Advanced SQL queries

D'Ambrosi Denis

October 16, 2023

Exercise 1

```
SELECT XMLELEMENT(
    NAME photos,
    XMLAGG(nester.nested)
) FROM (
    SELECT XMLELEMENT(
        NAME photo,
        XMLATTRIBUTES(url, titel AS title, photo.
            personemail AS owner),
        XMLAGG(
            XMLELEMENT(
                NAME shown,
                XMLATTRIBUTES(vorname AS firstname,
                    nachname AS lastname))
        )
    ) AS nested
    FROM photo, istabgebildet, person
    WHERE istabgebildet.photourl = photo.url AND
        istabgebildet.personemail = person.email
    GROUP BY photo.url
) AS nester;
```

Exercise 2

```
SELECT JSON_BUILD_OBJECT('photos', JSON_AGG(nested))
FROM (
    SELECT  url, titel AS title,
            photo.personemail AS owner,
            JSON_AGG(
                JSON_BUILD_OBJECT('firstname', person.
                    nachname, 'lastname', person.vorname)
            ) AS shown
    FROM photo, istabgebildet, person
    WHERE istabgebildet.photourl = photo.url AND
       istabgebildet.personemail = person.email
    GROUP BY photo.url
) AS nested;
```

Exercise 3

```sql
CREATE TABLE myFiles (
    key varchar(255) PRIMARY KEY,
    content jsonb
);

INSERT INTO myFiles (key, content)
SELECT 'photos' AS key, JSONB_BUILD_OBJECT('photos',
    JSON_AGG(photos)) AS content
FROM (
    SELECT  url, titel AS title,
            photo.personemail AS owner,
            JSON_AGG(
                JSON_BUILD_OBJECT('firstname', person.
                    nachname, 'lastname', person.vorname)
            ) AS shown
    FROM photo, istabgebildet, person
    WHERE istabgebildet.photourl = photo.url AND
        istabgebildet.personemail = person.email
    GROUP BY photo.url
) AS photos;

SELECT photo->>'url' AS url, photo->>'owner' AS owner
FROM (
    SELECT JSONB_ARRAY_ELEMENTS(content->'photos') AS
        photo
    FROM myFiles
    WHERE key = 'photos'
)
WHERE photo->'shown' @> '[{ "lastname": "Lea", "firstname
    ": "Meyer" }]'::jsonb;
```

Exercise 4

```
SELECT name, "numEmpl", MAX("numEmpl") OVER (PARTITION BY
    "parentId") AS "maxNumEmpl"
FROM department;
```

Exercise 5

```
WITH RECURSIVE dep AS (
        SELECT *
        FROM department
        WHERE "parentId" = 4 OR "deptId" = 4
    UNION
        SELECT department.*
        FROM department, dep
        WHERE department."parentId" = dep."deptId"
)
SELECT *
FROM dep;

SELECT "deptId", name, MAX("totalEmpl")
FROM (
    WITH RECURSIVE dep AS (
            SELECT *
            FROM department
        UNION ALL
            SELECT department."deptId", department.name,
                dep."parentId", department."numEmpl"
            FROM department
            INNER JOIN dep ON department."parentId" = dep
                ."deptId"
            WHERE department."parentId" <> department."
                deptId"
    )
    SELECT  department."deptId",
            department.name,
            d."totalEmpl" + department."numEmpl" AS "
                totalEmpl"
    FROM department
    INNER JOIN (
        SELECT "parentId", SUM("numEmpl") AS "totalEmpl"
        FROM dep
        GROUP BY "parentId"
    ) AS d
    ON department."deptId" = d."parentId"
    UNION
    SELECT "deptId", name, "numEmpl"
    FROM department
)
GROUP BY "deptId", name;
```