiin, mutta albumissa voi olla useita kuvia. Myös tässä on one-to-many -tyyppinen suhde.

Kolumnit:

- id: pääavain (PK)
- description: kuvateksti, jonka käyttäjä voi määrittää
- created: päivämäärä, jolloin kuva on lisätty palveluun
- image: osoite, josta palvelu hakee kuvan
- submitter_id: viiteavain (FK) viittaa tauluun auth_user
- album_id: viiteavain (FK) viittaa tauluun picapp_album
- is_public: määrittää, onko kuva julkinen vai yksityinen
- title: kuvan otsikko, käyttäjä voi määrittää

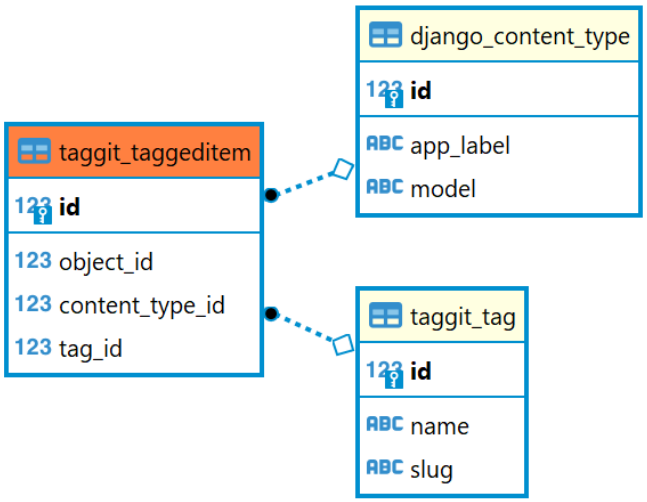
3.3 taggit_tag

Taggit_tag taulu on yksinkertainen, kolmen sarakkeen taulu jonka tarkoitus on säilöä kaikki ohjelmaan syötetyt hakusanat eli tagit. Taulu ei viittaa toisiin tauluihin, mutta on silti osa kokonaisuutta.

Kolumnit:

- `id`: pääavain (PK)
- `name`: hakusanana nimi
- `slug`: osoiterivissä näkyvä tieto kun hakusanalla haetaan kuvia

3.4 taggit_taggeditem



Kuva 2 taggit_taggetitem taulun kaavio

Taggit_taggeditem on liitostaulu, jonka tarkoitus on yhdistää tagit sisältöön, tarkemmin sisältötyyppiin. Projektissa tageja on käytetty ainoastaan kuville, mutta tageilla on mahdollista jäsentää myös muuta sisältöä, esimerkiksi albumeita tai käyttäjäryhmiä. Viiteavaimella `content_type_id` viitataankin tauluun `django_content_type`, joka sisältää listan mahdollisista sisältötyypeistä. Viiteavaimella `tag_id` viitataan edelliseen `taggit_tag` tauluun ja sen sisältämiin hakusanoihin.

Tämänkaltaista liitostaulua tarvitaan tietokantarakenteessa, jotta normalisoinnin ehdot toteutuisivat. Tämä tarkoittaa sitä, että yksittäisen taulun tulee olla samaan aikaan mahdollisimman kuvaava ja tarkka: tiedot on jaettu niinsanotusti atomitasolle. Liitostauluilla taulujen tietoja voidaan yhdistää SQL-hakuja tehdessä.

Kolumnit:

- `id`: pääavain (PK)
- `object_id`: viittaa objektiin, joka on luodaan ohjelman python-koodissa. Objekti on yhteydessä Django ORMiin, ja mahdollistaa SQL-kyselyt tietokantojen ja moduulien välillä.
- `content_type_id`: viiteavain (FK) viittaa tauluun django_content_type
- `tag_id`: viiteavain (FK) viittaa tauluun taggit_tag

Taulujen `taggit_tag` ja `taggit_taggeditem` kanssa kommunikoi myös djangon luoma taulu `django_content_type`, jonka tehtävä on säilöä erilaiset sisältötyypit. Sisältöä ovat esimerkiksi kuvat, albumit, käyttäjät ja ryhmät.

4 Jatkokehitys ja puutteet

Projektin parissa työskentely on tukenut hyvin websovellusten tietokantarakenteen ymmärtämistä. Seuraavaa tietokantapohjaista websovellusta suunnitellessa on jo huomattavasti helpompi hahmottaa kuinka erilaiset taulut ja liitostaulut kommunikoivat, ja Django-pohjaiseten sovellusten suhteen on parempi käsitys siitä, kuinka Django-viitekehys avustaa tietokantarakenteen luomisessa omilla automaattisilla tauluillaan.

Tässä projektissa tietokannan indeksointi jätettiin suunnitteluvaiheessa vähemmälle huomiolle alkuperäisen suunnitelman sisältäessä vain kolme taulua. Kuitenkin projektin edetessä huomattiin, kuinka pienetkin lisäykset ohjelmaan monipuolistavat heti taustalla olevaa tietokantarakennetta ja indeksointi olisi siksi hyvä ottaa huomioon suunnittelussa. Mikäli vastaavanlaista websovellusta alettaisiin rakentamaan tämän projektin aikana opittujen tietojen perusteella, lisättäisiin indeksointi ainakin kuvarakenteeseen jolloin yksittäisen kuvan tietojen hakeminen olisi nopeampaa.

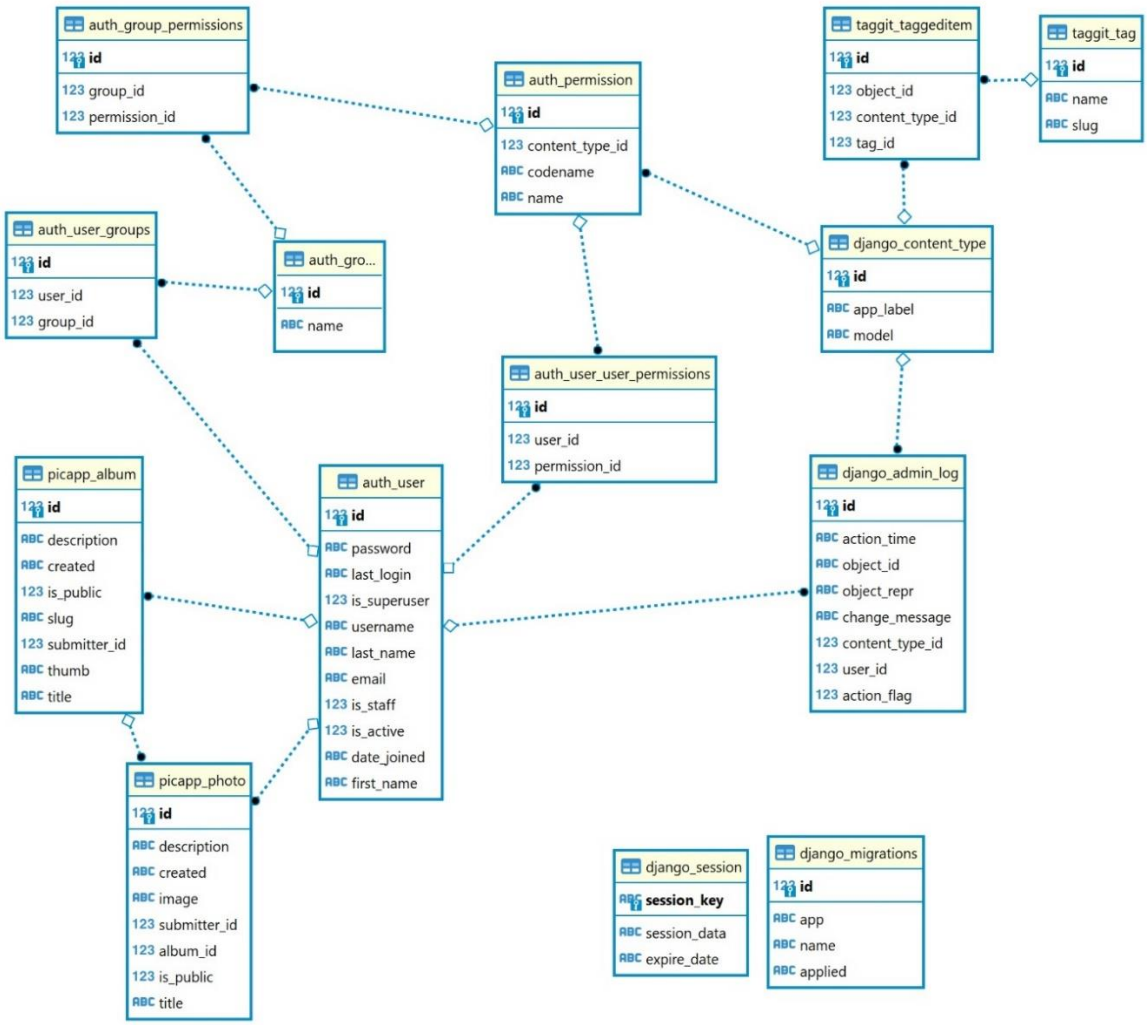
Projektin parissa työskentely on ollut mielenkiintoista, ja ryhmän sisällä syntyi useita ajatuksia siitä kuinka sovellusta voisi vielä kehittää. Kehitysvaiheessa testattiin mm. käyttäjäprofiilin luomista, joka osoittautui kuitenkin niin laajaksi kokonaisuudeksi että se

jouduttiin jättämään pois aikataulusyistä. Profiilia varten luotiin profile-niminen taulu, jonka tarkoitus olisi ollut pitää yksittäiseen käyttäjään liittyvät tiedot sisällään. Sarakkeita ideoitiin mm. käyttäjänimelle, sähköpostille, profiilikuvalle ja kirjautumistiedoille. Lisäksi taulun ajateltiin pitävän sisällään viittaukset kuviin ja albumeihin, joita käyttäjä on luonut.

Tällä hetkellä sovelluksessa on mahdollista lisätä, poistaa ja muokata kuvia, mutta esimerkiksi kuvien kommentointi ei ole mahdollista. Kommentointimahdollisuuden lisääminen sovellukseen tarkoittaisi tietokantarakenteen kannalta uuden comments-taulun luomista, joka pitäisi sisällään viittauksen picapp_photo tauluun tarkentaen, mihin kuvaan kommentti on jätetty. Lisäksi tarvittaisiin viittaus auth_user tauluun kuvastamaan sitä, kuka kommentin on jättänyt. Lisäksi taulussa olisi sarakkeet ainakin sisällölle (kommentti itsessään) sekä tieto siitä, koska kommentti on jätetty.

Yhden pienemmän taulun voisi tehdä vielä kuvan sijaintitiedolle. Sijaintitiedot voitaisiin lukea kuvan metadatasta, tai käyttäjä voisi halutessaan määrittää kuvan sijannin kuvan lisäyksen yhdeydessä.

Liite 1: ER-kaavio tietokannasta



Liite 1 ER-Kaaviossa on käytetty IDEF1X-notaatiota