Abonyi-Tóth Andor:

ADATBÁZISKEZELÉS SQL NYELVEN

(Cikksorozat a www.sulinet.hu portálról. Átszerkesztette és kiegészítette: Szentendrey Péter)

TARTALOM

BEVEZETŐ

Az **SQL** (**S**tructured **Q**uery **L**anguage - Strukturált Lekérdező nyelv) relációs adatbázisok kezelésére alkalmas, szabványos lekérdező nyelv, amelynek fejlesztése az 1970-es években kezdődött az IBM berkein belül, igaz akkor még SEQUEL-nek nevezték. (Structured English QUEry Language) Később más cégek termékeiben is megjelent a lekérdező nyelv (pl. Oracle, Microsoft), 1987-től pedig ANSI szabvány lett, amelyet manapság (néhány módosítással) csaknem minden relációs adatbáziskezelő rendszer alkalmaz.

ISMERKEDÉS AZ SQL-EL

Egy relációs adatbázisban az adattárolás alapjait a táblák jelentik. Egy adatbázis általában több táblából áll. A táblák a logikailag összetartozó adatok *sorokból* és *oszlopokból* álló elrendezése. A táblák sorait *adatrekordoknak*, oszlopait *rekordmezőknek* nevezzük. Az adatbázist alkotó egyedek a táblázat soraiban, az egyedtulajdonságok pedig az oszlopaiban találhatók.

A különböző tábláknak egyedi neve van. (például tanarok, diakok)
(az ékezetes betűk használatát kerüljük!)

Nézzünk egy egyszerű példát. Az alábbi tábla neve legyen: ***diakok***

nev	**szuletesi_hely**	**szuletesi_ev**
Kiss Irma	Budapest	1991
Horváth Béla	Kisvárda	1990
Jeney Éva	Karcag	1988
Turán Lajos	Abony	1990

A fenti táblázat 4 rekordot (személyenként egyet) tartalmaz, amelynek 3 mezője van. (Név, születési hely, születési év)

Hogyan tudjuk lekérdezni SQL nyelven a *diakok* táblában található neveket?

```
SELECT nev FROM diakok;
```

A lekérdezés eredménye a következő lenne:

nev
Kiss Irma
Horváth Béla
Jeney Éva
Turán Lajos

AZ SQL UTASÍTÁSOK FAJTÁI

Az SQL utasításokat csoportosíthatjuk a céljuk szerint. Mi az alábbi kategóriákba tartozó, leggyakrabban használt utasításokat tekintjük át.

- **Lekérdező utasítások** (DQL, vagyis DATA QUERY LANGUAGE)
  - az adatbázisból történő lekérdezéseket teszi lehetővé
- **Adatleíró utasítások** (DDL, vagyis Data Definition Language)
  - ezekkel tudunk például különböző táblákat (és más objektumokat) létrehozni, módosítani, törölni...
- **Adatmódosító utasítások** (DML, vagyis Data Manipulation Language)
  - ezek szolgálnak az adatok beírására, módosítására és törlésére

A SELECT UTASÍTÁS

Az előzőekben már használt SELECT utasítással a táblázatból szelektálhatunk (válogathatunk) ki adatokat, eredményül egy ún. *eredménytáblát* kapunk. Az eredménytábla hasonlít az adattáblákhoz, de csak ideiglenesen jön létre a lekérdezés futtatásakor és nem tárolódik az adatbázisban. Az eredménytáblát a SELECT utasítás ún. záradékaival szűrhetjük, rendezhetjük, csoportosíthatjuk, vagy akár újabb lekérdezést is végezhetünk rajta (beágyazott-, avagy allekérdezések).

Használata:

SELECT *oszlop_nevek* FROM *tábla_neve*;

Megjegyzés: az SQL nyelv parancsaiban a kis- és nagybetűk nincsenek megkülönböztetve. A jobb átláthatóság érdekében mi csupa nagy betűvel írjuk a nyelv alapszavait.

Az oszlopneveket vesszővel kell elválasztani. A fenti példánál maradva:

```
SELECT nev,szuletesi_hely FROM diakok;
```

Az eredmény:

nev	**szuletesi_hely**
Kiss Irma	Budapest
Horváth Béla	Kisvárda
Jeney Éva	Karcag
Turán Lajos	Abony

Ha az összes mező értékét le szeretnénk kérdezni, akkor nem szükséges felsorolni az összes oszlop nevét, használhatjuk a * karaktert is.

```
SELECT * FROM diakok;
```

A fenti lekérdezéssel az összes rekord tartalmát le tudjuk kérdezni. Az eredmény:

nev	**szuletesi_hely**	**szuletesi_ev**
Kiss Irma	Budapest	1991
Horváth Béla	Kisvárda	1990
Jeney Éva	Karcag	1988
Turán Lajos	Abony	1990

Az itt látható eredményhalmazban történő navigációhoz a különböző rendszerek felkínálnak olyan lehetőségeket, amelyekkel például az első, előző, következő, vagy utolsó rekordra "állhatunk", illetve kinyerhetjük az egyes mezők tartalmát.

A DISTINCT OPCIÓ

Mi a helyzet akkor, ha nekünk csak arról kell kimutatás, hogy a diákok mely évben születtek, és az mellékes, hogy egy adott évben több diák is született. Erre is van lehetőség, de ekkor a

SELECT DISTINCT *oszlop_nevek* FROM *tábla_neve*;

lekérdezést kell használnunk. A mi példánkban:

```
SELECT DISTINCT szuletesi_ev FROM
diakok;
```

Az eredmény:

szuletesi_ev
1991

1990
1988

Látható, hogy az eredményhalmaz minden születési évet tartalmaz, de csak egyszer.

A DISTINCT hatására a rendszer az eredménytáblából elhagyja azokat a rekordokat, amelyek többszörös adatokat tartalmaznak a kiválasztott mezőkben. Ahhoz, hogy a lekérdezés eredményében szerepeljenek, a SELECT utasításban felsorolt mezőkben lévő értékeknek egyedieknek kell lenniük. Például az Employees tábla több azonos vezetéknevű személy adatait is tartalmazhatja. Ha két olyan rekord is van, amelynél a LastName (vezetéknév) mező tartalma Kovács, az alábbi SQL-utasítás ezek közül csak egyet fog visszaadni:

```
SELECT DISTINCT
LastName
FROM Employees;
```

Ha a DISTINCT kijelentést elhagyjuk, a lekérdezés mindkét Kovács vezetéknevet tartalmazó rekordot visszaadja.

Ha a SELECT DISTINCT utasítás egynél több mezőt tartalmaz, akkor az összes mezőben található értékek *kombinációjának* kell egyedinek lennie egy rekord esetében ahhoz, hogy a rekord szerepeljen az eredményben.

LEKÉRDEZÉS EREDMÉNYÉNEK SZŰKÍTÉSE (WHERE)

Jó lenne, ha tovább tudnánk szűkíteni a lekérdezések eredményét. Természetesen van rá lehetőség, de ehhez egy további feltétellel kell kiegészíteni a lekérdezést, amelyet a WHERE kulcsszó után kell írni.

SELECT *oszlop_nevek* FROM *tábla_neve* WHERE *oszlopra vonatkozó feltétel;*

Hogy világos legyen, nézzünk meg egy példát.

```
SELECT nev FROM diakok WHERE szuletesi_ev=1990;
```

A fenti lekérdezés eredményeként az 1990-ben született diákok nevét kapjuk meg.

nev
Horváth Béla
Turán Lajos

Nézzünk egy másik példát!

```
SELECT nev FROM diakok WHERE
szuletesi_hely='Abony';
```

Az eredmény:

nev
Turán Lajos

Észrevetted a két lekérdezés közti különbséget? Amikor szöveget írunk a feltételbe, azt aposztróf jelek (') közé kell tenni. Némelyik rendszer (Pl. Access) azt is elfogadja, ha idézőjelet (") használunk.

Persze az egyenlőségjel csak egy lehetséges műveleti jel, az alábbiakat is használhatjuk.

MŰVELETI JELEK A WHERE FELTÉTELBEN

Művelet	**Leírás**
=	**egyenlő**
Példa	SELECT nev FROM diakok WHERE szuletesi_hely='Abony';
Eredmény	**nev** Turán Lajos
<>	**nem egyenlő**
Példa	SELECT nev FROM diakok WHERE szuletesi_hely<>'Abony';
Eredmény	**nev** Kiss Irma Horváth Béla Jeney Éva
<	**kisebb, mint ....**
Példa	SELECT nev FROM diakok WHERE szuletesi_ev<1990;
Eredmény	**nev** Kiss Irma
>	**nagyobb, mint ...**
Példa	SELECT nev FROM diakok WHERE szuletesi_ev>1990;
Eredmény	**nev** Jeney Éva
<=	**kisebb vagy egyenlő, mint ...**
Példa	SELECT nev FROM diakok WHERE szuletesi_ev<=1990;
Eredmény	**nev** Kiss Irma Horváth Béla Turán Lajos
>=	**nagyobb vagy egyenlő, mint ...**
Példa	SELECT nev FROM diakok WHERE szuletesi_ev>=1990;
Eredmény	**nev** Horváth Béla

	Turán Lajos
LIKE	**egy keresési minta alapján keres** A minta megadásánál használhatod a % karaktert, amely több karakter helyettesítésére szolgál. Egy karakter helyettesítésére a _ mintát használhatod. Példa: 'K%' : K betűvel kezdődő szöveg '%A' : A betűvel végződő szöveg 'A%U' : A betűvel kezdődő, U betűvel végződő szöveg '%MA%' : minden olyan szöveg, amely tartalmazza a MA karaktereket. 'K_A' : K betűvel kezdődő és A-val végződő, 3 karakterből álló szöveg
Példa	SELECT nev FROM diakok WHERE szuletesi_hely LIKE 'K%';
Eredmény	**nev** Horváth Béla Jeney Éva
BETWEEN	**egy keresési intervallumot adhatunk meg**
Példa	SELECT nev FROM diakok WHERE szuletesi_ev BETWEEN 1988 and 1990;
Eredmény (a mySQL környezetben)	**nev** Horváth Béla Jeney Éva Turán Lajos
	Vigyázz! Ezt a funkciót az egyes adatbáziskezelő rendszerek eltérően valósíthatják meg. Van amelyik a két megadott értéket is megjeleníti az eredményhalmazban, de van amelyik csak a köztük lévő értéket, sőt más kombináció is lehet. Feltétlenül meg kell bizonyosodnod erről az általad használt rendszerben.

FIGYELEM!

Az Access adatbáziskezelőben a Windows hagyományoknak megfelelően a ? és a * helyettesítő karaktereket használhatjuk!

Helyettesítő-karakter	Funkciója	Példa
*	Tetszőleges számú karaktert helyettesít, és a karakterláncban bárhol használható.	LIKE "Szabó*"
?	Egyetlen tetszőleges karaktert helyettesít.	LIKE "sz?l"
#	Egyetlen tetszőleges számjegyet helyettesít.	LIKE "#. emelet"
[karakterek]	A szögletes zárójelek között levő karakterek közül bármelyiket helyettesíti.	LIKE "sz[ééó]l"
[!karakterek]	Bármely, a szögletes zárójelek között nem szereplő karakter helyettesít.	LIKE "f[!aü]l"
[karakter1-karakter2]	A karakter1-től karakter2-ig terjedő tartományban levő bármely karaktert helyettesít.	LIKE "[1-5]. emelet"
[*], [?], [#], [[]	A * csillag, ? kérdőjel, # kettőskereszt, illetve [nyitó szögletes zárójel karakterek jelölésére szolgál. Abban az esetben használjuk, ha a LIKE operátorral megadott mintában magát a csillag, kérdőjel, kettőskereszt vagy nyitó szögletes zárójelet szeretnénk keresni.	LIKE "*mikor[?]"

A következő táblázat összefoglalja, hogy a **Like** művelettel miként vizsgálhatunk különböző mintákra vonatkozó kifejezéseket:

Helyettesítendő	**Minta**	**Megfelel (True értéket ad vissza)**	**Nem felel meg (False értéket ad vissza)**
Több karakter	a*a	aa, aBa, aBBBa	aBC
	ab	abc, AABB, Xab	aZb, bac
Speciális karakter	a[*]a	a*a	aaa
Több karakter	ab*	abcdefg, abc	cab, aab
Egyetlen karakter	a?a	aaa, a3a, aBa	aBBBa
Egyetlen számjegy	a#a	a0a, a1a, a2a	aaa, a10a
Karaktertartomány	[a-z]	f, p, j	2, &
Tartományon kívüli	[!a-z]	9, &, %	b, a
Nem számjegy	[!0-9]	A, a, &, ~	0, 1, 9
Kombinált	a[!b-m]#	An9, az0, a99	abc, aj0

Azt, hogy egy rekordban egy bizonyos mezőnek nincs értéke (üres) a következőképp fogalmazhatjuk meg a WHERE záradékban:

```
SELECT mezonev
FROM tablanev
WHERE mezonev IS NULL ;
```

Azt, hogy egy mező nem üres, az IS NOT NULL használatával fejezhetjük ki.

AZ EREDMÉNYHALMAZ RENDEZÉSE

Gyakorta előfordul, hogy az eredményhalmazt valamelyik mező alapján rendezni szeretnénk. Például az iskolaigazgatónak szüksége van a diákok nevére, amelyet ábécé sorrendben szeretne megkapni.

Hogy tudjuk ezt megvalósítani? Az **ORDER BY** kulcsszó után meg kell adnunk, hogy mely mező (vagy mely mezők) szerint történjen a sorrendbe állítás.

SELECT *oszlop_nevek* FROM *tábla_neve* ORDER BY *oszlop_neve1, ..., oszlop_neven;*

Példa: Kérdezzük le a névsort, név szerint rendezve!

```
SELECT * FROM diakok ORDER BY nev;
```

Az eredmény:

nev	**szuletesi_hely**	**szuletesi_ev**
Horváth Béla	Budapest	1991
Horváth Béla	Kisvárda	1990
Horváth Csilla	Kaposvár	1989
Jeney Éva	Karcag	1988
Kassai Tünde	Nagykanizsa	1991
Turán Lajos	Abony	1990

Ha azt szeretnénk, hogy amellett, hogy a nevek ábécé sorrendben jelenjenek meg, még a születési dátum szerint is rendezve legyen az eredményhalmaz, több mezőt is fel kell sorolnunk. (vigyázzunk a mezők sorrendjére!)

```
SELECT * FROM diakok ORDER BY nev,szuletesi_ev;
```

Az eredmény:

nev	**szuletesi_hely**	**szuletesi_ev**
Horváth Béla	Kisvárda	1990
Horváth Béla	Budapest	1991
Horváth Csilla	Kaposvár	1989
Jeney Éva	Karcag	1988
Kassai Tünde	Nagykanizsa	1991
Turán Lajos	Abony	1990

Látjuk, hogy az eredményhalmaz első két sora felcserélődött, hiszen a Budapesten született Horváth Béla nevű tanuló korábban született, mint az ugyanilyen nevű, Kisvárdán született társa.

De mi a helyzet, ha fordított sorrendben szeretnénk megkapni az adatokat? Erre is van lehetőség:

SELECT *oszlop_nevek* FROM *tábla_neve* ORDER BY *oszlop_neve1* ASC|DESC, ..., *oszlop_neven* ASC|DESC;

Az **ASC** kulcsszót kell szerepeltetnünk, ha növekvő, illetve a **DESC** kulcsszót ha csökkenő sorrendben szeretnénk az eredményt megkapni.

Példa:

Az igazgatónak olyan listára van szüksége, amely a születési dátumok szerint csökkenő, a nevek szerint viszont növekvő sorrendbe van rendezve.

```
SELECT * FROM diakok ORDER BY szuletesi_ev DESC,
nev ASC;
```

Az eredmény:

nev	**szuletesi_hely**	**szuletesi_ev**
Horváth Béla	Budapest	1991
Kassai Tünde	Nagykanizsa	1991
Horváth Béla	Kisvárda	1990
Turán Lajos	Abony	1990
Horváth Csilla	Kaposvár	1989
Jeney Éva	Karcag	1988

A TOP OPCIÓ

TOP n [PERCENT] Az ORDER BY záradék segítségével felállított sorrend elején vagy végén található, megadott tartományba eső rekordokat adja vissza. Tegyük fel, hogy az 1994-es évfolyam első 25 tanulójának nevét szeretnénk megkapni:

```
SELECT TOP 25
FirstName, LastName
FROM Students
WHERE GraduationYear = 1994
ORDER BY GradePointAverage DESC;
```

Ha az ORDER BY záradékot elhagyjuk, a lekérdezés 25 tetszőleges, a WHERE záradék feltételeit kielégítő rekordot ad vissza a Students (tanulók) táblából.

A TOP kijelentés nem választ az egyenlő értékek közül. Az előbbi példában, ha a 25. és 26. legmagasabb átlagos vizsgapontszám egyenlő, a lekérdezés 26 rekordot fog visszaadni.

A PERCENT fenntartott szó segítségével a rekordok bizonyos százalékát is lekérdezhetjük az ORDER BY záradék által felállított sor elejéről vagy végéről. Tegyük fel, hogy az első 25 tanuló helyett az osztály utolsó 10 százalékára vagyunk kíváncsiak:

```
SELECT TOP 10 PERCENT
FirstName, LastName
FROM Students
WHERE GraduationYear = 1994
ORDER BY GradePointAverage ASC;
```

A WHERE FELTÉTELEK ÖSSZEKAPCSOLÁSA

Emlékezzünk, hogy a **WHERE** kulcsszó után megadhattunk egy feltételt, amely alapján szűrtük az eredményhalmazt.

```
SELECT nev FROM diakok WHERE szuletesi_ev=1990;
```

Azonban ezen feltételeket össze is tudjuk kapcsolni különböző műveletekkel. Például AND (és), illetve OR (vagy).

AZ OR (VAGY) ÉS AND (ÉS) MŰVELETEK

A műveletek használati módja:

SELECT ***oszlop_nevek*** **FROM** ***tábla_neve*** **WHERE** ***oszlop_neve műveleti_jel érték*** **AND** ***oszlop_neve műveleti_jel érték;***

SELECT ***oszlop_nevek*** **FROM** ***tábla_neve*** **WHERE** ***oszlop_neve műveleti_jel érték*** **OR** ***oszlop_neve műveleti_jel érték*****;**

Példa (AND művelet)

Készítsünk egy listát azon diákokról, akiknek a családi neve Horváth és 1990-ban, vagy később születtek.

```
SELECT * FROM diakok
WHERE nev LIKE 'Horváth%' AND szuletesi_ev>=1990;
```

Az eredmény:

nev	**szuletesi_hely**	**szuletesi_ev**
Horváth Csilla	Kaposvár	1989
Horváth Béla	Kisvárda	1990

Példa (OR művelet)

Készítsünk egy listát azon diákokról, akik 1991-ben vagy 1990-ban születtek!

```
SELECT * FROM diakok
WHERE szuletesi_ev=1991 OR szuletesi_ev=1990;
```

Az eredmény:

nev	**szuletesi_hely**	**szuletesi_ev**
Horváth Béla	Kisvárda	1990
Turán Lajos	Abony	1990
Horváth Béla	Budapest	1991
Kassai Tünde	Nagykanizsa	1991

AZ IN MŰVELET

Ha az alapján szeretnénk szűrni, hogy a mező értéke egy adott felsoroláshalmazba tartozik-e, használhatjuk az **IN** műveletet.

SELECT ***oszlop_nevek*** **FROM** ***tábla_neve*** **WHERE** ***oszlop_neve*** **IN (érték1, érték2, ...) ;**

Példa:

Azt szeretnénk megtudni, hogy mely tanulók születtek Karcagon, illetve Kisvárdán. Ehhez a következő lekérdezésre van szükség:

```
SELECT * FROM diakok
WHERE szuletesi_hely IN ('Karcag', 'Kisvárda');
```

Az eredmény:

nev	szuletesi_hely	szuletesi_ev
Jeney Éva	Karcag	1988
Horváth Béla	Kisvárda	1990

BETWEEN ... AND MŰVELET

Ha egy intervallum alapján akarjuk szűkíteni a feltételt, használhatjuk a **BETWEEN ... AND** műveleteket is:

SELECT *oszlop_nevek* FROM *tábla_neve* WHERE *oszlop_neve* BETWEEN érték1 AND érték2 ;

Készítsünk egy listát azon diákokról, akik 1988 és 1990 között születtek.

```
SELECT * FROM diakok
WHERE szuletesi_ev BETWEEN 1988 AND 1990;
```

Az eredmény:

nev	szuletesi_hely	szuletesi_ev
Horváth Béla	Kisvárda	1990
Horváth Csilla	Kaposvár	1989
Jeney Éva	Karcag	1988
Turán Lajos	Abony	1990

Ez a művelet azonban nem csak számok esetén működik, akár kilistázhatjuk azon tanulók nevét is, akik ábécésorrendben Jeney Éva és Turán Lajos között helyezkednek el.

```
SELECT * FROM diakok
WHERE nev BETWEEN 'Jeney Éva' AND 'Turán Lajos';
```

nev	szuletesi_hely	szuletesi_ev
Jeney Éva	Karcag	1988
Turán Lajos	Abony	1990
Kassai Tünde	Nagykanizsa	1991

(Vigyázzunk arra, hogy ez a funkció más-más eredményt adhat az elérő adatbáziskezelő rendszerekben. Van ahol csak a két érték között található eredmények lesznek részei az eredményhalmaznak, a határértékek nem,

ezért ezen funkció működését nagyon fontos letesztelni az adott rendszerben! Az Access adatbáziskezelő a határértékeket is hozzáveszi az eredményhalmazhoz, tehát a BETWEEN 1 AND 20 feltétel egyenértékű a >=1 AND <= 20 feltétellel.)

TAGADÁS (NOT)

Amennyiben pont arra a halmazra van szükségünk, amelyet a feltétel tagadásával kapnánk, használhatjuk a NOT szócskát is.

SELECT *oszlop_nevek* FROM *tábla_neve* WHERE *oszlop_neve* NOT BETWEEN *érték1* AND *érték2* ;

Készítsünk egy listát azon diákokról, akik NEM 1988 és 1990 között születtek.

```
SELECT * FROM diakok
WHERE szuletesi_ev NOT BETWEEN 1988 AND 1990;
```

Az eredmény:

nev	szuletesi_hely	szuletesi_ev
Horváth Béla	Budapest	1991
Kassai Tünde	Nagykanizsa	1991

AZ ALIAS HASZNÁLATA

Ha az eredménytáblában nem az eredeti oszlopneveket szeretnénk viszontlátni, használhatunk helyettük álneveket, vagyis aliasokat is.

SELECT *oszlop_neve1* AS oszlop_alias1, *oszlop_neve2* AS oszlop_alias2 FROM *tábla_neve;*

Példa:

A táblázatunk a következő:

nev	szuletesi_hely	szuletesi_ev
Jeney Éva	Karcag	1988
Horváth Csilla	Kaposvár	1989
Horváth Béla	Kisvárda	1990
Turán Lajos	Abony	1990
Horváth Béla	Budapest	1991
Kassai Tünde	Nagykanizsa	1991

Mit tehetünk akkor, ha olyan eredményhalmazra van szükségünk amely csak a nev, és szuletesi_ev oszlopokat tartalmazza, és az oszlopok neve angolul szerepel?

```
SELECT nev AS Name, szuletesi_ev AS Birthdate FROM diakok;
```

Az eredmény:

Name	Birthdate

Jeney Éva	1988
Horváth Csilla	1989
Horváth Béla	1990
Turán Lajos	1990
Horváth Béla	1991
Kassai Tünde	1991

SZÁMÍTOTT MEZŐK LÉTREHOZÁSA

Az adatbázisokat célszerű úgy megtervezni, hogy ne tároljunk bennük olyan adatokat, amelyek kiszámíthatók a többi tárolt adatból. (Pl. ha tároljuk árucikkek nettó árát és az ÁFA kulcsot, akkor felesleges tárolni a bruttó árat.) Az ilyen, számított értékeket a lekérdezésekben a következőképpen jeleníthetjük meg:

```
SELECT Netto_Ar * AFA / 100 AS Bruttó_Ar FROM Termekek ;
SELECT Date ( ) + 10 AS Tiz_Nap_Mulva FROM …
SELECT Vezeteknev & ’  ’ & Keresztnev AS Teljes_Nev FROM Szemelyek ;
```

A számításokban alkalmazhatjuk a szokásos matematikai műveleteket, valamint az adatbáziskezelő rendszer által biztosított különféle operátorokat és függvényeket. (Lásd az adatbáziskezelő dokumentációját / súgóját.)

ÖSSZESÍTŐ (AGGREGÁLÓ) FÜGGVÉNYEK

A lekérdezés eredményeként előálló táblák egyes oszlopaiban lévő értékeken végrehajthatunk bizonyos összesítő műveleteket, amelyek egyetlen értéket állítanak elő.

AVG()..... átlag
SUM()..... összeg
COUNT()..... darabszám
MAX()..... maximális érték
MIN()..... minimális érték
FIRST() / LAST()..... egy lekérdezés eredményhalmazának első vagy utolsó rekordjából ad vissza mezőértéket.

A Null érték hatása numerikus számításokra

Ha összesítő függvény segítségével számítunk összeget, átlagot, darabszámot vagy más mennyiséget mező értékein, az abban a mezőben Null értékeket tartalmazó rekordok nem számítanak bele az eredménybe. (Ez akkor is igaz, ha az összesítést a lekérdezés tervezőrácsának Összesítés sora, az Egyszerű lekérdezés Varázsló vagy egyéni kifejezés segítségével számítjuk.) Ha például a Count függvénnyel számoljuk meg a mező értékeinek számát, ez a nem Null értékű rekordok számát adja vissza. Ha a Null értéket tartalmazókat is be szeretnénk venni az eredménybe, a Count függvényt csillag (*) helyettesítő karakterrel kell használni.

Ha műveleti jel (például +, -, *, /) is szerepel a kifejezésben (például [Raktáron]+[Megrendelve]), és a kifejezés mezőinek egyike Null értéket tartalmaz, az egész kifejezés eredménye Null érték lesz.

Példák:

Az üzletkötők átlagfizetése: SELECT AVG (sal) FROM amp WHERE job = ’SALESMAN’;

Hány dolgozó van: SELECT COUNT(*) FROM emp;

Hány különböző beosztás van: SELECT COUNT(DISTINCT job) FROM emp;

Mivel az oszlopfüggvény eredménye egyetlen értéket állít elő, az oszlopfüggvény mellé vagy más oszlopfüggvényeket írhatunk, vagy olyan értéket írhatunk, amelyik az összes kiválasztott sorban azonos.

SELECT job, AVG(sal) FROM emp WHERE job =’SELESMAN’;

SELECT COUNT (*), AVG(sal) FROM emp;

De hibás a következő:

SELECT COUNT (*), ename FROM emp;

CSOPORTKÉPZÉS – A GROUP BY ZÁRADÉK

A megadott mezőlista azonos értékű rekordjait egyetlen rekordcsoporttá alakítja. Ha SQL összesítő függvényt, például Sum vagy Count függvényt adunk meg a SELECT utasításban, akkor minden rekordcsoporthoz létrejön összegérték.

Szintaxis

```
SELECT mezőlista
FROM tábla
WHERE feltétel
[GROUP BY mezőcsoportlista]
```

Egy GROUP BY záradékot tartalmazó SELECT utasítás a következő részekből áll:

Rész	Leírás
Mezőlista	A megjelenítendő mező vagy mezők neve az alias nevükkel együtt, az SQL összesítő függvények, a kiválasztó kijelentések (ALL, DISTINCT, TOP) vagy a SELECT utasítás egyéb beállításai.
Tábla	A rekordok keresésekor használni kívánt tábla neve.
Feltétel	A kiválasztás feltételei. **Ha az utasítás WHERE záradékot tartalmaz, az adatbázismotor csak azt követően csoportosítja az értékeket, hogy a WHERE záradékban megadott feltételeket már alkalmazta a rekordokra!**
Mezőcsoportlista	Legfeljebb 10 mező neve, amelyekkel a rekordokat csoportosítjuk. A mezőcsoportlistában megadott sorrend határozza meg a csoportosítási szinteket, a legmagasabbtól a legalacsonyabb szintig.

A GROUP BY mezők Null értékei nem maradnak ki a csoportosításból. Az SQL összesítő függvények azonban nem veszik figyelembe a Null értékeket.

A csoportosításból kizárni kívánt sorokat a WHERE záradékkal határozhatjuk meg, csoportosítás *után* pedig a HAVING záradékkal szűrhetjük a rekordokat.

A SELECT mezőlista minden mezőjének szerepelnie kell vagy a GROUP BY záradékban, vagy az SQL összesítő függvény argumentumai között!

Ha például a tanulók nevét, osztálykódját és életkorát is tartalmazó TANULOK táblából ki szeretnénk íratni az egyes osztályok átlagéletkorát, akkor ezt a következő utasítással érhetjük el:

```
SELECT osztalykod, Avg (eletkor) AS átlagéletkor
FROM tanulok
GROUP BY osztalykod ;
```

A HAVING ZÁRADÉK

Megadja, hogy mely csoportosított rekordok jelennek meg egy GROUP BY záradékot tartalmazó SELECT utasítás végrehajtásakor. *Miután* a GROUP BY csoportosította a rekordokat, a HAVING megjeleníti a GROUP BY záradékkal csoportosított összes olyan rekordot, amely eleget tesz a HAVING záradék feltételeinek.

Szintaxis

```
SELECT mezőlista
FROM tábla
WHERE feltétel
GROUP BY csoportmezőlista
[HAVING csoportokra vonatkozó feltétel]
```

A HAVING záradékot tartalmazó SELECT utasítás részei:

Rész	Leírás
Mezőlista	A visszakeresendő mező vagy mezők neve az alias nevükkel együtt, az SQL összesítő függvények, a kiválasztó kijelentések (ALL, DISTINCT, TOP) vagy a SELECT utasítás egyéb beállításai.
Tábla	A rekordok lekérdezéséhez használni kívánt tábla neve.
Feltétel	A kiválasztási feltétel. Ha az utasítás WHERE záradékot tartalmaz, az adatbázismotor csak azt követően csoportosítja az értékeket, hogy a WHERE záradékban megadott feltételeket alkalmazta a rekordokra.
Mezőcsoportlista	Legfeljebb 10 mező neve, amelyekkel a rekordokat csoportosítjuk. A mezőcsoportlistában megadott sorrend határozza meg a csoportosítási szinteket, a legmagasabbtól a legalacsonyabb szintig.
csoportosítási feltétel	Kifejezés, amely meghatározza, hogy a csoportosított rekordok közül melyek kerüljenek megjelenítésre.

A HAVING hasonló a WHERE záradékhoz, de a WHERE a csoportképzés előtt, az egyes rekordokat szűri, a HAVING pedig a csoportképzés után, a csoportokra érvényesít szűrőfeltételt. Miután a rekordokat a GROUP BY záradékkal csoportosítottuk, a HAVING záradékkal megadhatjuk, hogy mely rekordok jelenjenek meg. Az alábbi példa azokat az osztályokat jeleníti meg, amelyek átlagéletkora nagyobb mint 18:

```
SELECT osztalykod, Avg (eletkor) AS átlagéletkor
FROM tanulok
GROUP BY osztalykod
HAVING Avg (eletkor) > 18 ;
```

Egy HAVING záradék legfeljebb 40 kifejezést tartalmazhat, amelyek logikai operátorokkal, például az And vagy az Or operátorral kapcsolhatók egymáshoz.

TÁBLÁK ÖSSZEKAPCSOLÁSA (JOIN)

Az adatbáziskezelés leggyakrabban használt műveletei között mindenképpen meg kell említeni az összekapcsolás műveletét is. E művelet fontosságának egyik legfőbb oka az, hogy az adatbázis tervezése során, a normalizálással az információkat több táblára bontjuk szét. Egy összetettebb lekérdezéshez szükséges információk több táblában szétszórva helyezkednek el, így a lekérdezés során össze kell gyűjteni ezen adatokat a különböző táblákból, ahol az összetartozás bizonyos mezők értékeinek kapcsolatán alapszik. Azt a folyamatot, amikor több táblából származó adatokból állítunk elő egy újabb eredménytáblázatot, összekapcsolásnak, egyesítésnek vagy join-nak nevezzük.

Az SQL-ben két táblázat egyesítésének legegyszerűbb formája, amikor a két táblázat Descartes szorzatát képezzük, mely során az eredménytáblázat egy rekordja úgy áll elő, hogy az egyik táblázat rekordjaihoz hozzáfűzzük a másik táblázat egy-egy rekordját, ahol az eredménytáblázat *minden lehetséges párosítást tartalmaz*. (Ha tehát az egyik táblázat 3 rekordot tartalmaz, a másik 4-et, akkor az eredmény 12 rekordból áll.)

Két táblázat Descartes szorzatának előállításához a következő SQL utasítást kell kiadni:

SELECT * FROM táblázatnév1, táblázatnév2;

Az így előálló egyesítéstáblázatot ezután tetszőlegesen tovább lehet alakítani a már megismert záradékokkal. Erre rendszerint szükség is van, hiszen csak nagyon ritkán van szükség két táblázat rekordjainak teljes Descartes szorzatára, legtöbbször csak a Descartes szorzat bizonyos részhalmazára van szükségünk. A szorzat táblázat szelekciójával valósítható meg például a táblázatok összekapcsolására szolgáló kulcs és kapcsolókulcs szerkezet alapján előálló rekord párok kijelzése.

SELECT tábla_neve1.oszlop_neve, tábla_neve2.oszlop_neve
FROM tábla_neve1, tábla_neve2
WHERE feltétel;

Példa: Egy listát szeretnénk készíteni arról, hogy melyik diák melyik osztályba jár és ki az osztályfőnöke.

Megoldás:

```
SELECT diakok.nev, diakok.osztaly, tanarok.nev
FROM diakok, tanarok
WHERE diakok.osztaly=tanarok.osztalyfonok;
```

Az eredmény:

nev	**osztaly**	**nev**
Horváth Csilla	11.A	Kovács Lajos
Horváth Béla	10.C	Kis Tamás
Turán Lajos	10.C	Kis Tamás
Horváth Béla	9.A	Szép Béla
Jeney Éva	12.B	Nyers Jolán

Persze jobb lenne, ha az eredményhalmazban nem lenne két név oszlop. Sebaj, a korábban ismertetett ALIAS használatával orvosolhatjuk a problémát.

```
SELECT diakok.nev AS diak, diakok.osztaly, tanarok.nev as osztalyfonok
FROM diakok, tanarok
WHERE diakok.osztaly=tanarok.osztalyfonok;
```

Az eredmény:

diak	**osztaly**	**osztalyfonok**
Horváth Csilla	11.A	Kovács Lajos
Horváth Béla	10.C	Kis Tamás

Turán Lajos	10.C	Kis Tamás
Horváth Béla	9.A	Szép Béla
Jeney Éva	12.B	Nyers Jolán

A fenti módszer lényege, hogy a táblák közötti kapcsolatot a WHERE záradékban adjuk meg. (A kapcsolómezők egyenlőségét szabva feltételül.) Van ennek egy másik módja is: használhatjuk a **JOIN** kulcsszót.

INNER JOIN

Az INNER JOIN használatával, a lekérdezés eredményébe nem kerülnek bele azon *tábla1*-beli elemek, amelyeknek nincs megfelelőjük a *tábla2* táblában.

SELECT *oszlop_neve1, oszlop_neve2, oszlop_neve3*
FROM *tábla_neve1* INNER JOIN *tábla_neve2* ON *tábla_neve1.mező_neve = tábla_neve2.mező_neve;*

Példa:

```
SELECT diakok.nev AS diak, diakok.osztaly, tanarok.nev as osztalyfonok
FROM diakok INNER JOIN tanarok ON diakok.osztaly=tanarok.osztalyfonok;
```

Az eredmény ugyanaz lesz, mint az eggyel korábbi esetben. Az INNER JOIN használata annyiban jobb a táblák WHERE záradékon keresztül történő kapcsolásánál, hogy így külön helyen szerepelnek a kapcsolatokat leíró feltételek és a lekérdezés eredményét szűkítő feltételek, és ezáltal a lekérdezés SQL kódja áttekinthetőbb lesz.

LEFT JOIN

Arra is van lehetőségünk, hogy az első tábla minden adatát megjelenítsük az eredményben, attól függetlenül, hogy nincs a feltételben megadott tulajdonságú mező a második táblában. Ekkor a LEFT JOIN kulcsszót kell használni.

```
SELECT diakok.nev AS diak, diakok.osztaly, tanarok.nev as osztalyfonok
FROM diakok LEFT JOIN tanarok ON diakok.osztaly=tanarok.osztalyfonok;
```

Az eredmény:

diak	**osztaly**	**osztalyfonok**
Horváth Béla	10.C	Kis Tamás
Jeney Éva	12.B	Nyers Jolán
Turán Lajos	10.C	Kis Tamás
Horváth Csilla	11.A	Kovács Lajos
Kassai Tünde	9.B	NULL
Horváth Béla	9.A	Szép Béla

RIGHT JOIN

A RIGHT JOIN pedig pont azt teszi lehetővé, hogy a második táblában lévő összes adatot jelenítsük meg, függetlenül attól, hogy az első táblában van-e hozzátartozó mező.

```
SELECT diakok.nev AS diak, diakok.osztaly, tanarok.nev as osztalyfonok
FROM diakok RIGHT JOIN tanarok ON diakok.osztaly=tanarok.osztalyfonok;
```

Az eredmény:

diak	osztaly	osztalyfonok
Horváth Csilla	11.A	Kovács Lajos
NULL	NULL	Magyar Zoltán
NULL	NULL	Nagy Tímea
Horváth Béla	10.C	Kis Tamás
Turán Lajos	10.C	Kis Tamás
Horváth Béla	9.A	Szép Béla
Jeney Éva	12.B	Nyers Jolán

ÖNILLESZTÉS

Bizonyos esetekben előfordul, hogy egy táblát önmagához kell kapcsolnunk. Például, ha DIAKOK táblából osztálytárs párokat akarunk kiíratni. Ekkor a DIAKOK táblát önmagához kell kapcsolnunk (illesztenünk). Mivel ilyenkor ugyanaz a tábla két „példányban” is szerepel a lekérdezésben, ezét a táblához két álnevet is kell rendelnünk.

```
SELECT d1.nev, d2.nev
FROM diakok d1, diakok d2
WHERE d1.osztaly=d2.osztaly ;
```

BEÁGYAZOTT LEKÉRDEZÉS (ALLEKÉRDEZÉS)

Tegyük fel, hogy a TANULOK táblában tároljuk a tanulók nevét és életkorát. Feladatunk az, hogy írassuk ki azokat a tanulókat, akik idősebbek az átlagnál. Ehhez olyan lekérdezést kell készítenünk, amelynek WHERE záradékában egy újabb lekérdezés (allekérdezés) segítségével határozzuk meg az átlagéletkort:

```
SELECT nev, eletkor
FROM tanulok
WHERE eletkor > (SELECT avg (eletkor) FROM tanulok) ;
```

A SELECT INTO UTASÍTÁS

Ez az utasítás a lekérdezés eredményéből egy új táblát hoz létre. (Az Access szóhasználatában táblakészítő lekérdezést hoz létre.)

Szintaxis

```
SELECT mező1 [, mező2 [, ...]] INTO új tábla
FROM forrás
```

A SELECT...INTO utasítás a következő részekből áll:

Rész	Leírás
mező1, mező2	Az új táblába másolandó mezők neve.
új tábla	A létrehozandó tábla neve. A névnek követnie kell az elnevezési konvenciókat. Ha az új tábla neve megegyezik egy már létező tábla nevével, akkor elfogható hiba lép fel.
Forrás	Annak a már meglévő táblának a neve, amelyből a rekordokat kiválasztjuk. A forrás lehet egy vagy több tábla, illetve lekérdezés.

Táblakészítő lekérdezéssel rekordokat archiválhatunk, biztonsági másolatokat készíthetünk a táblákról, vagy olyan másolatokat állíthatunk elő, amelyeket másik adatbázisba exportálhatunk, vagy egy adott időszak adatait tartalmazó jelentések alapjaként használhatunk. A Havi eladások körzetenként nevű jelentést például úgy is előállíthatjuk, hogy minden hónapban lefuttatjuk ugyanazt a táblakészítő lekérdezést.

Az új táblához elsődleges kulcsot is definiálhatunk. Az új tábla létrehozásakor annak mezői öröklik a lekérdezés tábláiban szereplő mezők adattípusát és mezőméretét, más mező- vagy táblatulajdonságot azonban nem.

Ha meglévő táblához szeretnénk adatokat hozzáadni, akkor az INSERT INTO utasítás használatával hozzáfűző lekérdezést kell létrehoznunk.

Ha a táblakészítő lekérdezés futtatása előtt látni szeretnénk az új táblába kerülő rekordokat, vizsgáljuk meg egy azonos kiválasztási feltételeket használó SELECT utasítás eredményét.

AZ INSERT INTO UTASÍTÁS

Szép, és jó, hogy le tudjuk kérdezni az adatbázis tartalmát, de hogyan illeszthetünk be új rekordokat?

Erre szolgál az **INSERT INTO** utasítás. Használati módja:

INSERT INTO *tábla_neve* (*oszlop_neve1,oszlop_neve2,...*) VALUES (*érték1, érték2, ...*);

Ha minden mezőbe írunk adatot, akkor nem szükséges felsorolni az összes oszlop nevét, elég a következőt írni, természetesen az értékek megfelelő sorrendjének betartásával.

INSERT INTO *tábla_neve* VALUES (*érték1, érték2, ...*);

Példa:

Illesszünk be a *diakok* táblába két új sort, az alábbi adatok alapján:

Név: Horváth Csilla
Születési hely: Kaposvár
Születési év: 1989

Név: Horváth Béla
Születési hely: Budapest
Születési év: 1991

Ekkor a következőt kell írnunk:

```
INSERT INTO diakok (nev, szuletesi_hely, szuletesi_ev)
VALUES ('Horváth Csilla','Kaposvár', 1989);
INSERT INTO diakok (nev, szuletesi_hely, szuletesi_ev)
```

```
VALUES ('Horváth Béla','Budapest', 1991);
```

vagy (mivel az összes mező értékét megadtuk) elég a következőt írni:

```
INSERT INTO diakok
VALUES ('Horváth Csilla','Kaposvár', 1989);
INSERT INTO diakok
VALUES ('Horváth Béla','Budapest', 1991);
```

Tegyük fel, hogy új diák érkezett az iskolába, de nem sikerült megtudni minden adatát, csak a neve és a születési éve ismert. Ettől még beírhatjuk az adatait az adatbázisba, majd később pótoljuk a hiányzó adatokat.

Név: Kassai Tünde
Születési év: 1991

A megfelelő SQL utasítás az adatok beillesztésére:

```
INSERT INTO diakok (nev, szuletesi_ev)
VALUES ('Kassai Tünde',1991);
```

Most a *diakok* adattábla így néz ki:

nev	**szuletesi_hely**	**szuletesi_ev**
Kiss Irma	Budapest	1991
Horváth Béla	Kisvárda	1990
Horváth Béla	Budapest	1991
Jeney Éva	Karcag	1988
Turán Lajos	Abony	1990
Horváth Csilla	Kaposvár	1989
Kassai Tünde		1991

Gyakran előfordul, hogy a táblába illesztendő rekordot (rekordokat) egy másik táblából választjuk ki. Ilyenkor a következő szerkezetet kell alkalmaznunk:

INSERT INTO *tábla_neve* SELECT * FROM *másik_tábla_neve* WHERE *szűrőfeltételek*;

AZ UPDATE UTASÍTÁS

Időközben sikerült kideríteni Tünde születési helyét, ami Nagykanizsa. De hogyan illesszük be a mező tartalmát? Ehhez újabb utasítást kell megtanulnunk.

Az UPDATE utasítás segítségével az egyes rekordok tartalmát módosíthatjuk.

UPDATE *tábla_neve* SET *oszlop_neve = új_érték* WHERE *oszlop_neve = érték*;

Ez alapján illesszük be a hiányzó születési helyet!

```
UPDATE diakok
SET szuletesi_hely='Nagykanizsa'
```

```
WHERE nev='Kassai Tünde';
```

Vigyázzunk, mert ha a feltételnek több rekord is megfelel, a módosítás mindegyikre végrehajtódik.

Ha pl. az

```
UPDATE diakok
SET szuletesi_hely='Nagykanizsa'
WHERE szuletesi_ev=1991;
```

utasítást adtuk volna ki, akkor Kassa Tünde és Kiss Irma születési helye is megváltozott volna.

Ha törölni szeretnénk egy oszlop (mező) értékét, akkor a SET oszlop_neve = NULL szerkezetet használjuk!

A DELETE UTASÍTÁS

Nem csak rekordok módosításra lehet szükség, hanem például törlésére is, amelyhez a **DELETE** utasítást kell használnunk.

DELETE FROM *tábla_neve* WHERE *oszlopnév = érték*;

Példa: Kiss Irma más városba költözik, ezért kénytelen más iskolába folytatni tanulmányait. Emiatt töröljük az adatbázisból.

```
DELETE FROM diakok
WHERE nev='Kiss Irma';
```

Ezután a *diakok* tábla tartalma a következő lesz:

nev	**szuletesi_hely**	**szuletesi_ev**
Horváth Béla	Kisvárda	1990
Horváth Béla	Budapest	1991
Jeney Éva	Karcag	1988
Turán Lajos	Abony	1990
Horváth Csilla	Kaposvár	1989
Kassai Tünde	Nagykanizsa	1991

Ha egy táblából mindent törölni akarunk, akkor a **DELETE FROM tábla** utasítást használhatjuk. (Jól gondoljuk meg, hogy tényleg ezt akarjuk-e!)

Ha nem teljes rekordokat, csak bizonyos mezők értékét szeretnénk törölni, akkor az UPDATE utasítást kell használnunk!