KaffeDB

Innlevering 1

Gruppe 170

Kent Roger Vistven
Fredrik Randsborg Bølstad

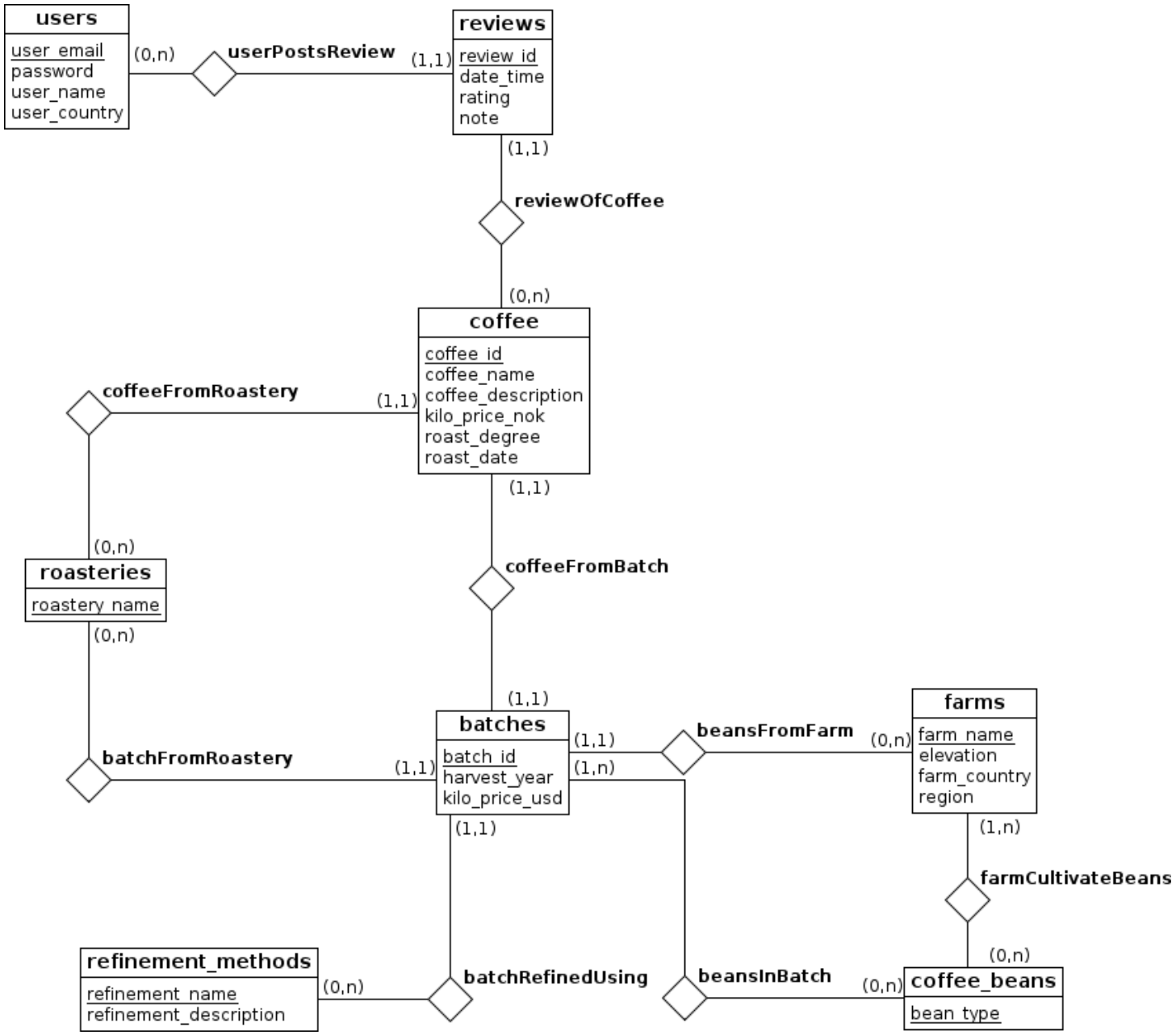
TDT4145

Datamodellering og databasesystemer

Innholdsfortegnelse

1 ER-modell

1.1 Diagram



1.2 Antakelser

- En epost-adresse er knyttet til en-og-bare-en bruker.
- Et parti med kaffebønner brukes til en-og-bare-en type kaffe.
- Brennerier gir unike navn til egne kaffer (ulike brennerier kan benytte like navn).
- Alle foredlingsmetoder har unike navn.
- Alle gårder har unike navn.
- **coffee.**`kilo_price_nok` er kiloprisen for kaffen i NOK.
- **batches.**`kilo_price_usd` er i betaling til gård per kilo med bønner i USD.
- **farms.**`elevation` er høyde over havet i meter.

1.3 Kommentarer

- Vi bruker `coffee_id` som primærnøkkel i **`coffee`**-klassen og krever samtidig at alle kombinasjoner av (`coffe_name`, `roastery_name`) må være unike.
  - Det betyr at (`coffe_name`, `roastery_name`) kan brukes som sammensatt primærnøkkel.
  - Isåfall vil `coffee_name` være en delvis nøkkel med identifiserende relasjonsklasse til **`roasteries`**.
  - I stedet genererer vi surrogatnøkkelen `coffee_id`, fordi det er mer naturlig for administrator av databasen å skille kaffetyper på ett attributt.
  - Siden `coffee_id` unikt identifiserer entiteter blir **`coffee`** en regulær relasjonsklasse.
- Tilsvarende gjelder også nøkkelen `review_id` i **`reviews`**-klassen.
  - (`user_email`, `coffee_id`, `date_time`) kunne alternativt vært brukt som primærnøkkel.
- Det kan tenkes at et brenneri ønsker å benytte samme kaffenavn for ulike produksjoner.
  - F.eks. om de vil opprettholde merkevaren ”Morgenkaffe” over tid.
  - En slik mini-verden har vanskelig for å tilfredsstille brukerhistorie 1, hvor kaffenavn og brenneri alene skal gi en unik kaffetype.
  - Dette kunne bl.a. vært løst ved å anta at brukerinput gjelder nyeste produksjon, men vi har i stedet valgt vår restriksjon fordi det gir en mer oversiktlig modell.

2 Relasjonsskjema

2.1 Tabeller

users(user_email, password, user_name, user_country)

- Primærnøkkel er `user_email` (brukerens epost-adresse, naturlig nøkkel).
- Tabellen er på 4NF.
  - Alle attributter er atomiske (1NF)
  - `user_email` er eneste nøkkelattributt (2NF)
  - Det er ingen transitive avhengigheter (3NF)
  - Alle avhengigheter har `user_email` på venstre side (BCNF)
  - `user_email` gir aldri mer enn én verdi uansett hvilke kolonner som velges (4NF)

reviews(review_id, user_email, coffee_id, date_time, rating, note)

- Primærnøkkel er `review_id` (generert nøkkel).
- `user_email` og `coffee_id` er fremmednøkler til hhv. **users** og **coffee**.
- Tabellen er på 4NF.
  - Alle attributter er atomiske (1NF)
  - Ikke-nøkkelattributtene er kun funksjonelt avhengig av hele kandidatnøkler (2NF)
    - Ikke-nøkkelattributtene er `rating` og `note`
    - Kandidatnøklene er `review_id` og (`user_email`, `coffee_id`, `date_time`)
  - Det er ingen funksjonelle avhengigheter mellom ikke-nøkkelattributter (3NF)
  - Alle avhengigheter har en supernøkkel på venstre side av avhengigheten (BCNF)
  - En vilkårlig supernøkkel gir maks én mulig verdi for samtlige kolonner (4NF)

coffee(coffee_id, coffee_name, coffee_description, kilo_price_nok, roast_degree, roast_date, roastery_name)

- Primærnøkkel er coffee_id (generert nøkkel).
- roastery_name er fremmednøkkel til **roasteries**.
- Tabellen er på 4NF.
  - Attributtene er atomiske (1NF)
  - Alle avhengigher har hele kandidatnøkler på venstre side (2NF, 3NF, BCNF)
  - ∗ Kandidatnøklene er coffee_id og (coffee_name, roastery_name)
  - Ingen supernøkler gir mer enn én verdi for en vilkårlig kolonne (4NF)

farms(farm_name, elevation, farm_country, region)

- Primærnøkkel er farm_name (gårdsnavn, naturlig nøkkel).
- Tabellen er på 4NF.
  - Attributtene er atomiske (1NF)
  - farm_name er på venstre side av alle avhengigheter (2NF, 3NF, BCNF)
    - ∗ Vi antar at region ikke entydig gir land (land kan ha regioner som deler navn)
  - farm_name gir en-og-bare-en verdi for alle kolonner (4NF)

batches(batch_id, coffee_id, roastery_name, farm_name, refinement_name, harvest_year, kilo_price_usd)

- Primærnøkkel er batch_id (generert nøkkel).
- Fremmednøkler:
  - coffee_id → **coffee**
  - roastery_name → **roasteries**
  - farm_name → **farms**
  - refinement_name → **refinement_methods**

- Tabellen er på 4NF.
  - Attributtene er atomiske (1NF) og ingen nøkler er sammensatte (2NF)
  - Alle funksjonelle avhengigheter har en supernøkkel på venstre side (3NF, BCNF)
  - Supernøkler gir maks én verdi for alle kolonner (4NF)

refinement_methods(refinement_name, refinement_description)

- Primærnøkkel er `refinement_name` (navn på foredlingsmetode, naturlig nøkkel).
- Tabellen er på 4NF.
  - Attributtene er atomiske (1NF)
  - Eneste funksjonelle avhengighet er `refinement_method` → `refinement_description` (2NF, 3NF, BCNF).
  - Tabellen er 2D og BCNF (4NF)

Følgende tabeller er 4NF fordi attributtene er atomiske (1NF) og består av to kolonner som begge er nøkler (2NF, 3NF, BCNF, 4NF).

beans_in_batch(batch_id, bean_type)
farm_cultivate_beans(farm_name, bean_type)

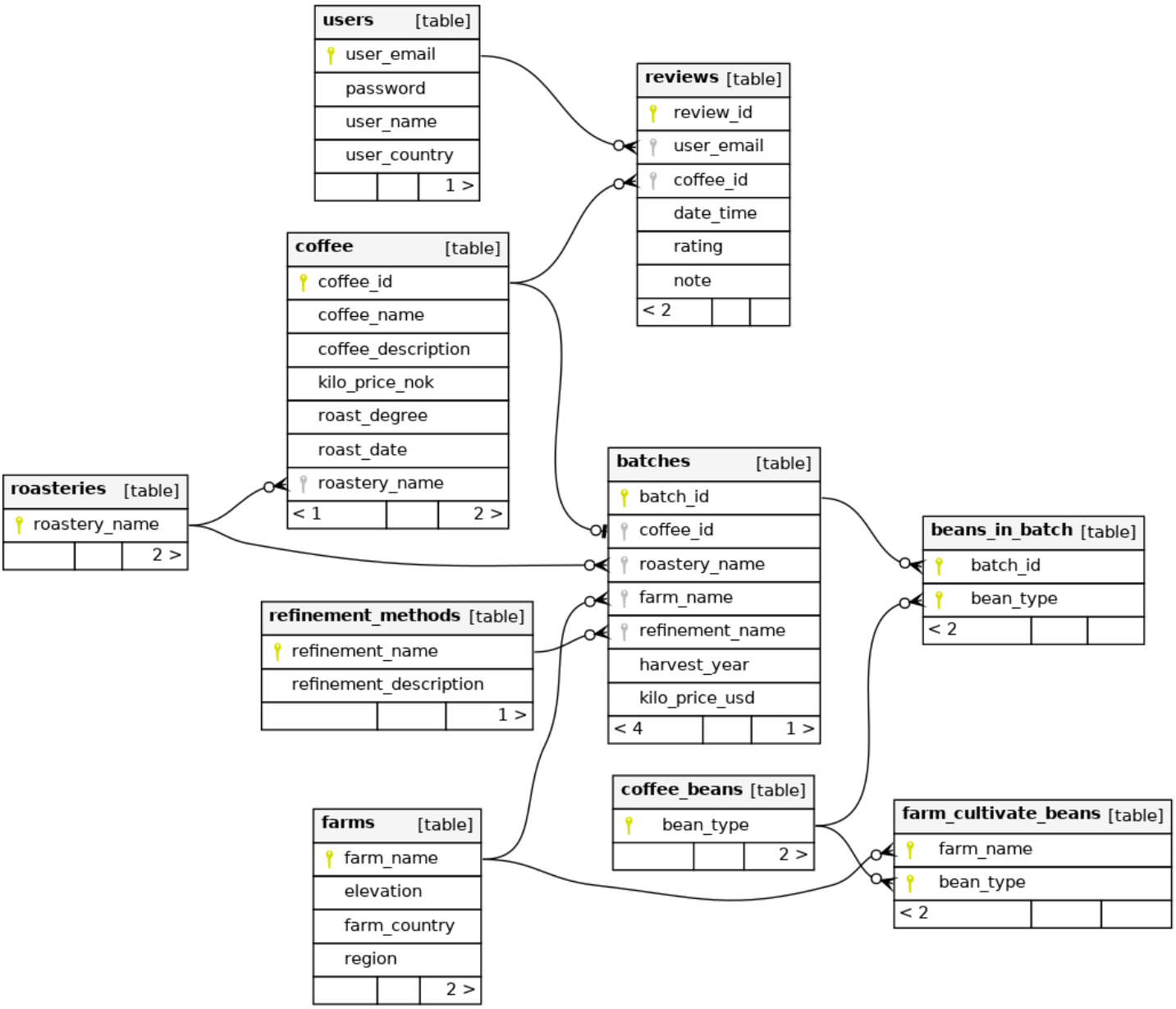
- (`batch_id`, `bean_type`) er sammensatt primærnøkkel i **`beans_in_batch`**.
  - `batch_id` og `bean_type` er fremmednøkler til hhv. **`batches`** og **`coffee_beans`**.
- (`farm_name`, `bean_type`) er sammensatt primærnøkkel i **`farm_cultivate_beans`**.
  - `farm_name` og `bean_type` er fremmednøkler til hhv. **`farms`** og **`coffee_beans`**.

Følgende tabeller har bare én dimensjon og er derfor ikke på normalform:

roasteries(roastery_name)
coffee_beans(bean_type)

- `roastery_name` er primærnøkkel (navn på brenneri, naturlig nøkkel)
- `bean_type` er primærnøkkel (navn på kaffebønne, naturlig nøkkel)

2.2 Diagram


users [table]
user_email
password
user_name
user_country
1 >
reviews [table]
review_id
user_email
coffee_id
date_time
rating
note
< 2
coffee [table]
coffee_id
coffee_name
coffee_description
kilo_price_nok
roast_degree
roast_date
roastery_name
< 1
2 >
roasteries [table]
roastery_name
2 >
batches [table]
batch_id
coffee_id
roastery_name
farm_name
refinement_name
harvest_year
kilo_price_usd
< 4
1 >
beans_in_batch [table]
batch_id
bean_type
< 2
refinement_methods [table]
refinement_name
refinement_description
1 >
coffee_beans [table]
bean_type
2 >
farms [table]
farm_name
elevation
farm_country
region
2 >
farm_cultivate_beans [table]
farm_name
bean_type
< 2

3 Brukerhistorier

3.1 Brukerhistorie 1

> *En bruker smaker kaffen Vinterkaffe 2022 fra Trondheims-brenneriet Jacobsen & Svart (brent 20.01.2022), gir den 10 poeng og skriver «Wow – en odyssé for smaksløkene: sitrusskall, melkesjokolade, aprikos!». Kaffen er lysbrent, bærtørket Bourbon (c. arabica), kommer fra gården Nombre de Dios (1500 moh.) i Santa Ana, El Salvador, har en kilopris på 600 kr og er ifølge brenneriet «En velsmakende og kompleks kaffe for mørketiden». Kaffen ble høstet i 2021 og gården fikk utbetalt 8 USD per kg kaffe. Input fra brukerens side er brenneri, kaffenavn, poeng og smaksnotat.*

Gitt opplysningene har vi følgende informasjon i **`coffee`**-tabellen:

coffee

coffee_id	`int`	1
coffee_name	`varchar`	Vinterkaffe 2022
coffee_description	`text`	'En velsmakende og kompleks kaffe for mørketiden'
kilo_price_nok	`float`	600.0
roast_degree	`varchar`	lys
roast_date	`date`	2022-01-20
roastery_name	`varchar`	Jacobsen & Svart

Brukerens input skal skrives til **`reviews`**-tabellen som har disse kolonnene:

`reviews``( review_id, user_email, coffee_id, date_time, rating, note )`

`user_email` tilsvarer brukerens epost-adresse (primærnøkkel i **`users`**-tabellen). Vi antar at dette kan hentes fra innsendingsskjemaet (f.eks. fordi brukeren er logget på systemet). Ettersom smakstidspunkt ikke er oppgitt, settes `date_time` til nåtid av databasesystemet. Poengsum og smaksnotat hentes fra input.

Da gjenstår det å hente primærnøkkelen til "Vinterkaffe 2022" fra tabellen for kaffetyper (**coffee.**`coffee_id`), slik at **reviews.**`coffee_id` peker på denne kaffetypen.

Databasesystemet vårt krever at kombinasjonen (kaffenavn, brenneri) er unik, altså at det er maks ett tilfelle av en kombinasjon (`coffee_name`, `roastery_name`) i **`coffee`**-tabellen.

Følgende spørring vil derfor gi ønsket `coffee_id`:

```
SELECT coffee_id FROM coffee
WHERE roastery_name = 'Jacobsen & Svart' AND coffee_name = 'Vinterkaffe 2022';
```

Innsetting i database kan gjøres med:

```
INSERT INTO reviews (review_id, user_email, coffee_id, date_time, rating, note)
VALUES (
    null, 'epost@server.no',
    (SELECT coffee_id FROM coffee WHERE coffee_name == 'Vinterkaffe 2022'),
    datetime('now'), 10,
    'Wow - en odyssé for smaksløkene: sitrusskall, melkesjokolade, aprikos!'
);
```

3.2 Brukerhistorie 2

> *En bruker skal kunne få skrevet ut en liste over hvilke brukere som har smakt flest unike kaffer så langt i år, sortert synkende. Listen skal inneholde brukernes fulle navn og antallet kaffer de har smakt.*

Vi kan finne antall unike kaffer som brukere har smakt ved å telle rader i **`reviews`**-tabellen med unik (`user_email`, `coffee_id`). Tuppelen må være unik siden en bruker kan ha smakt samme kaffe mange ganger. Vi kan avgrense tellingen til årets smaker ved å kreve at `date_time` skal være større enn eller lik begynnelsen av året.

Fullt navn hentes fra **`users`**-tabellen via en `JOIN`-operasjon. Resultatet grupperes etter `user_email` for å skille mellom eventuelle brukere med samme navn.

```
SELECT name, count(*)
FROM ( SELECT DISTINCT user_email, coffee_id FROM reviews
       WHERE date_time >= datetime('now', 'start of year') )
JOIN users USING(user_email)
GROUP BY user_email
ORDER BY count(*) DESC;
```

3.3 Brukerhistorie 3

> *En skal kunne se hvilke kaffer som gir forbrukeren mest for pengene ifølge KaffeDBs brukere (høyeste gjennomsnittsscore kontra pris), sortert synkende. Listen skal inneholde brennerinavn, kaffenavn, pris og gjennomsnittsscore for hver kaffe.*

Gjennomsnittsscore beregnes ved å hente rating-verdier fra **`reviews`**-tabellen og gruppere etter kaffetype. Resten av opplysningene ligger i **`coffee`**-tabellen som kan skjøtes på med `JOIN`. Sorteringsnøkkelen kan være gjennomsnittsscore delt på pris.

En spørring kan derfor se slik ut:

```
SELECT roastery_name, coffee_name, kilo_price_nok, AVG(rating)
FROM (SELECT rating, coffee_id FROM reviews)
JOIN coffee USING(coffee_id)
GROUP BY coffee_id
ORDER BY AVG(rating)/kilo_price_nok DESC;
```

3.4 Brukerhistorie 4

> *En bruker søker etter kaffer som er blitt beskrevet med ordet «floral», enten av brukere eller brennerier. Brukeren skal få tilbake en liste med brennerinavn og kaffenavn.*

Tekstsøk kan gjøres med `LIKE '%floral%'` i **reviews.**`note` og **coffee.**`coffee_description`.[1] Samlet liste kan deretter presenteres ved å ta `UNION` av resultatene.

```
SELECT roastery_name, coffee_name
FROM ( SELECT coffee_id FROM reviews
       WHERE note LIKE '%floral%' )
JOIN coffee USING(coffee_id)
UNION
SELECT roastery_name, coffee_name FROM coffee
WHERE coffee_description LIKE '%floral%';
```

[1] Se også SQLite FTS5 Extension som kan vurderes dersom det etter hvert stilles store krav til ytelse.

3.5 Brukerhistorie 5

> *En annen bruker er lei av å bli skuffet av vaskede kaffer og deres tidvis kjedelige smak, og ønsker derfor å søke etter kaffer fra Rwanda og Colombia som ikke er vaskede. Systemet returnerer en liste over brennerinavn og kaffenavn.*

Brukeren ønsker å filtrere kaffetyper på henholdsvis land (**farms.**`farm_country`) og foredlingsmetode (**refinement_method.**`refinement_name`). Denne filtreringen kan i første omgang gjøres med `SELECT`, `FROM` og `WHERE` i de respektive tabellene.

Alle kaffetyper er fremstilt fra et bestemt parti med kaffebønner, som i vår database er representert av tabellen **`batches`**:

`batches`(`batch_id`, `coffee_id`, `roastery_name`, `farm_name`, `refinement_name`, `harvest_year`, `kilo_price_usd`)

Siden både gård (`farm_name`) og foredlingsmetode (`refinement_name`) er fremmednøkler i denne tabellen, vil en `NATURAL JOIN` med tidligere filtrering gi alle parti som matcher brukerens kriterier.

En videre `NATURAL JOIN` med `coffee`-tabellen gir brennerinavn og kaffenavn.

```
SELECT roastery_name, coffee_name
FROM (
        SELECT farm_name FROM farms
        WHERE farm_country IN ('Rwanda', 'Colombia')
    ), (
        SELECT refinement_name FROM refinement_methods
        WHERE refinement_name != 'Vasket'
    )
NATURAL JOIN batches
NATURAL JOIN coffee;
```

4 SQL-script

```
BEGIN TRANSACTION;
PRAGMA foreign_keys=ON;

CREATE TABLE users(
    user_email VARCHAR(255)     PRIMARY KEY,
    password VARCHAR(255)       NOT NULL,
    user_name VARCHAR(255)      NOT NULL,
    user_country VARCHAR(255)   NOT NULL
);

CREATE TABLE coffee(
    coffee_id INTEGER           PRIMARY KEY,
    coffee_name VARCHAR(255)    NOT NULL,
    coffee_description TEXT     NOT NULL,
    kilo_price_nok REAL         NOT NULL,
    roast_degree VARCHAR(7)     CHECK(roast_degree IN('mørk', 'middels', 'lys')),
    roast_date DATE             NOT NULL,
    roastery_name VARCHAR(255)  NOT NULL,
    FOREIGN KEY (roastery_name) REFERENCES roasteries(roastery_name)
                                ON UPDATE CASCADE
                                ON DELETE RESTRICT,
    UNIQUE (coffee_name, roastery_name)
);

CREATE TABLE reviews(
    review_id INTEGER           PRIMARY KEY,
    user_email VARCHAR(255)     NOT NULL,
    coffee_id INTEGER           NOT NULL,
    date_time DATE              NOT NULL,
    rating INTEGER              CHECK(rating BETWEEN 0 AND 10),
    note TEXT,
    FOREIGN KEY (user_email)    REFERENCES users(user_email)
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
    FOREIGN KEY (coffee_id)     REFERENCES coffee(coffee_id)
                                ON DELETE RESTRICT,
```

```
    UNIQUE(user_email, coffee_id, date_time)
);

CREATE TABLE roasteries(
    roastery_name VARCHAR(255)  PRIMARY KEY
);

CREATE TABLE farms(
    farm_name VARCHAR(255)      PRIMARY KEY,
    elevation INTEGER           CHECK(elevation >= 0),
    farm_country VARCHAR(255)   NOT NULL,
    region VARCHAR(255)         NOT NULL
);

CREATE TABLE refinement_methods(
    refinement_name VARCHAR(255) PRIMARY KEY,
    refinement_description TEXT NOT NULL
);

CREATE TABLE batches(
    batch_id INTEGER            PRIMARY KEY,
    coffee_id INTEGER           NOT NULL,
    roastery_name VARCHAR(255)  NOT NULL,
    farm_name VARCHAR(255)      NOT NULL,
    refinement_name VARCHAR(255) NOT NULL,
    harvest_year INTEGER        NOT NULL,
    kilo_price_usd REAL         NOT NULL,
    FOREIGN KEY (coffee_id)     REFERENCES coffee(coffee_id)
                                ON DELETE CASCADE,
    FOREIGN KEY (farm_name)     REFERENCES farms(farm_name)
                                ON UPDATE CASCADE
                                ON DELETE RESTRICT,
    FOREIGN KEY (roastery_name) REFERENCES roasteries(roastery_name)
                                ON UPDATE CASCADE
                                ON DELETE RESTRICT,
    FOREIGN KEY (refinement_name) REFERENCES refinement_methods(refinement_name)
                                ON UPDATE CASCADE
                                ON DELETE RESTRICT,
    UNIQUE(coffee_id)
```

```
);

CREATE TABLE coffee_beans(
    bean_type VARCHAR(8)        PRIMARY KEY
                                CHECK(bean_type IN ('arabica', 'robusta', 'liberica'))
);

CREATE TABLE beans_in_batch(
    batch_id INTEGER            NOT NULL,
    bean_type VARCHAR(8)        NOT NULL,
    FOREIGN KEY (batch_id)      REFERENCES batches(batch_id)
                                ON DELETE CASCADE,
    FOREIGN KEY (bean_type)     REFERENCES coffee_beans(bean_type)
                                ON UPDATE CASCADE
                                ON DELETE RESTRICT,
    PRIMARY KEY(batch_id, bean_type)
);

CREATE TABLE farm_cultivate_beans(
    farm_name VARCHAR(255)      NOT NULL,
    bean_type VARCHAR(8)        NOT NULL,
    FOREIGN KEY (farm_name)     REFERENCES farms(farm_name)
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
    FOREIGN KEY (bean_type)     REFERENCES coffee_beans(bean_type)
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
    PRIMARY KEY(farm_name, bean_type)
);

COMMIT;
```