ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Korszerű adatbázisok előadás 02

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Miről lesz szó?

- Partíciók, ablakok
  - Az ablakméret megadása fizikailag és logikailag
- Fontosabb analitikus függvények
  - ROW_NUMBER(), RANK(), DENSE_RANK()
  - LAG() és LEAD()
  - FIRST_VALUE(), LAST_VALUE()
  - NTILE()

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Probléma – rossz megoldással

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

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

```
Column 'Termek.TERMEKKOD' is invalid in the select
list because it is not contained in either an
aggregate function or the GROUP BY clause.
```

Ez a megoldás hibás, mert a SELECT után csak olyan oszlopnevek lehetnek, amelyek benne vannak a GROUP BY utáni oszlopok között vagy egy aggregáló függvény belsejében

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Probléma – egy jobb megoldás

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

```
SELECT
  termekkod,
  listaar,
  (SELECT AVG(listaar) FROM termek t2
   WHERE t2.kat_id=termek.kat_id
   GROUP BY kat_id) AS 'Kategória átlagár'
FROM termek
```

Ez a megoldás egy beágyazott SELECT segítségével számítja ki az átlagárakat.

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Probléma – még egy jobb megoldás

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

```
SELECT
  termekkod,
  listaar,
  t2.Kat_atlag_ar
FROM termek JOIN
(
  SELECT kat_id, AVG(listaar) AS 'Kat_atlag_ar'
  FROM termek
  GROUP BY kat_id
) t2 ON termek.kat_id = t2.kat_id
```

Ez a megoldás is beágyazott lekérdezést használ.

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Probléma – legjobb megoldás - **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)

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Partíciók

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

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

TERMEKKOD	LISTAAR	KAT_ID	Kategória átlagár
…	…	…	…
08070483T	324	4	621,133333333333
08070484T	345	4	621,133333333333
08070485T	291	4	621,133333333333
04050035T	24	5	75,3333333333333
04050267T	129	5	75,3333333333333
04110258T	73	5	75,3333333333333
04030285T	11	6	125,4
03050457T	26	6	125,4
03050458T	148	6	125,4
…	…	…	…

Ablakok létrehozása A ROWS és RANGE segítségével

Ablak: a partíció szűkítését jelenti a kezdőpont és a végpont megadásával
ROWS, RANGE: az ablakot határozzák meg, használatukhoz az ORDER BY rész kötelező

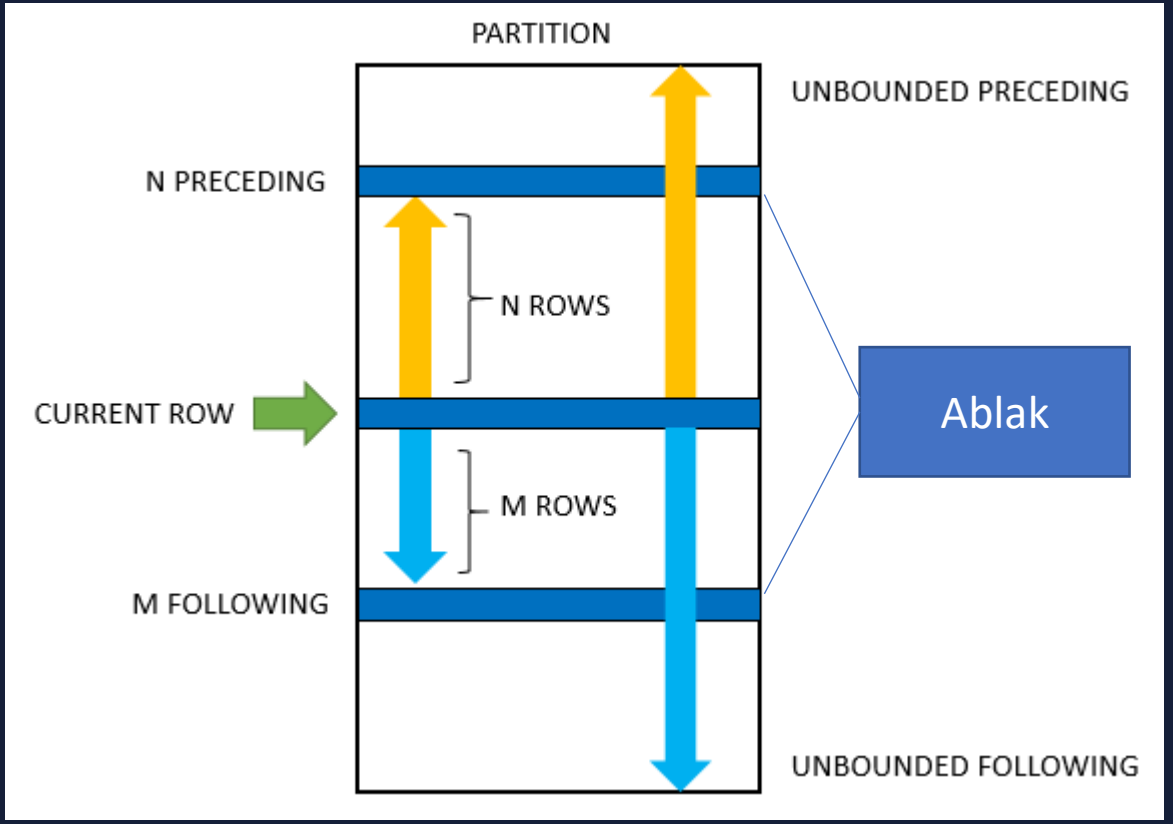
Az ablak definíció formája:

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



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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
```

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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]
```

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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.

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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.

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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
```

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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
```

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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
```

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

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, elhagyása esetén ez az aktuális sor (CURRENT ROW) lesz
- Ha a PARTITION BY kimarad, akkor csak egy csoport lesz, amely minden rekordot tartalmaz

A gyakorlaton használt webshop adatbázis