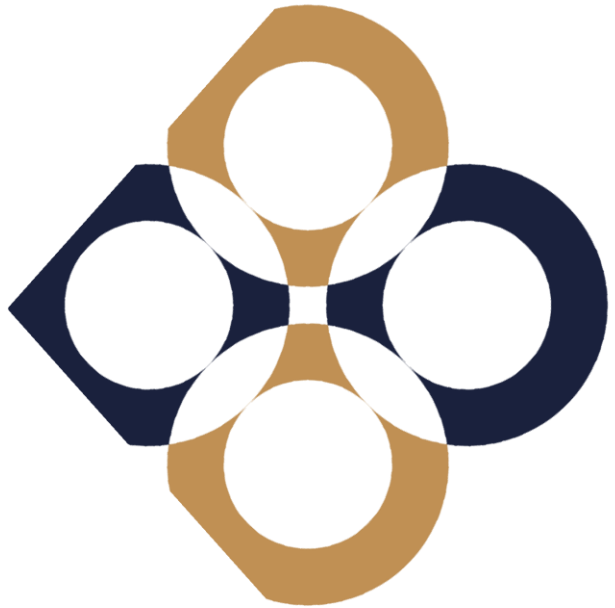
BUDAPESTI CORVINUS EGYETEM

Adatbázisok gyakorlat 05

Partíciók. Ablakok. Analitikus függvények



Partíciók

Azon rekordok csoportja, amelyeken az aggregálást el kell végezni

A GROUP BY alternatívái
Formája*:

OVER(
PARTITION BY kifejezés
)

Példa:
Jelenítsük meg a termékek kódja és listaára mellett a termékkategória átlagárát is!

```
SELECT TERMEKKOD, LISTAAR,
        AVG(LISTAAR) OVER(PARTITION BY KAT_ID)
        AS 'Kategória átlagár' FROM
Termek
```

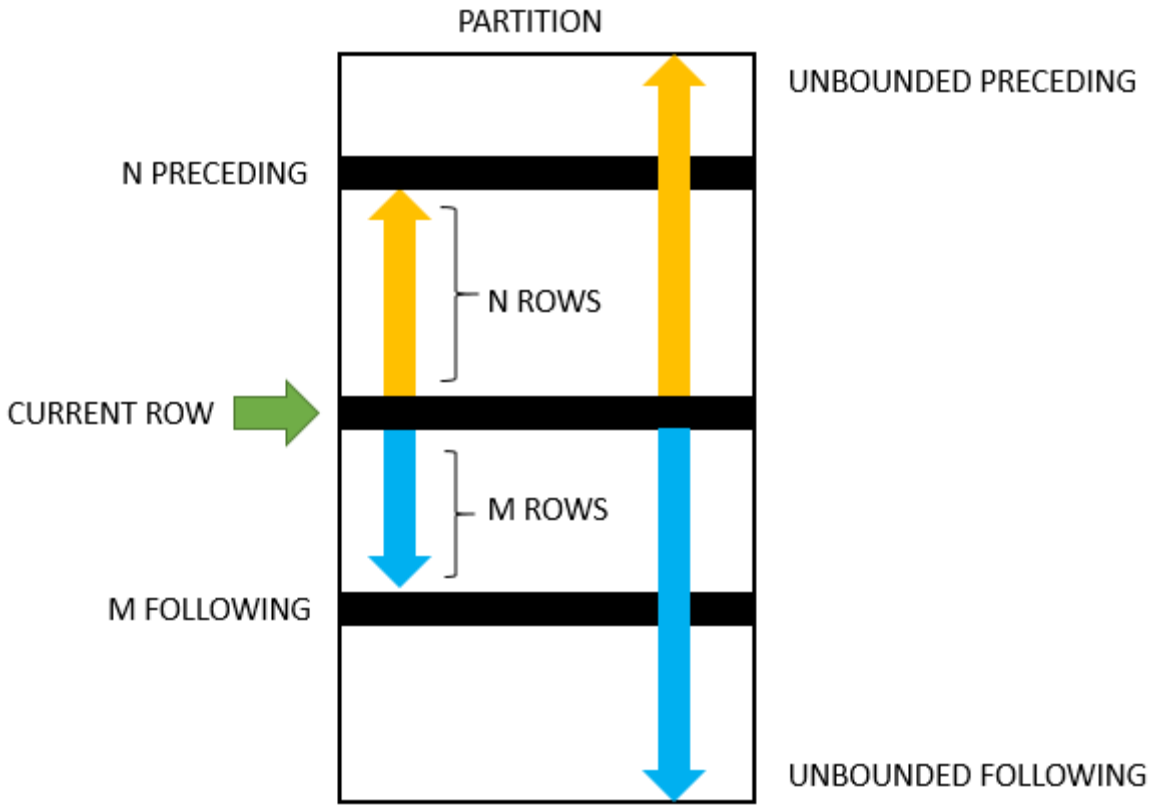
*A partíció kiegészíthető rendezéssel is (lásd következő diák):
OVER (PARTITION BY kifejezés ORDER BY kifejezés)

Ablakok létrehozása A ROWS és RANGE segítségével

ROWS, RANGE: az ablakot (partíció elejét és végét) határozzák meg
Használatukhoz az ORDER BY rész kötelező!

Formája:

OVER(PARTITION BY kifejezés*
ORDER BY kifejezés*
ROWS | RANGE BETWEEN
kezdőpont AND végpont)



* A kifejezés - itt és az összes többi utasítás/függvény leírásban - a gyakorlatban többnyire oszlopnevet vagy oszlopnevek listáját jelenti

ROWS

A ROWS az ablak méretét **fizikailag** adja meg
(legtöbbször az aktuális sort megelőző és/vagy követő sorok számát konkrétan megadja)
Kezdőpont, végpont lehet: CURRENT ROW, n PRECEDING, n FOLLOWING.
Speciálisan: UNBOUNDED PRECEDING (kezdőpont), UNBOUNDED FOLLOWING (végpont)*

Formája:
OVER(
PARTITION BY kifejezés
ORDER BY kifejezés
ROWS BETWEEN kezdőpont
AND végpont
)

* A partíció legelső, illetve legutolsó sorát jelentik meg

Példa:
Listázzuk az egyes megrendelések dátumát, a termék kódját és mennyiségét, valamint a sorszám szerinti előző 5 megrendelés átlagos mennyiségét is!

```
SELECT rt.TERMEKKOD, r.REND_DATUM, rt.MENNYISEG,
        AVG(rt.MENNYISEG) OVER(PARTITION BY
        rt.TERMEKKOD ORDER BY r.SORSZAM ROWS BETWEEN
        5 PRECEDING AND 1 PRECEDING)
         AS 'ELőző 5 rendelés mennyiség átlaga' FROM
Rendeles_tetel rt
    JOIN Rendeles r ON r.SORSZAM = rt.SORSZAM
```

RANGE

A RANGE az ablak méretét **logikailag** adja meg
(nem a sorok számát adja meg, hanem a legelső, legutolsó vagy az aktuális sort, mint az intervallum kezdő- vagy végpontját)
Kezdőpont, végpont lehet: CURRENT ROW, UNBOUNDED PRECEEDING (kezdőpont) és UNBOUNDED FOLLOWING (végpont)

Formája:

OVER(
PARTITION BY kifejezés
ORDER BY kifejezés
RANGE BETWEEN kezdőpont
AND végpont
)

Példa:
Jelenítsük meg, hogy az egyes ügyfelek az adott rendelési dátumig bezárólag összesen hányszor rendeltek!
Megjelenítendő a rendelés dátuma, az ügyfél login-ja és a rendelés darabszáma

```
SELECT DISTINCT REND_DATUM,[LOGIN], COUNT(*)
        OVER(PARTITION BY [LOGIN] ORDER BY
        REND_DATUM RANGE BETWEEN UNBOUNDED
        PRECEDING AND CURRENT ROW)
        AS 'Eddigi rendeléseinek száma' FROM
Rendeles
ORDER BY REND_DATUM, [LOGIN]
```

ROW_NUMBER()

A lekérdezés eredménysoraihoz sorszámokat rendel.

Formája:

ROW_NUMBER()
OVER (
PARTITION BY kifejezés
ORDER BY kifejezés)

Példa:
Készítsünk sorszámozott listát nemenként az ügyfelekről! A sorszámozás szempontja az ügyfél email-címe legyen!

```
SELECT ROW_NUMBER() OVER(PARTITION BY nem
            ORDER BY email)
            AS 'Nemenkénti sorszám', * FROM
Ugyfel
```

A ROW_NUMBER() mindig **szigorúan monoton növekvő** számokat ad vissza!
Több partíció esetén a sorszámozás minden partíciónál újra kezdődik.

RANK()

Megadja, hogy az adott rekord hányadik a partícióban az adott rendezettség szerint.*

Formája:

RANK()
OVER (
PARTITION BY kifejezés
ORDER BY kifejezés)

Példa:
Listázzuk a termékek kódját, megnevezését, kategória kódját, készlet mennyiségét és azt, hogy a termék a készlet alapján hányadik a kategóriájában

```
SELECT TERMEKKOD, MEGNEVEZES, KAT_ID, KESZLET, RANK()
OVER (PARTITION BY KAT_ID
ORDER BY KESZLET DESC)
AS 'Készlet szerinti helyezés kategóriájában' FROM Termek
```

* A RANK() mindig **monoton növekvő számokat** ad vissza!
- Az azonos értékű sorok ugyanazt a sorszámot kapják.
- A következő sorszám az aktuálisnál annyival lesz nagyobb, ahány azonos értékű sor van.

DENSE_RANK()

Megadja, hogy az adott rekord hányadik a partícióban az adott rendezettség szerint.*

Formája:

ROW_NUMBER()
OVER (
PARTITION BY kifejezés
ORDER BY kifejezés)

Példa
Az előző példa DENSE_RANK() függvénnyel

```
SELECT TERMEKKOD, MEGNEVEZES, KAT_ID, KESZLET,
        DENSE_RANK() OVER (PARTITION BY KAT_ID
        ORDER BY KESZLET DESC)
    AS 'Készlet szerinti helyezés kategóriájában' FROM Termek
```

* A DENSE_RANK() **mindig monoton növekvő** számokat ad vissza!

- Az azonos értékű sorok ugyanazt a sorszámot kapják.
- A következő sorszám az aktuálisnál eggyel nagyobb lesz

LAG()

Megadja egy adott sorhoz képest x-sorral korábbi oszlop értékét partíciónként egy adott rendezési szempont szerint

Formája:

LAG(kifejezés, x, default érték)
OVER (PARTITION BY kifejezés
ORDER BY kifejezés)

Példa:
Listázzuk minden rendelési tétel sorszámát, a termék kódját és mennyiségét, valamint az adott termék előző rendelésének mennyiségét!

```
SELECT SORSZAM, TERMEKKOD, MENNYISEG,
        LAG(MENNYISEG,1,0) OVER(PARTITION BY
            TERMEKKOD ORDER BY SORSZAM)
            AS 'Előző rendelési mennyiség'
  FROM Rendeles_tetel
```

A default érték akkor jelenik meg, ha nincs x sorral korábbi elem
Ha x és default érték elmarad, akkor 1 sorral ugrik vissza

LEAD()

Megadja egy adott sorhoz képest x-sorral későbbi oszlop értékét partíciónként egy adott rendezési szempont szerint

Formája:

LEAD(kifejezés, x, default)
OVER (PARTITION BY kifejezés
ORDER BY kifejezés)

Példa:
Listázzuk minden rendelési tétel sorszámát, a termék kódját és mennyiségét, valamint az adott termék kettővel későbbi rendelésének mennyiségét!

```
SELECT SORSZAM, TERMEKKOD, MENNYISEG,
        LEAD(MENNYISEG,2,0) OVER(PARTITION BY
        TERMEKKOD ORDER BY SORSZAM)
    AS 'Két rendeléssel későbbi rendelési mennyiség' FROM
Rendeles_tetel
```

Ha x és default érték elmarad, akkor 1 sort lép előre

FIRST_VALUE()

Megadja egy adott sorrendben lévő csoport (partíció) legelső elemét.

Formája:

FIRST_VALUE(kifejezés)
OVER (ORDER BY kifejezés
PARTITION BY kifejezés)

Példa:
Listázzuk az egyes ügyfelek adatait és első rendelésük dátumát! A lista ne tartalmazzon duplikált sorokat!

```
SELECT DISTINCT u.*,
        FIRST_VALUE(r.REND_DATUM) OVER (Partition BY
        u.LOGIN ORDER BY r.REND_DATUM)
        AS 'Első rendelés' FROM
Ugyfel u
    JOIN Rendeles r ON u.LOGIN = r.LOGIN
```

LAST_VALUE()

Megadja egy adott sorrendben lévő csoport(partíció) legutolsó elemét.*

Formája:

LAST_VALUE(kifejezés)
OVER (ORDER BY kifejezés
PARTITION BY kifejezés)

* A LAST_VALUE esetén vigyázni kell, mivel futtatáskor a partíció legutolsó eleme alapértelmezés szerint az aktuális sor! Megoldás lehet a RANGE vagy helyette fordított sorrend és FIRST_VALUE()

Példa:
Listázzuk az ügyfelek adatai és azt, hogy melyik ügyfél utoljára milyen módon legelőször, illetve legutoljára! A lista ne tartalmazzon duplikált sorokat!

```
SELECT DISTINCT u.*, FIRST_VALUE(r.FIZ_MOD)
            OVER (Partition BY u.LOGIN ORDER BY r.SORSZAM)
             AS 'Fizetési mód legelső rendeléskor', LAST_VALUE(r.FIZ_MOD)
             OVER (Partition BY u.LOGIN ORDER BY r.SORSZAM RANGE
             BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED
             FOLLOWING)
             AS 'Fizetési mód legutolsó rendeléskor' FROM
Ugyfel u
JOIN Rendeles r ON u.LOGIN = r.LOGIN
```

NTILE()

A partíció elemeit adott számú osztályba sorolja a megadott sorrend alapján

Formája:

NTILE(osztályok száma)
OVER (ORDER BY kifejezés
PARTITION BY kifejezés)

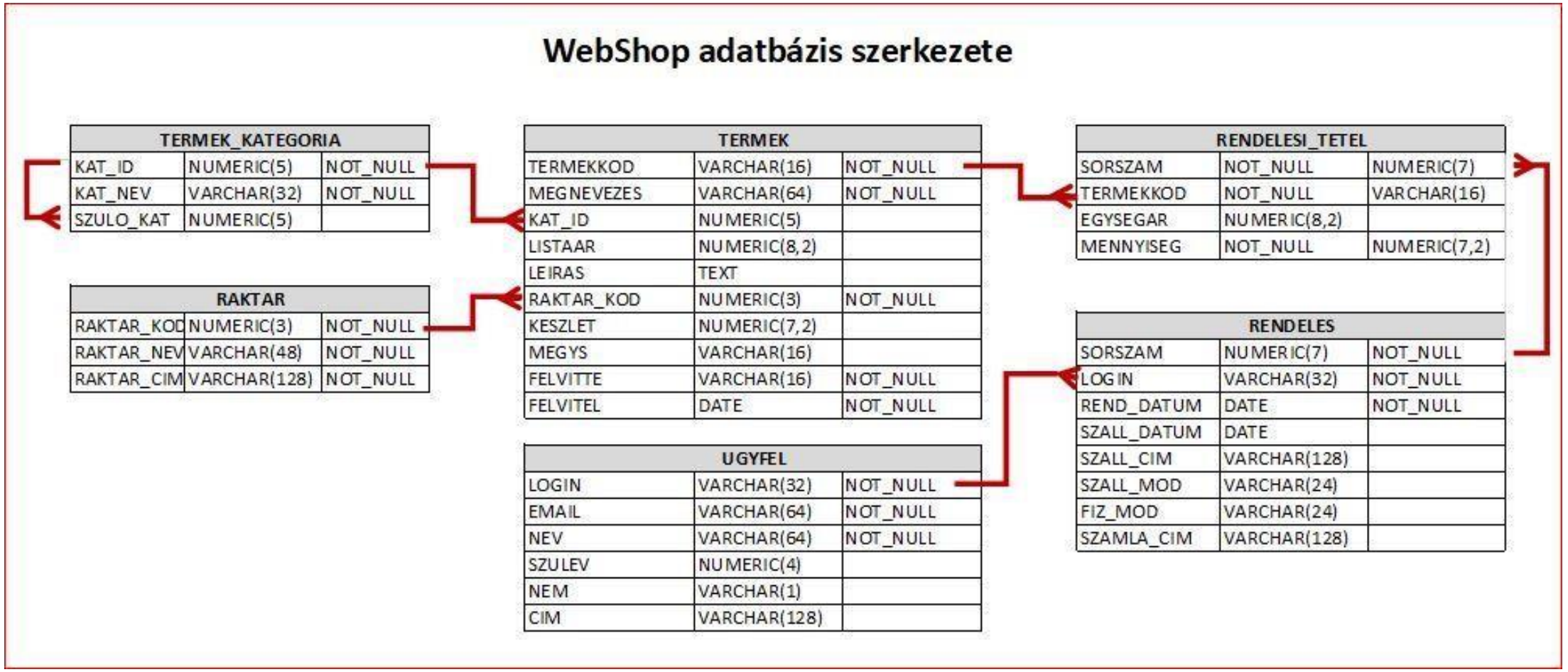
Példa:
Soroljuk be a termékeket kategóriájukban a listaáruk alapján 5 osztályba!

```
SELECT *,
        NTILE(5) OVER(PARTITION BY KAT_ID
        ORDER BY LISTAAR)
        AS 'Osztály'
FROM Termek
```

Megjegyzések

- Az analitikus függvények segítségével sok feladat egyszerűbben megoldható, mint „hagyományos” módon, viszont ilyenkor a lekérdezés többnyire lassúbb lesz
- Bizonyos feladatok a RANGE és a ROWS segítségével is megoldhatók, viszont duplikált sorok esetén a RANGE és a ROWS különböző eredményt adhat
- Egy lekérdezésben több ablak-függvény is szerepelhet
- A ROWS/RANGE esetén a végpont elhagyható, ez esetben alapértelmezés szerint a CURRENT ROW lesz
- Ha a PARTITION BY kimarad, akkor csak egy csoport lesz, amely minden rekordot tartalmaz

A gyakorlaton használt webshop adatbázis



Köszönöm a figyelmet!