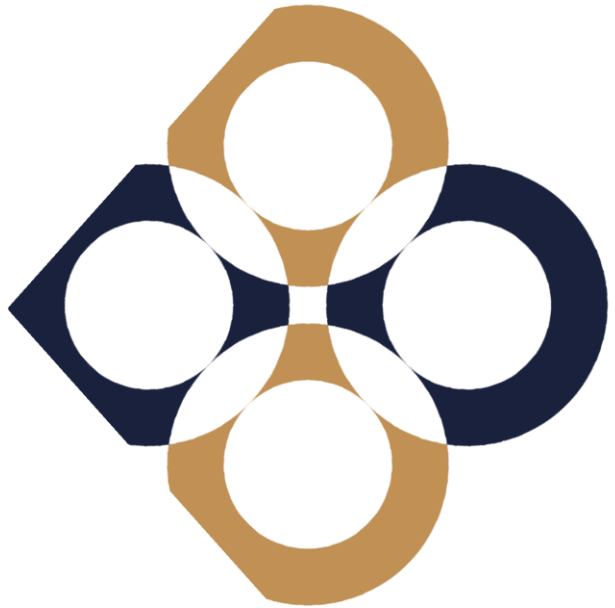
BUDAPESTI CORVINUS EGYETEM

Adatbázisok

Gyakorlat 06 –
Beágyazott lekérdezések



Beágyazott lekérdezés (Subquery, al-lekérdezés, alkérdés)

Lekérdezés a lekérdezésen belül

Rendszerint egy (külső) SELECT utasításon belüli (belső) SELECT utasítást jelent*

Először a belső SELECT fut le, majd annak eredményét megkapva a külső SELECT hajtódik végre**

Egy külső SELECT-be több belső SELECT is beágyazható

*Lekérdezést beágyazhatunk az INSERT, DELETE és UPDATE utasításokba is, de ezekkel most nem foglalkozunk

** A korrelált alkérdéseknél ez soronként történik

Beágyazott lekérdezés tipikus formája + példa

```
SELECT    select_list
FROM      table
WHERE     expr operator
                (SELECT    select_list
                 FROM      table);
```

```
SELECT ProductID,
       Name,
       ListPrice
FROM   production.Product
WHERE  ListPrice > (SELECT AVG(ListPrice)
                    FROM   Production.Product)
```

subquery

Beágyazott lekérdezések csoportosítása

Milyen eredményt ad vissza a beágyazott lekérdezés?

- Egy értéket (scalar)
- Több értéket (multi-valued)
- Táblát (table-valued)*

Hivatkozik-e a belső SELECT a külső SELECT valamely oszlopára?

- Ha igen, akkor korrelált alkérdésről beszélünk (correlated subquery)
- Ha nem, akkor önálló alkérdezésről beszélünk (self-contained subquery)

* Ezzel nem foglalkozunk

Hová kerülhet a beágyazott lekérdezés?

- A SELECT részbe – SELECT oszlop1, oszlop2, (subquery), oszlop4
- A FROM részbe – SELECT s.oszloplista FROM (subuqery) s
- **A WHERE részbe*** - lásd 3. dia
- A **HAVING részbe** - SELECT … HAVING kifejezés operátor (subquery)

* Ez a leggyakoribb eset, ezért főleg ezzel foglalkozunk. Kisebb súllyal, de a HAVING-es alkérdés is szerepelni fog a példák és feladatok között

Milyen tipikus esetekben használhatunk beágyazott lekérdezést?

- Ha szeretnénk összehasonlítani egy kifejezés értékét a beágyazott lekérdezés eredményével (legtöbbször <, >, =)
- Ha szeretnénk eldönteni, hogy egy kifejezés eredménye benne van-e a beágyazott lekérdezés eredményhalmazában (IN)
- Ha szeretnénk eldönteni, hogy a beágyazott lekérdezés eredményhalmaza üres-e (EXISTS)

Önálló alkérdés - összehasonlítás

Melyek azok a rendelési tételek, amelyek rendelési mennyisége az átlagos rendelési mennyiségnél nagyobb?

```
SELECT *
FROM rendeles_tetel
WHERE mennyiseg >
(
 SELECT AVG(mennyiseg)
 FROM rendeles_tetel
)
```

Önálló alkérdés – összehasonlítás + ANY, ALL*

Az ANY operátor igaz értéket ad vissza, ha az összehasonlítás eredménye az alkérdés legalább egy eredménysorára teljesül
Az ALL operátor igaz értéket ad vissza, ha az összehasonlítás eredménye az alkérdés minden eredménysorára teljesül

Példa:
Melyek azok a termékek, amelyek nem a legolcsóbbak (listaáruk nem a legkisebb)

```
SELECT megnevezes
FROM Termek
WHERE listaar > ANY
(
  SELECT listaar
  FROM Termek
)
```

*A SOME operátorral nem foglalkozunk

Önálló alkérdés - IN

Melyek azok az ügyfelek, akik már adtak le rendelést?

```
SELECT Nev
FROM Ugyfel
WHERE [login] IN
(
 SELECT DISTINCT [login]
 FROM rendeles
)
```

Korrelált alkérdés - Összehasonlítás

Melyek azok a termékek, amelyek listaára kategóriájukban a legmagasabb?

```
SELECT t.termekkod, t.MEGNEVEZES FROM
Termek t
WHERE t.LISTAAR = (
  SELECT max(t2.LISTAAR)
  FROM Termek t2
  WHERE t.KAT_ID = t2.KAT_ID
)
```

Korrelált alkérdés – Összehasonlítás + ANY, ALL

Melyek azok az a termékek, amelyek saját raktárukban a legolcsóbbak?

```
SELECT t.TERMEKKOD, t.megnevezes
FROM Termek t
WHERE t.listaar <= ALL (
  SELECT t2.listaar
  FROM Termek t2
  WHERE t.RAKTAR_KOD = t2.RAKTAR_KOD
)
```

Korrelált alkérdés - IN

Listázzuk azon ügyfeleket, akik rendeltek már ,’ Esküvői meghívó terméket!

```
SELECT u.NEV
FROM Ugyfel u
WHERE 'Esküvői meghívó' IN
(
    SELECT t.megnevezes
    FROM Rendeles r
            JOIN Rendeles_Tetel rt ON r.SORSZAM = rt.SORSZAM
            JOIN Termek t ON rt.TERMEKKOD = t.TERMEKKOD
    WHERE u.LOGIN = r.LOGIN
)
```

Korrelált alkérdés - EXISTS

Az EXISTS operátor igaz értéket ad vissza, ha a beágyazott SELECT eredményhalmaza nem üres

Példa:
Melyek azok a termékek, amelyekből legalább egyszer rendeltek már 50 darabnál többet?

```
SELECT t.megnevezes
from Termek t
where EXISTS (
    SELECT *
    FROM Rendeles_tetel rt
    WHERE rt.TERMEKKOD = t.TERMEKKOD
            AND rt.MENNYISEG > 50

)
```

Alkérdés - HAVING

Példa:
Melyek azok az ügyfelek, amelyek 2017-ben többször rendeltek, mint 2016-ban?
Elég az ügyfelek azonosítóját (LOGIN) megjeleníteni!

```
SELECT u.LOGIN
FROM Rendeles r JOIN Ugyfel u ON r.LOGIN = u.LOGIN WHERE
YEAR(rend_datum)=2017
GROUP BY u.login
HAVING COUNT(*)> (
    SELECT COUNT(*)
    FROM Rendeles r2 JOIN Ugyfel u2 ON r2.LOGIN = u2.LOGIN
    WHERE YEAR(rend_datum)=2016 AND u2.LOGIN = u.LOGIN
)
```

Beágyazott lekérdezés – fontosabb korlátozások

- Mindig zárójelbe kell tenni
- Összehasonlítás esetén mindig a reláció jobb oldalán áll
- Nem lehet benne ORDER BY*, INTO
- Ha van benne GROUP BY, akkor nem lehet benne DISTINCT
- Ha csak egy értéket ad vissza, akkor nem lehet benne GROUP BY és HAVING sem
- A visszaadott érték(ek)nek (join) kompatibilisnek kell lennie a külső SELECT WHERE feltételével
- Bizonyos adattípusok nem használhatók (ntext, text, image)

* Kivéve, ha TOP, FOR XML vagy OFFSET is szerepel az alkérdésben

Megjegyzések

- A beágyazott lekérdezések helyett többnyire más megoldást is használhatunk (pl: JOIN)
- A beágyazott lekérdezések átláthatóbbá teszik a kódot, viszont performacia szempontjából nem a legjobbak
- Ugyanaz a feladat sokszor többféle operátor használatával is megoldható (pl: IN, EXISTS).
- Nagyobb rekordszám esetén performancia szempontjából legtöbbször az EXISTS a legjobb választás
- Az IN és az EXISTS operátorok tagadhatók is (NOT IN, NOT EXISTS)
- A beágyazott lekérdezések egymásba is ágyazhatók

A gyakorlaton használt webshop adatbázis

Köszönöm a figyelmet!