Hluti 1: Ættir og landsvæði í Norður konungsríkinu

Jakob Stefán Ívarsson

2024-10-10

Hluti 1: Ættir og landsvæði í Norður konungsríkinu

Spurning 1: Tengja konungsríki við hús í Game of Thrones heiminum

Í þessari fyrirspurn notum við `WITH` setningu til að búa til CTE (Common Table Expression) sem heitir `kingdoms_houses`. CTE býr til tímabundna töflu sem tengir hús úr töflunni `got.houses` við konungsríki úr `atlas.kingdoms` út frá því hvaða landsvæði hvert hús tilheyrir. Við framkvæmum LEFT JOIN til að tryggja að öll hús komi fram, jafnvel þó þau hafi ekki samsvörun við konungsríki.

```
WITH kingdoms_houses AS (
    SELECT
        k.gid AS kingdom_id,
        h.id AS house_id,
        k.name AS kingdom_name,
        h.name AS house_name
    FROM
        got.houses h
    LEFT JOIN
        atlas.kingdoms k
    ON
        k.name = h.region
)
```

Því næst er INSERT ... ON CONFLICT notað til að uppfæra eða bæta við gögnunum í martell.tables_mapping töfluna.

```
INSERT INTO martell.tables_mapping (kingdom_id, house_id)
SELECT kingdom_id, house_id
FROM kingdoms_houses
ON CONFLICT (house_id)
DO UPDATE
SET kingdom_id = EXCLUDED.kingdom_id;
```

Þetta tryggir að ef samsvörun er til staðar (mismunandi gögn fyrir sama house_id), þá uppfærast gögnin í töflunni.

Dæmi: Ef við höfum t.d. “House Stark of Winterfell” sem er í “The North”:

Kóðinn mun sjá að landsvæðið “The North” í got.houses passar við konungsríkið “The North” í atlas.kingdoms. Kóðinn tengir saman house_id fyrir “House Stark of Winterfell” og kingdom_id fyrir “The

North" í tímabundnu töflunni kingdoms_houses. Þegar gögnin eru sett inn í martell.tables_mapping töfluna, þá mun það skrá samsvörunina. Ef "House Stark of Winterfell" er nú þegar skráð með öðru konungsríki, þá mun það uppfæra gögnin til að sýna rétt konungsríki. Þannig myndar þetta töflu sem sýnir öll 444 húsin og þeirra samsvarandi konungsríki.

Spurning 2: Finna og tengja staði og hús

Í þessari fyrirspurn er markmiðið að finna gagntæka vörpun (one-to-one mapping) á milli staða úr `atlas.locations` og húsa úr `got.houses`. Fyrst er CTE sem heitir `location_house_mapping` skilgreint. Við notum ýmis skilyrði í `CASE` setningunni til að ákveða forgangsröðun á samsvörun (1-5) og notum `COALESCE` til að velja besta samsvörun. Forgangsröðunin gengur út á það hversu sterk samsvörunin er s.s. ef hún finnst strax með samanburð á hús nöfnunum og location staðsetningunum eða ef það þarf að skoða nánari dálka. Þannig finnur kóðinn samsvörun milli húsa og staðsetninga og leitar af samsvörun. Kóðinn athugar t.d. hvort hús nafnið finnst í location nöfnunum ef ekki þá skoðar hann t.d. seats, titles eða summary dálkinn. Summary dálkurinn hefur minnsta forgang.

```
WITH location_house_mapping AS (
    SELECT
        l.gid AS location_id,
        h.id AS house_id,
        l.name AS location_name,
        h.name AS house_name,
        h.region AS house_region,
        h.seats,
        h.titles,
        l.summary AS location_summary,
        CASE
            WHEN h.name ILIKE '%' || l.name || '%' AND
                 h.name ~* ('\m' || l.name || '\M') THEN 1
            WHEN EXISTS (
                SELECT 1
                FROM UNNEST(h.seats) AS seat
                WHERE seat ILIKE '%' || l.name || '%'
                AND seat ~* ('\m' || l.name || '\M')
            ) THEN 2
            WHEN EXISTS (
                SELECT 1
                FROM UNNEST(h.titles) AS title
                WHERE title ILIKE '%' || l.name || '%'
                AND title ~* ('\m' || l.name || '\M')
            ) THEN 3
            WHEN l.summary ILIKE '%' || h.name || '%' AND
                 l.summary ~* ('\m' || h.name || '\M') THEN 4
            ELSE 5
        END AS match_priority,
        COALESCE(
            (SELECT seat FROM UNNEST(h.seats) AS seat
             WHERE seat ILIKE '%' || l.name || '%'
             AND seat ~* ('\m' || l.name || '\M') LIMIT 1),
            (SELECT title FROM UNNEST(h.titles) AS title
             WHERE title ILIKE '%' || l.name || '%'
             AND title ~* ('\m' || l.name || '\M') LIMIT 1),
            h.name
```

```
        ) AS matched_detail
    FROM
        atlas.locations l
    LEFT JOIN
        got.houses h
    ON
        (h.name ILIKE '%' || l.name || '%' AND h.name ~* ('\m' || l.name || '\M'))
        OR EXISTS (
            SELECT 1
            FROM UNNEST(h.seats) AS seat
            WHERE seat ILIKE '%' || l.name || '%'
            AND seat ~* ('\m' || l.name || '\M')
        )
        OR EXISTS (
            SELECT 1
            FROM UNNEST(h.titles) AS title
            WHERE title ILIKE '%' || l.name || '%'
            AND title ~* ('\m' || l.name || '\M')
        )
        OR (l.summary ILIKE '%' || h.name || '%' AND l.summary ~* ('\m' || h.name || '\M'))
)
```

Við notum síðan ROW_NUMBER til að reikna raðnúmer innan hvers staðar og húss og fá þannig eina samsvörun fyrir hvern stað. Þaðan af framkvæmum við upsertu líkt og í fyrra dæmi.

```
INSERT INTO martell.tables_mapping (house_id, location_id)
SELECT house_id, location_id
FROM filterum
WHERE row_num = 1 AND house_row_num = 1
ON CONFLICT (house_id)
DO UPDATE
SET location_id = EXCLUDED.location_id;
```

Dæmi: Segjum að við höfum staðinn “Winterfell” í atlas.locations og húsið “House Stark of Winterfell” í got.houses.

Kóðinn mun athuga:

Passar nafn hússins “House Stark of Winterfell” við staðinn “Winterfell”? Já, því það inniheldur nafnið “Winterfell”. Þetta gefur forgang 1. Ef ekki skoðar hann fleiri dálka s.s. seats og titles og að lokum summary. Ef engin önnur betri samsvörun finnst, mun hann nota þessa samsvörun til að fylla inn matched_detail (til að sýna hvað það var sem myndaði samsvörunina).

Framhald spurning 2 - Niðurstöður um Norðrið

Þessi hluti sýnir einnig samsvörun staða og húsa en núna einungis í Norðrinu þ.e. region dálknum “The North”.

```
WITH location_house_mapping AS (
    SELECT
        l.gid AS location_id, --ýta hér á kóðann til að fá upp réttu statement skipanir í keyrsluhnapp
        h.id AS house_id,
        l.name AS location_name,
```

```
        h.name AS house_name,
        h.region AS house_region,
        h.seats,
        h.titles,
        l.summary AS location_summary,
        CASE
            WHEN h.name ILIKE '%' || l.name || '%' AND
                 h.name ~* ('\m' || l.name || '\M') THEN 1
            WHEN EXISTS (
                SELECT 1
                FROM UNNEST(h.seats) AS seat
                WHERE seat ILIKE '%' || l.name || '%'
                AND seat ~* ('\m' || l.name || '\M')
            ) THEN 2
            WHEN EXISTS (
                SELECT 1
                FROM UNNEST(h.titles) AS title
                WHERE title ILIKE '%' || l.name || '%'
                AND title ~* ('\m' || l.name || '\M')
            ) THEN 3
            WHEN l.summary ILIKE '%' || h.name || '%' AND
                 l.summary ~* ('\m' || h.name || '\M') THEN 4
            ELSE 5
        END AS match_priority,
        COALESCE(
            (SELECT seat FROM UNNEST(h.seats) AS seat
             WHERE seat ILIKE '%' || l.name || '%'
             AND seat ~* ('\m' || l.name || '\M') LIMIT 1),
            (SELECT title FROM UNNEST(h.titles) AS title
             WHERE title ILIKE '%' || l.name || '%'
             AND title ~* ('\m' || l.name || '\M') LIMIT 1),
            h.name
        ) AS matched_detail
    FROM
        atlas.locations l
    LEFT JOIN
        got.houses h
    ON
        (h.name ILIKE '%' || l.name || '%' AND h.name ~* ('\m' || l.name || '\M'))
        OR EXISTS (
            SELECT 1
            FROM UNNEST(h.seats) AS seat
            WHERE seat ILIKE '%' || l.name || '%'
            AND seat ~* ('\m' || l.name || '\M')
        )
        OR EXISTS (
            SELECT 1
            FROM UNNEST(h.titles) AS title
            WHERE title ILIKE '%' || l.name || '%'
            AND title ~* ('\m' || l.name || '\M')
        )
        OR (l.summary ILIKE '%' || h.name || '%' AND l.summary ~* ('\m' || h.name || '\M'))
),
```

```
filterum AS (
    SELECT
        location_id,
        house_id,
        ROW_NUMBER() OVER (PARTITION BY location_id ORDER BY match_priority, house_name) AS row_num,
        ROW_NUMBER() OVER (PARTITION BY house_id ORDER BY match_priority, location_name) AS house_row_nu
    FROM
        location_house_mapping
    WHERE match_priority < 5
)
SELECT *
FROM location_house_mapping
WHERE match_priority < 4
AND house_region = 'The North';
```

Dæmi (Svipað og fyrra dæmi):

Ef við höfum staðinn "Winterfell" og húsið "House Stark of Winterfell":

Fyrst athugar kóðinn hvort nafn hússins samsvarar nafni staðarins. Ef það passar, gefur hann forgang 1. Ef ekki, athugar hann hvort eitthvað sæti (seat) eða titill (title) inniheldur nafnið "Winterfell" og gefur þá forgang 2 eða 3. Ef það er engin samsvörun þar, skoðar hann lýsinguna (summary) og gefur forgang 5 ef það passar.

Spurning 3: Finna stærstu ættir norðanmanna

Í þessari fyrirspurn er markmiðið að finna stærstu ættirnar í Norðrinu, miðað við persónur sem eru hliðhollar „The North" og hafa fleiri en 5 meðlimi. Fyrst er skilgreint CTE sem tengir saman ættir og persónur úr `got.houses` og `got.characters`.

Hér var til dæmis notað "UNNEST(sworn_members) AS member_id" til að brjóta niður fylki (array) af hliðhollum meðlimum hvers húss í einingar. Þetta gerir það að verkum að hver lína í niðurstöðunni hefur eitt house_id og einn member_id fyrir hvern hliðhollan meðlim.

Við notum LEFT JOIN got.characters c ON nh.member_id = c.id: Hér tengjum við northern_houses við got.characters með því að nota member_id til að finna samsvarandi persónu í got.characters. LEFT JOIN er notað til að tryggja að við fáum allar línur úr northern_houses, jafnvel þó að það sé engin samsvörun í got.characters.

Síðan gerum við SPLIT_PART(c.name, ' ', ARRAY_LENGTH(REGEXP_SPLIT_TO_ARRAY(c.name,' '), 1)) AS family: Hér er SPLIT_PART og REGEXP_SPLIT_TO_ARRAY notað til að draga út ættarnafn persónunnar. Þetta virkar þannig að það skiptir nafninu upp í hluta (miðað við bil) og tekur síðasta hlutann (ættarnafnið).Þannig Eddard Stark -> Stark.

```
WITH northern_houses AS (
    SELECT
        id AS house_id,
        name AS house_name,
        UNNEST(sworn_members) AS member_id
    FROM
        got.houses
    WHERE
        region = 'The North'
),
northern_characters AS (
```

```
    SELECT
        nh.house_name,
        c.id AS character_id,
        c.name AS character_name,
        SPLIT_PART(c.name, ' ', ARRAY_LENGTH(REGEXP_SPLIT_TO_ARRAY(c.name, ' '), 1)) AS family
    FROM
        northern_houses nh
    LEFT JOIN
        got.characters c
    ON
        nh.member_id = c.id
),
missing_characters
    AS (
    SELECT
        nh.house_name,
        nh.member_id
    FROM
        northern_houses nh
    LEFT JOIN
        got.characters c
    ON
        nh.member_id = c.id
    WHERE
        c.id IS NULL
),
family_count AS (
    SELECT
        family,
        COUNT(DISTINCT character_id) AS member_count
    FROM
        northern_characters
    WHERE
        character_id IS NOT NULL
    GROUP BY
        family
    HAVING
        COUNT(DISTINCT character_id) > 5
)

SELECT
    family,
    member_count
FROM
    family_count
ORDER BY
    member_count DESC,
    family ASC;
```

Dæmi um hvernig kóðinn virkar: Við höfum t.d. House Stark of Winterfell. Þar getum við séð að þeir hafa ótal mörg member_id í sworn_members dálkinum í got.houses. Member_id eru skipt upp í t.d. fylki: {30,36,37} þar sem hver tala er karakter í got.characters. Kóðinn athugar þessa karaktera miðað við skilyrðin af eftirnafni. Þannig í þessu tilfelli skoðar hann Stark eftirnafnið. Kóðinn rennur þá í gegnum listann og

telur hversu margir í got.characters eru með Stark eftirnafnið miðað við þessi member_id sem eru í fylkinu. Hann rennur í gegn og sér til dæmis Eddard Stark, Jon Snow, Catelyn Stark. Af þessum þremur myndi hann telja einungis 2 og skrá þá niður. Þetta gerir hann fyrir öll id í sworn_members fyrir öll húsin í norðrinu sem urðu tekin fyrir í samsvöruninni og skráir þá niður ef fjöldinn sem ber ákveðið eftirnafn er fleiri en 5. Við sjáum síðan í lok kóðans þar hópar hann niðurstöðurnar eftir ættarnafni og raðar þeim fyrst eftir fjölda meðlima í lækkandi röð og síðan í stafrófsröð ef fjöldinn er sá sami.