

Adatbázisok

Dóka-Molnár Andrea
Gaskó Noémi

SQL

- A legtöbb relációs ABKR az adatbázist az SQL-nek (Structured Query Language, *angolul ejtsd: sequel*) nevezett lekérdezőnyelv segítségével kérdezi le és módosítja. (lekérdezés eredménye – reláció)
- IBM dolgozta ki az 1970-es években ('SEQUEL' → SQL).
- Szabványosítás: **1986 – ANSI** (American National Standards Institute); **1987 – ISO** (International Standards Organization).

SQL

- SQL szabványok:
 - SQL:86
 - SQL:89 (megnevezés: SQL1)
 - SQL:92 (megnevezés: SQL2)
 - SQL:99 (megnevezés: SQL3) – pl. triggererek
 - SQL:2003 - XML-re vonatkozó jellemzők (SQL/XML – új fejezet), pl. XML adattípus
 - SQL:2008 - INSTEAD OF trigger, TRUNCATE utasítás, FETCH záradék.
 - SQL:2011
 - SQL:2016

SQL

- Nyelvjárások: **Microsoft SQL (MSSQL)**, **MySQL**, **Oracle**, **SQL Server**, **PostgreSQL**, **Access** stb.
- Az SQL megvalósítások között vannak különbségek
 - **Transact-SQL** - a Microsoft saját nyelve, az SQL procedurális kiterjesztése; **PL/SQL** – Az Oracle saját nyelve
- Minden ABKR-nek saját grafikus kezelőfelülete van:
 - **MSSQL** - SQL Server Management Studio (SSMS)
 - **MySQL** – MySQL Workbench
 - **Oracle** – Oracle SQL Developer

SQL szintek

Fontosabb utasítások

- **Adatdefiníciós nyelv** - Data Definition Language (**DDL**)
 - **CREATE** – séma létrehozása
 - **ALTER** – séma módosítása
 - **DROP** – séma törlése
 - **TRUNCATE** – reláció csonkolása
- **Adatmanipulációs nyelv** - Data Manipulation Language (**DML**)
 - **INSERT** – adatok beszúrása
 - **UPDATE** – adatok módosítása
 - **DELETE** – adatok törlése

SQL szintek

Fontosabb utasítások

- **Lekérdezőnyelv** – [Data] Query Language ([D]QL)
 - **SELECT** – adatok lekérdezése
- **Adatvezérlő nyelv** - Data Control Language (**DCL**)
 - tranzakció kezelése: **COMMIT, ROLLBACK, SAVEPOINT**
 - adatvédelem, felhasználói hozzáférés szabályozása (pl. **GRANT**)

Adatdefiníciós nyelv

= **SQL Data Definition Language**

- Az adatbázis és az adatbázis objektumok létrehozására, kezelésére szolgál.

Lehetséges műveletek:

- Relációsémák létrehozása, módosítása, törlése
- Indexek létrehozása és kezelése
- Nézetek létrehozása
- Megszorítások létrehozása, törlése
- Triggerek (SQL3)

Relációtípusok SQL-ben

Reláció \longleftrightarrow tábla (TABLE) (SQL-beli megnevezés)

3 típusú tábla SQL-ben:

Reláció típusok SQL-ben

Reláció \longleftrightarrow tábla (TABLE) (SQL-beli megnevezés)

3 típusú tábla SQL-ben:

1. (Alap)tábla/tárolt reláció (table):

- A tárolás alapegysége; sorok halmaza.
- Fizikailag léteznek az adatbázisban \longleftrightarrow az adatbázisrendszer valamilyen fizikai struktúrában tárolja őket.
- Nem változnak addig, amíg valamilyen táblamódosító SQL-utasítás meg nem változtatja őket.
- DML műveletek végrehajtása – minden esetben.

Relációtípusok SQL-ben

2. Nézet(tábla) (view):

- Számításokból kapott relációk.
- Az eredmény nem tárolódik az adatbázisban, csak a nézet értelmezése.
- Egyes esetekben módosíthatjuk is őket.

3. Ideiglenes/temporális tábla (temporary table):

- Ideiglenes ideig tárolódnak (és csak bizonyos *session*-kben láthatóak), aztán törlődnek.
- DML műveletek végrehajtása – minden esetben.

Relációsémák létrehozása SQL-ben

- **CREATE TABLE** utasítás segítségével:

```
CREATE TABLE R(  
    A1 D1 [kiegészítés1], A2 D2 [kiegészítés2],  
    ..., An Dn [kiegészítésn],  
    [megszorítás1], ..., [megszorításk])
```

- R – reláció neve
- A_i – r reláció i -edik attribútumának neve ($i=1, \dots, n$)
(*a reláción belül egyedi kell legyen*)
- D_i – A_i attribútum doméniumában levő értékek típusa (*Ne feledjük a tartománymegszorítást!*)
- kiegészítés lehet: alapértelmezett attribútumérték megadása, megszorítás

Relációsémák létrehozása SQL-ben

- **Példa:**

```
CREATE TABLE Alkalmazottak (  
    SzemSzam INT,  
    Nev VARCHAR(30),  
    Fizetes REAL,  
    SzulDat DATE,  
    Nem CHAR(1),  
    ReszlegID INT  
);
```

- **Származtatott attribútum:**

```
CREATE TABLE Ertekek(  
    minimum INT, maximum INT,  
    Atlag AS (minimum + maximum) / 2);
```

Adattípusok SQL-ben

- Függ az SQL megvalósításától (*mi: MSSQL-t részletezzük*).
- Karakter sorok:
 - rögzített- (**CHAR(n)**) vagy változó hosszúságúak (**VARCHAR(n)**)
 - **NCHAR(n), NVARCHAR(n)** - Unicode karakterek tárolására
 - Ha nem adjuk meg a (maximális) méretet (**n**), alapértelmezés szerint 1 karakter hosszúságúra „vágja le” a sztringet.
 - Szöveg: egyes-aposztróf közé.
 - Két egyes-aposztróf = egynek felel meg (pl. 'Joe's Bar' ↔ Joe's Bar)
- Egész számok: **INT/INTEGER, TINYINT**
- Lebegőpontos értékek: **FLOAT, REAL**

smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to 9223372036854775807
decimal	variable	user-specified precision,exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision,exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision,inexact	6 decimal digits precision
double precision	8 bytes	variable-precision,inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Adattípusok SQL-ben

- Dátum- és időértékek: **DATE, TIME, DATETIME**
 - A formátumot meg kell adni. (pl. DATE: '2007-09-30', TIME: '15:30:02.5')
- **BIT** – integer, mely 1, 0 vagy NULL értékeket vehet fel.
 - Megj.: Logikai típusként használható.
- Más adattípusok: **TABLE, CURSOR** stb.

További infók: <https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-2017>

Saját adattípus definiálása

- Értéktartomány: egy adattípus, esetleg alapértelmezett értékkel és megszorításokkal.
- Attribútum típusának definiálhatunk egy értéktartományt.
- Több attribútumnak is lehet ugyanaz az értéktartománya, ami sok esetben hasznos lehet.
 - Pl. Ha változik valami, csak egy helyen kell változtatni.
- Általános szintaxis:
CREATE TYPE <név> **AS** <típusleírás>;
- Saját adattípus törlése:
DROP TYPE <név>

Saját adattípus definiálása

- Pl. Alias (másodnév) típus létrehozása VARCHAR adattípus alapján

```
CREATE TYPE CNP  
FROM varchar(13) NOT NULL
```

- Több a hátránya, mint az előnye. *Ezzel ellentétben:*

- Pl. Saját tábla típus definiálása

```
CREATE TYPE SzemelyTabla AS TABLE (  
    Nev VARCHAR(50),  
    Fizetes INT)
```

Relációsémák törlése/csonkolása

- **DROP TABLE** – reláció törlése

- eldobjuk a teljes leírást és mindazt, ami ehhez kapcsolódott hozzáférhetetlenné válik

Általános szintaxis: `DROP TABLE <reláció_név>`

- **TRUNCATE TABLE** – reláció csonkolása

- Törli a tábla összes sorát vagy a tábla egy bizonyos részét.

Általános szintaxis: `TRUNCATE TABLE <reláció_név>;`

Relációsémák törlése/csonkolása

- Különbség a DROP és TRUNCATE között:
 - TRUNCATE TABLE törli az összes sort a táblából, viszont a struktúráját nem változtatja (oszlopok, indexek, megszorítások maradnak).
 - DROP TABLE – tábla definícióját is törli az adatokkal együtt.

Relációsémák módosítása

- **ALTER TABLE** – relációséma (leírás) módosítására
`ALTER TABLE <reláció_név> <opciók>`

Opciók:

- Új attribútum definiálása:
`ADD <attr_név> <típus> <kiegészítés>`
 - Attribútum törlése: `DROP COLUMN <attr_név>`
 - Alapértelmezés szerinti érték megadására
 - Megszorítások definiálására/törlésére (`ADD/DROP CONSTRAINT`)
-
- Mindezek csak az épp aktuális adatokkal konzisztensen végezhetők el. (Pl. nem törölhető olyan attribútum, amire még van hivatkozás.)

Relációsémák módosítása

- **Példa:**

```
ALTER TABLE Alkalmazottak  
    ADD COLUMN Telefon CHAR(10);  
ALTER TABLE Alkalmazottak  
    DROP COLUMN SzülDat;
```

Alapértelmezés szerinti értékek

- Új sor beszúrása esetén, nem mindig ismerjük minden oszlop értékét.

Megoldás:

- **NULL érték megadása**

Alapértelmezés szerinti értékek

- Új sor beszúrása esetén, nem mindig ismerjük minden oszlop értékét.

Megoldás:

- **NULL érték megadása**
 - *Mit jelképez a NULL érték SQL-ben?*

Alapértelmezés szerinti értékek

- Új sor beszúrása esetén, nem mindig ismerjük minden oszlop értékét.

Megoldás:

- **NULL érték megadása**
 - *Mit jelképez a NULL érték SQL-ben?* – Ismeretlen vagy nem létező értéket.
- **Alapértelmezés szerinti** (kivéve, ha van NOT NULL megszorítás)
 - *Mikor nem lehet NULL értékkel definiálni egy attribútumot?*

Alapértelmezés szerinti értékek

- Új sor beszúrása esetén, nem mindig ismerjük minden oszlop értékét.

Megoldás:

- **NULL érték megadása**
 - *Mit jelképez a NULL érték SQL-ben?* – Ismeretlen vagy nem létező értéket.
- **Alapértelmezés szerinti (kivéve, ha van NOT NULL megszorítás) érték megadása**
 - *Mikor nem lehet NULL értékkel definiálni egy attribútumot?* – Ha az attribútum a tábla elsődleges kulcsa (vagy annak egy részét képezi).
 - **Egy alapértelmezés szerinti érték megadása NULL helyett \longleftrightarrow DEFAULT kulcsszóval.**

Alapértelmezés szerinti értékek

■ Példa:

```
CREATE TABLE Alkalmazottak (  
    SzemSzám INT [vagy: SzemSzám CNP],  
    Név VARCHAR(30),  
    Fizetés REAL,  
    SzülDat DATE DEFAULT '1900-01-01',  
    Nem CHAR(1) DEFAULT '?',  
    RészlegID INT,  
    Telefon CHAR(10) DEFAULT 'ismeretlen'  
);
```

Megszorítások

- **Megszorítások:** azon követelmények, melyeket az adatbázis adatai ki kell elégítsenek ahhoz, hogy helyeseknek tekinthessék őket. Vagyis **biztosítják az adatbázis konzisztenciáját** \leftrightarrow ne kerüljenek hibás értékek a táblákba, logikai ellentmondásokat ne tartalmazzanak a táblák.
- **Megszorítások típusai:**
 - Egyedi kulcs feltétel
 - Hivatkozási épség megszorítás
 - Értelmezéstartomány-megszorítások
 - Általános megszorítások

Megszorítások osztályozása

- **Egyedi kulcs feltétel:** egy relációban nem lehet két sor, melyeknek ugyanaz a kulcsértéke; vagyis: ha C egy R reláció kulcsa, akkor $\forall t_1, t_2 \in R$ sorok esetén
$$\pi_C(t_1) \neq \pi_C(t_2) \quad (\text{ld. relációs algebrai műveletek})$$
- **Hivatkozási épség megszorítás:** megköveteli, hogy egy objektum által hivatkozott érték létezzen az adatbázisban. (ld. külső kulcs)
- **Értelmezéstartomány-megszorítások:** egy attribútum az értékeit csak a megadott értékalmazból vagy érték-tartományból veheti fel.
- **Általános megszorítások:** tetszőleges követelmények, amelyeket be kell tartani az adatbázisban.

A hivatkozási épség megszorításról bővebben

Külső kulcs egy **KK** attribútum(halmaz) egy R_f relációból ú.h.:

- értékeinek halmaza megegyezik egy R_a reláció elsődleges kulcsának az értékalmazával;

- feladata: az R_f és R_a közötti kapcsolat modellezése.

■ R_f - reláció, mely **hivatkozik**; R_a – reláció, **amelyre hivatkozik**.

■ Más megnevezés:

- R_a – **apa** reláció, R_f – **fiú** reláció
- egy sorhoz az R_a -ból tartozhat több sor az R_f -ből
- R_a -ban elsődleges kulcs az az attribútum, amely a kapcsolatot megteremti. (*Fordítva nem állhat fenn a kapcsolat.*)

A hivatkozási épség megszorításról bővebben

- A hivatkozási épség megszorítás jelentése:
 - az R_a relációban elsődleges kulcsként kell deklarálni azt az attribútumot (esetleg attribútumhalmazt), melyre az R_f hivatkozik;
 - KK minden értéke az R_f -ből kell létezzen az R_a relációban, mint elsődleges kulcs értéke.
- **Pl:** Az Alkalmazottak (R_f) reláció hivatkozik a Részlegek (R_f) relációra a RészlegID külső kulcs segítségével ($KK=\{RészlegID\}$).

A hivatkozási épség megszorításról bővebben

Újabb példa:

Szállítók (SzállID, SzállNév, SzállCím)

Áruk (ÁruID, ÁruNév, MértEgys)

Szállít (SzállID, ÁruID, Ár)

- A Szállít reláció hivatkozik a Szállítók relációra a *SzállID* külső kulcs segítségével, illetve az Áruk relációra az *ÁruID* segítségével.
 - Szállítók, Áruk - apa relációk, Szállít – fiú reláció.

Példa külső kulcsra

Részlegek (RID, RNév) – *apa reláció*

<i>RészlegID</i>	<i>RNév</i>
1	Tervezés
2	Könyvelés
9	Beszerzés

Alkalmazottak (SzemSzám, Név, *RészlegID*, Fizetés) – *fiú rel.*

<i>SzemSzám</i>	<i>Név</i>	<i>RészlegID</i>	<i>Fizetés (euró)</i>
111111	Nagy Éva	2	300
222222	Kiss Csaba	9	400
456777	Szabó János	9	900
234555	Szilágyi Pál	2	700
123444	Vincze Ildikó	1	800
333333	Kovács István	2	500

A hivatkozási épség megszorítás sérülése

- A hivatkozási épség megszorítás az adatbázis aktualizálása esetén három esetben sérülhet:
 1. Új sor hozzáillesztése az R_f (fiú) relációhoz.
 2. Törlés az R_a (apa) relációból.
 3. Módosítás az R_f (fiú) relációban, illetve az R_a (apa) relációban.

A hivatkozási épség megszorítás sérülése

1) Új sor hozzáillesztése az R_f (fiú) relációhoz:

- a KK küls kulcs értékét csak akkor vihetjük be az R_f reláció megfelelő oszlopába, ha az már létezik elsődleges kulcsként az R_a relációban.

A hivatkozási épség megszorítás sérülése

1) Új sor hozzáillesztése az R_f (fiú) relációhoz:

- a KK küls kulcs értékét csak akkor vihetjük be az R_f reláció megfelelő oszlopába, ha az már létezik elsődleges kulcsként az R_a relációban.

Példa:

<i>RészlegID</i>	<i>RNév</i>
1	Tervezés
2	Könyvelés
9	Beszerzés

<i>SzemSzám</i>	<i>Név</i>	<i>RészlegID</i>	<i>Fizetés (euró)</i>
111111	Nagy Éva	2	300
222222	Kiss Csaba	9	400
456777	Szabó János	9	900
234555	Szilágyi Pál	2	700
123444	Vincze Ildikó	1	800
333333	Kovács István	2	500

Nem illeszthetünk olyan alkalmazottat az Alkalmazottak relációba, amelynek a *RészlegID* attribútum értéke nem létezik a Részlegek relációban (pl. 5).
Lehetséges értékek az aktuális állapotban: 1, 2 és 9.

A hivatkozási épség megszorítás sérülése

2) Törlés az R_a (apa) relációból:

- R_a relációból nem törölhetjük ki azokat az elsődleges kulcsértékeket, melyekre van hivatkozás az R_f relációban.

A hivatkozási épség megszorítás sérülése

2) Törlés az R_a relációból:

- R_a relációból nem törölhetjük ki azokat az elsődleges kulcsértékeket, melyekre van hivatkozás az R_f relációban.

Példa:

<i>RészlegID</i>	<i>RNév</i>	<i>SzemSzám</i>	<i>Név</i>	<i>RészlegID</i>	<i>Fizetés (euró)</i>
1	Tervezés	111111	Nagy Éva	2	300
2	Könyvelés	222222	Kiss Csaba	9	400
9	Beszerzés	456777	Szabó János	9	900
		234555	Szilágyi Pál	2	700
		123444	Vincze Ildikó	1	800
		333333	Kovács István	2	500

A példa esetén nem törölhetjük ki egyik részleget sem, mert mindegyikre van hivatkozás. Ha mégis kitörölnénk, az Alkalmazottak relációban maradnának ún. „lógó” sorok.

A hivatkozási épség megszorítás sérülése

3) Módosítás az R_f (fiú) relációban, illetve az R_a (apa) relációban:

- a) Ha az R_f relációban módosítunk egy KK értéket, csak az R_a relációban létezőre módosíthatjuk.

A hivatkozási épség megszorítás sérülése

3) Módosítás az R_f (fiú) relációban, illetve az R_a (apa) relációban:

a) Ha az R_f relációban módosítunk egy **KK** értéket, csak az R_a relációban létezőre módosíthatjuk.

Példa:

<i>RészlegID</i>	<i>RNév</i>
1	Tervezés
2	Könyvelés
9	Beszerzés

<i>SzemSzám</i>	<i>Név</i>	<i>RészlegID</i>	<i>Fizetés (euró)</i>
111111	Nagy Éva	2	300
222222	Kiss Csaba	9	400
456777	Szabó János	9	900
234555	Szilágyi Pál	2	700
123444	Vincze Ildikó	1	800
333333	Kovács István	2	500

Az Alkalmazottak relációban a „Kovács István”-ra vonatkozó bejegyzés esetén a RészlegID-t módosíthatjuk 1-re vagy 9-re, de nem módosíthatjuk például 3-ra.

A hivatkozási épség megszorítás sérülése

3) Módosítás az R_f (fiú) relációban, illetve az R_a (apa) relációban:

b) Ha az R_a relációban levő elsődleges kulcsot akarjuk módosítani, csak akkor tehetjük meg, ha nincs rá hivatkozás az R_f relációból.

A hivatkozási épség megszorítás sérülése

3) Módosítás az R_f (fiú) relációban, illetve az R_a (apa) relációban:

b) Ha az R_a relációban levő elsődleges kulcsot akarjuk módosítani, csak akkor tehetjük meg, ha nincs rá hivatkozás az R_f relációból.

Példa:

<i>RészlegID</i>	<i>RNév</i>
1	Tervezés
2	Könyvelés
9	Beszerzés

<i>SzemSzám</i>	<i>Név</i>	<i>RészlegID</i>	<i>Fizetés (euró)</i>
111111	Nagy Éva	2	300
222222	Kiss Csaba	9	400
456777	Szabó János	9	900
234555	Szilágyi Pál	2	700
123444	Vincze Ildikó	1	800
333333	Kovács István	2	500

A Részlegek tábla egyetlen RészlegID-jét sem módosíthatjuk, mivel mindegyikre van hivatkozás. Mivel a RészlegID elsődleges kulcs a Részlegek táblában, egy, már létező értékre való módosítás sem opció. ⁴⁶

Megszorítások SQL-ben

- SQL-ben az épségi megszorításokat az adatbázisséma részeként adjuk meg (CREATE TABLE vagy ALTER TABLE parancsokban).
 - Mikor egy adatbázis alkalmazás fut, az ABKR ellenőrzi, hogy a megszorítások teljesülnek-e, és ha nem, nem engedi meg a változtatást.
- Megszorítások általános alakja:
`[CONSTRAINT név] megszorítás típusa`
- Megj.: Érdemes elnevezni a megszorításokat. Ellenkező esetben a rendszer generál egyet számukra. Ez a név hibaüzenetekben jelenik meg, illetve az ALTER TABLE utasításban használhatjuk.

Megszorítások SQL-ben

- **Megszorítások típusai:**
 - **Kulcsok** és **külső kulcsok** - hivatkozási épség fenntartása
 - **Attribútumokra vonatkozó** megszorítások
 - NOT NULL feltételek
 - Egy attribútumra vonatkozó CHECK feltételek
 - **Sorokra vonatkozó** megszorítások
 - Sorra vonatkozó CHECK feltételek
 - Önálló megszorítások (Assertions) – MS SQL-ben nincs implementálva.

Egyedi kulcsok SQL-ben

- **(Ism.) Egyedi kulcs feltétel:** egy relációban nem lehet két sor, melyeknek ugyanaz a kulcsértéke, vagyis: ha C egy R reláció kulcsa $\forall t_1, t_2 \in R$ sorok esetén $\pi_C t_1 \neq \pi_C t_2$. ()
- Attribútum- vagy attribútum-lista kulcsként való deklarálásának módjai:
 - **UNIQUE** kulcsszó – reláció kulcsjelöltjeinek megadása.
 - **PRIMARY KEY** kulcsszó – elsődleges kulcs megadására (*kulcsjelöltek közül egyet választunk*).
- Kulcs esetén nincs értelme a DEFAULT értéknek.

Egyedi kulcsok megadása SQL-ben

- **Attribútum deklarációjában** (ha a kulcs **egyetlen** attribútumból áll)


Példa:

```
CREATE TABLE Áruk (  
    ÁruID INT PRIMARY KEY,  
    ÁruNév CHAR(30) UNIQUE,  
    MértEgys CHAR(10));
```


Egyedi kulcsok megadása SQL-ben

■ 2. Példa:

```
CREATE TABLE Alkalmazottak (  
    SzemSám INT PRIMARY KEY,  
    Név VARCHAR(30),  
    Fizetés REAL,
```



megszorítás

- SzülDat DATE DEFAULT '1900-01-01' Nem
 - CHAR(1),
 - RészlegID INT
 -);
- 
- alapértelmezett érték

Összetett kulcsok megadása SQL-ben

- **Adatbázis-séma részeként** (ha a kulcs **egy** vagy **több** attribútumból áll)

`PRIMARY KEY (attribútum lista)`

- CREATE TABLE utasítás kiegészítő részében meg lehet adni a kulcs deklarációját is.
- **Összetett kulcsok deklarációja** - csak ebben a formában.

Példa:

```
CREATE TABLE Szállít (  
    SzállID INT,  
    ÁruID INT,  
    Ár REAL,  
    PRIMARY KEY (SzállID, ÁruID) );
```

Elsődleges kulcsok és egyedi attribútumok

■ Hasonlóság:

- ABKR megoldja: ne lehessen két sor a táblában, melyben a kulcsként (PRIMARY KEY, UNIQUE) deklarált attribútumérték ugyanaz.

■ Különbség:

- Egy relációban **csak 1 elsődleges kulcs (PK)** lehet, míg **egyedi attribútum (U)** több is.
- **Külső kulcs** - csak **elsődleges kulcsára** hivatkozhat egy relációnak.
- A **PRIMARY KEY** egyetlen attribútuma sem kaphat **NULL** értéket DE a **UNIQUE** megszorításnál **szerepelhetnek** **NULL** értékek.
- *ABKR-től függően:* az **elsődleges kulcsnak** megfelelően az ABKR **indexállományt** hoz létre. Egyedi attribútum deklarálása esetén az adatbázis adminisztrátor kell az indexállományt létrehozza a keresés gyorsítása érdekében.

Külső (idegen) kulcs megadása SQL-ben

- **REFERENCES** kulcsszó használatának két lehetősége:
 - **Attribútumként** (egyetlen attribútumból álló kulcsra)

Példa:

```
CREATE TABLE Csoportok (  
    CsopKod CHAR(3) PRIMARY KEY,  
    Evfolyam INT,  
    SzakKod CHAR(3)  
);  
  
CREATE TABLE Diakok (  
    DiakID INT PRIMARY KEY,  
    DNeve VARCHAR(30),  
    DCsopKod CHAR(3) REFERENCES Csoportok(CsopKod)  
);
```

Külső (idegen) kulcs megadása SQL-ben

- **REFERENCES** kulcsszó használatának két lehetősége:
 - **Sémaelemként** (**egy vagy több** attribútumból álló kulcsra)
`FOREIGN KEY (attribútum lista)`
`REFERENCES relációnév (attribútum lista)`

Példa:

```
CREATE TABLE Csoportok (  
    CsopKod CHAR(3) PRIMARY KEY,  
    Evfolyam INT );  
  
CREATE TABLE Diakok (  
    DiakID INT PRIMARY KEY,  
    DCsopKod CHAR(3),  
    DNeve VARCHAR(30),  
    FOREIGN KEY (DCsopKod) REFERENCES Csoportok (CsopKod)  
);
```

Hivatkozási épség fenntartása

- (Ism.) A hivatkozási épség megszorítás az adatbázis aktualizálása esetén sérülhet:

1. R_f (fiú) relációba történő beszúrásnál vagy a relációban történő módosításnál R_a -ban nem szereplő értéket adunk meg.

2. R_a (apa) relációból való törlés vagy módosítás „lógó” sorokat eredményez az R_f (fiú) relációban.

- Példa: R_a = Csoportok, R_f = Diákok.
- Ha külső kulcsot deklarálnak, azt jelenti, hogy a külső kulcs bármely nem-NULL értéke elő kell forduljon a hivatkozott reláció megfelelő attribútumában.
- Az ABKR **négy lehetséges megoldást** ajánl az adatbázis tervezőjének, ahhoz, hogy ezt a megszorítást az adatbázis módosításai közben fenn tudja tartani.

Hivatkozási épség fenntartása

1) Alapértelmezés szerinti eljárás:

- ha a feltétel megsérülne, a módosítást az ABKR visszautasítja.

2) NULL értékre állítás módszere (SET NULL):

- a törölt vagy módosított apa sorhoz tartozó fiú sorokban a külső kulcs értékét NULL-ra változtatja.

Példa: Diakok táblában a csoport kódjának (DCsopKod) értékeit állítsuk NULL-ra az érintett sorokban.

3) Alapértelmezés szerinti értékre állítás módszere (SET DEFAULT):

- A megfelelő fiú sorokban a külső kulcs értéke az alapért. szerinti értékre módosul.
- Ha nincs alapértelmezett érték megadva → NULL.
- Elég körülményes – helyette: 1,2 vagy trigger

Hivatkozási épség fenntartása

4) Továbbgyűrűző eljárás (CASCADE):

- Alkalmazás: a hivatkozott (apa) táblában történő törlés és módosítás esetén.
- Törlés az apa relációban: a megfelelő értékeket tartalmazó sorok törlése a hivatkozó (fiú) táblából.
- Módosítás az apa relációban: az ABKR módosítja a hivatkozó (fiú) táblában is a megfelelő értékeket.
- Megj. Óvatosan a CASCADE-del (*nagy adatbázis esetén nem tudhatjuk majd, hova tűntek bizonyos soraink*)!

Példa: Diákok tábla értékeit igazítjuk a változáshoz.

- Csoport törlése: töröljük a Diákok tábla megfelelő sorait.
- Csoport módosítása: Diákok táblában is változik az érték.⁵³

Hivatkozási épség fenntartása

Eljárás kiválasztása:

- Idegen kulcsot deklarálásakor - SET NULL, SET DEFAULT és a CASCADE stratégia módosításra és törlésre is megadható.

Szintaxis:

```
<idegen kulcs deklarálása>  
ON [UPDATE | DELETE  
    {NO ACTION|CASCADE|SET NULL|SET DEFAULT} ]
```

- Ha ezt nem adjuk meg – alapértelmezés szerinti stratégia (\Leftrightarrow NO ACTION).

Hivatkozási épség fenntartása

1. példa:

```
CREATE TABLE Diakok (  
    DiakID INT PRIMARY KEY,  
    DCsopKod CHAR(3),  
    DNev VARCHAR(30),  
    FOREIGN KEY(DCsopKod) REFERENCES Csoportok(CsopKod)  
        ON DELETE SET NULL  
        ON UPDATE CASCADE  
);
```

Hivatkozási épség fenntartása

2. példa:

```
CREATE TABLE Szállít (
    SzállID INT REFERENCES Szállítók (SzállID)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
    ÁruID INT REFERENCES Áruk (ÁruID)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
    Ár REAL,
    PRIMARY KEY (SzállID, ÁruID)
);
```

Hol a hiba?

Hivatkozási épség fenntartása

2. példa:

```
CREATE TABLE Szállít (
    SzállID INT REFERENCES Szállítók (SzállID)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
    ÁruID INT REFERENCES Áruk (ÁruID)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
    Ár REAL,
    PRIMARY KEY (SzállID, ÁruID)
);
```

Hol a hiba? Indoklás?

Egymásra hivatkozás

Példa: *Beállíthatjuk-e ezt az egymásba hivatkozást?*

```
CREATE TABLE Alkalmazottak (  
    SzemSzám INT,  
    Név VARCHAR(30),  
    Fizetés REAL,  
    SzülDat DATE DEFAULT '1900-01-01',  
    RészlegID INT NOT NULL REFERENCES  
        Részlegek(RészlegID)  
);  
  
CREATE TABLE Részlegek (  
    RészlegID INT,  
    Rnév VARCHAR(20),  
    ManagerSzemSzám INT NOT NULL REFERENCES  
        Alkalmazottak (SzemSzám)  
);
```

Egymásra hivatkozás

Példa: *Beállíthatjuk-e ezt az egymásba hivatkozást?*

```
CREATE TABLE Alkalmazottak (  
    SzemSzáma INT,  
    Név VARCHAR(30),  
    Fizetés REAL,  
    SzülDat DATE DEFAULT '1900-01-01',  
    RészlegID INT NOT NULL REFERENCES  
        Részlegek(RészlegID)  
);
```

```
CREATE TABLE Részlegek (  
    RészlegID INT,  
    Rnév VARCHAR(20),  
    ManagerSzemSzáma INT NOT NULL REFERENCES  
        Alkalmazottak (SzemSzáma)  
);
```

Igen, ha valamelyik tábla esetén
megengedjük a NULL értéket a
külső kulcsnak!

Attribútumokra vonatkozó megszorítások

- Relációséma definálásakor adhatók meg.

1) Attribútum értéke nem lehet NULL.

- NOT NULL kulcsszóval
- Példa:

```
CREATE TABLE Áruk (  
    ÁruID INT PRIMARY KEY,  
    ÁruNév CHAR(30) NOT NULL,  
    MértEgys CHAR(10)  
);
```

- **Fontos:** Egyedi kulcs értéke sohasem lehet NULL (azonosító szerep elvesztése).
 - ABKR-ek általában nem engedik meg, hogy egyedi kulcsnak deklaráljunk olyan attribútumot, mely értéke lehet NULL.

Attribútumokra vonatkozó megszorítások

- 2) Bonyolultabb megszorítás rendelése egy attribútumhoz
- CHECK kulcsszó segítségével (`CHECK <feltétel>`)
 - Az ABKR visszautasítja azokat a hozzáillesztéseket, módosításokat, ahol ez a feltétel nem áll fenn.
 - A *feltételben* csak az adott attribútum neve szerepelhet, *más attribútumok (más relációk attribútumai is) csak alkérdésben szerepelhetnek.*
 - NULL értékek megadása megengedett.

Attribútumokra vonatkozó megszorítások

2) Bonyolultabb megszorítás rendelése egy attribútumhoz

- CHECK kulcsszó segítségével:

- Példa:

```
CREATE TABLE Csoportok (  
    CsopKod CHAR(3) PRIMARY KEY,  
    Evfolyam INT  
    CHECK (Evfolyam >= 1 and Evfolyam <= 5),  
    SzakKod CHAR(3)  
);
```

Másképp:

```
Evfolyam INT CHECK (Evfolyam IN (1, 2, 3, 4, 5))
```

Attribútumokra vonatkozó megszorítások

2) Bonyolultabb megszorítás rendelése egy attribútumhoz

- CHECK kulcsszó segítségével:

- Más példa:

```
CREATE TABLE Diakok (  
    DiakID INT PRIMARY KEY,  
    DNeve VARCHAR(30),  
    DCsopKod CHAR(3)  
    CHECK (DCsopKod IN (SELECT CsopKod FROM Csoportok))  
);
```

Attribútumokra vonatkozó megszorítások ellenőrzése

- Attribútum-alapú ellenőrzést csak beszúrásnál és módosításnál hajt végre a rendszer.
- Példa: `CHECK (Evfolyam >= 1 AND Evfolyam <= 5)`
 - Ha a beszúrt vagy módosított évfolyam értéke <1 vagy $>5 \implies$ a rendszer nem hajtja végre az utasítást.
- Példa: `CHECK (DCsopKod IN (SELECT CsopKod
FROM Csoportok))`
 - *Csoportok táblából való törlésnél: a feltételt nem ellenőrzi a rendszer.*

Sorra vonatkozó megszorítások

- Egy megszorítás hivatkozhat több attribútumra is, és akár több relációt érintő feltételeket is tartalmazhat.
- Sorra vonatkozó CHECK feltételek
 - A CHECK megszorítást a séma elemeként adjuk meg.
 - A feltétel(ek) egyetlen reláció sorá(i)ra tesznek megszorításokat.
 - A feltételben tetszőleges oszlop és reláció szerepelhet.
 - Más relációk attribútumai: csak alkérdésben jelenhetnek meg.
- Csak beszúrásnál és módosításnál ellenőrzi a rendszer.

Sorra vonatkozó megszorítások

- Egy megszorítás hivatkozhat több attribútumra is, és akár több relációt érintő feltételeket is tartalmazhat.

Sorra vonatkozó CHECK feltételek:

- Példa:

```
CREATE TABLE FilmSzínész (  
    név VARCHAR(30) PRIMARY KEY,  
    cím VARCHAR(255) NOT NULL,  
    nem CHAR(1),  
    születésiDátum DATE,  
    CHECK (nem = 'N' OR név NOT LIKE 'Ms. %')  
);
```

Sorra vonatkozó megszorítások

- Egy megszorítás hivatkozhat több attribútumra is, és akár több relációt érintő feltételeket is tartalmazhat.

Sorra vonatkozó CHECK feltételek:

- Példa:

```
CREATE TABLE FilmSzínész (  
    név VARCHAR(30) PRIMARY KEY,  
    cím VARCHAR(255) NOT NULL,  
    nem CHAR(1),  
    születésiDátum DATE,  
    CHECK (nem = 'N' OR név NOT LIKE 'Ms.%.')  
);
```

- Ha egy színész neme férfi, akkor a neve nem kezdődhet ‘Ms.’-el.

Sorra vonatkozó megszorítások

■ Példa:

```
CREATE TABLE Alkalmazottak (  
    SzemSzám CHAR(13) PRIMARY KEY,  
    Név VARCHAR(30),  
    RészlegID INT REFERENCES Részlegek (RészlegID),  
    Fizetés INT);  
  
CREATE TABLE Részlegek (  
    RészlegID INT PRIMARY KEY,  
    RNév VARCHAR(30),  
    ManagerSzemSzám INT REFERENCES  
        Alkalmazottak (SzemSzám),  
    CHECK (ManagerSzemSzám NOT IN  
        (SELECT SzemSzám FROM Alkalmazottak  
        WHERE Fizetés < 500)) );
```

Sorra vonatkozó megszorítások

- Példa:

```
CREATE TABLE Alkalmazottak (  
    SzemSzám CHAR(13) PRIMARY KEY,  
    Név VARCHAR(30),  
    RészlegID INT REFERENCES Részlegek (RészlegID),  
    Fizetés INT);  
  
CREATE TABLE Részlegek (  
    RészlegID INT PRIMARY KEY,  
    RNév VARCHAR(30),  
    ManagerSzemSzám INT REFERENCES  
        Alkalmazottak (SzemSzám),  
    CHECK (ManagerSzemSzám NOT IN  
        (SELECT SzemSzám FROM Alkalmazottak  
        WHERE Fizetés < 500)) );
```

- Egy manager fizetése legalább 500 euró kell legyen.

Megszorítások elnevezése

- CONSTRAINT kulcsszó segítségével
- Példa (CREATE TABLE utasításban) - *Filmszínész reláció definiálásában:*

```
nev CHAR(30) CONSTRAINT Nev_PK PRIMARY KEY,
```

```
nem CHAR(1) CONSTRAINT FerfiVagyNo_Ck  
CHECK (nem IN ('F', 'N')),
```

```
CONSTRAINT Titulus_Ck  
CHECK (nem = 'N' OR név NOT LIKE 'Ms.\%')
```

- Megj. NOT NULL megszorítás – nem nevezhető el
 - DEFAULT – SQL-ben megszorításként jelenik meg.

Megszorítások elnevezése

- CONSTRAINT kulcsszó segítségével
- Példa (ALTER TABLE utasításban)-*megszorítás rendelése a táblához:*

```
ALTER TABLE Filmszineszek  
ADD CONSTRAINT Nev_PK PRIMARY KEY (nev),
```

```
ALTER TABLE Filmszineszek  
ADD CONSTRAINT FerfiVagyNo_Ck  
CHECK (nem IN ('F', 'N')),
```

```
ALTER TABLE Filmszineszek  
ADD CONSTRAINT Titulus_Ck  
CHECK (nem = 'N' OR név NOT LIKE 'Ms.\%')
```

Megszorítások elnevezése és törlése

- Ajánlott nevet adni a hivatkozási épség megszorításoknak is: törölhetjük őket, esetleg újakat értelmezhetünk, anélkül, hogy a tábladeklarációt megváltoztatnánk.

```
ALTER TABLE Diakok
```

```
ADD CONSTRAINT Csopkod_FK FOREIGN KEY (DCsopKod)  
REFERENCES Csoportok (CsopKod),
```

- **Megszorítás törlése:** ALTER TABLE utasításban – *DROP CONSTRAINT* kulcsszóval:

```
ALTER TABLE Filmszineszek
```

```
DROP CONSTRAINT Csopkod_FK
```


Adatmanipulációs nyelv

= **SQL Data Manipulation Language (DML)**

- Adatbázis adatainak karbantartására (bevitel, módosítás, törlés) szolgál.
- **Lehetséges műveletek:**
 - Bevitel: **INSERT**
 - Módosítás: **UPDATE**
 - Törlés: **DELETE**
 - *[Adatlekérés: SELECT]*
- A karbantartó műveletek által kezelt adatoknak ki kell elégíteni az érintett oszlopokra/táblákra vonatkozó megszorításokat.
- Lehetőség van ú.n. **triggererek** (kioldók) létrehozására, melyek a karbantartó műveletek végrehajtásának következményeként (előtt/helyett vagy után) automatikusan aktivizálódnak és módosíthatják a műveletek hatását. *(ismét lsd.: későbbi kurzusokon)*

Adatok felvitele (INSERT)

```
INSERT INTO R (A1, A2, ..., An)  
VALUES (v1, v2, ..., vn) ;
```

- Művelet eredménye: az R relációba egy új sor kerül, ahol az A_i attribútum értéke v_i , $\forall i = \{1, 2, \dots, n\}$.
- Ha a reláció minden attribútumának megadjuk az új értékét, az attribútumlistát elhagyhatjuk. **Vigyázat:** az értékek sorrendje meg kell egyezzen az attribútumok relációbeli sorrendjével!
- Ha az attribútumlista nem tartalmazza R összes attribútumát, akkor a hiányzó attribútumok az alapértelmezés szerinti (*NULL* vagy *DEFAULT* megszorításnak megfelelő) értéket kapják.

Adatok felvitele

- Legyen a Turistak reláció

`Turistak(TID, TNeV, Email, HKod)`

és a következő parancs:

```
INSERT INTO Turistak(TID, TNeV, HKod, Email)
VALUES (4, 'Kukorica Jancsi', 5, 'jancsi@email.hu');
```

- Minden beszúrási művelet esetén az ABKR ellenőrzi a megszorításokat.
 - Ha a beszúráásra kerülő sor nem teljesít egyet is a megszorítások közül, hibát jelez és nem hajtja végre a beszúrási műveletet.
- Az előbbi parancsot megadhatjuk attribútumlista nélkül is:

```
INSERT INTO Turistak
VALUES (4, 'Kukorica Jancsi', 'jancsi@email.hu', 5);
```

Adatok felvitele

Turistak(TID, TNeV, Email, HKod)

■ Példa:

```
INSERT INTO Turistak(TID, TNeV, HKod)
VALUES (5, 'Csongor', 1);
```

- Email attribútum nincs megadva + nincs alapértelmezett értéke + lehet NULL is → értéke NULL értékkel fog feltöltödni.

Adatok felvitele

Turistak(TID, TNeV, Email, HKod)

■ Példa:

```
INSERT INTO Turistak(TID, TNeV, HKod)
VALUES (5, 'Csongor', 1);
```

- Email attribútum nincs megadva + nincs alapértelmezett értéke + lehet NULL is → értéke NULL értékkel fog feltöltődni.

■ Újabb példa:

```
INSERT INTO Turistak(TID, TNeV, HKod)
VALUES (5, 'Tünde', 1);
```

- a parancs hibát fog jelezni ↔ TID **elsődleges kulcs**

Adatok felvitele

- Több sor beszúrása – attribútumlista helyett alkérdés megadása

- **Példa:** *Részlegek relációba történő beszúrás*

Részlegek (RészlegID, Név, Helység, ManSzemSzám) ;
Alkalmazottak (SzemSzám, Név, Fizetés, Cím,
RészlegID) ;

- A két reláció kölcsönösen hivatkozik egymásra külső kulcsos megszorítással.

Adatok felvitele

■ Példa: *Részlegek relációba történő beszúrás*

Részlegek (RészlegID, Név, Helység, ManSzemSzám) ;
Alkalmazottak (SzemSzám, Név, Fizetés, Cím,
RészlegID) ;

- A két reláció kölcsönösen hivatkozik egymásra külső kulcsos megszorítással.
- Az Alkalmazottak táblába először begyűjtjük az összes managert, átvesszük az összes létező részleget a Részlegek relációba a manager személyi számával együtt.

```
1) INSERT INTO Részlegek (RészlegID, ManSzemSzám)
2)   SELECT DISTINCT RészlegID, SzemSzám
3)   FROM Alkalmazottak;
```

Adatok felvitele

■ **Példa:** *Részlegek relációba történő beszúrás*

Részlegek (RészlegID, Név, Helység, ManSzemSzám) ;
Alkalmazottak (SzemSzám, Név, Fizetés, Cím,
RészlegID) ;

- 1) INSERT INTO Részlegek (RészlegID, ManSzemSzám)
- 2) SELECT DISTINCT RészlegID, SzemSzám
- 3) FROM Alkalmazottak;

■ A lekérdezést teljes egészében ki kell értékelni, mielőtt bármely sort beszúrnánk.

- A 2-3. sorok közötti lekérdezést az 1. sor beszúrása előtt kellene elvégezni. Így az 1. sorban beszúrt új részlegek nem lehetnek hatással az esetleges feltételekre.

Adatok törlése (DELETE)

DELETE FROM R WHERE <feltétel>;

- Művelet eredménye: az R relációból **kitörlődik az összes sor**, amely megfelel a feltételnek.
- Példa: `Turisták(TID, TNeve, Email, HKod)` reláció:

```
DELETE FROM Turisták  
WHERE TID = 4;
```
- Egy adott sor törlése esetén a legjobb, ha a **reláció elsődleges kulcsának az értékét**, vagy **egy egyedi kulcsnak az értékét** adjuk meg. *Ellenkező esetben*: valószínűleg több sor is törlődik.

Adatok törlése

- A WHERE feltétel tartalmazhat alkérdéseket is.

- Példa:

Helysegek(HKod, HNev, ...)

Szallasok (SzID, SzNev, *Hkod*, ..., *Ar/Ej*)

Turistak(TID, TNev, Email, *HKod*)

```
DELETE FROM Turistak
```

```
WHERE HKod IN (SELECT HKod FROM Szallasok);
```

Adatok törlése

- A WHERE feltétel tartalmazhat alkérdéseket is.

- Példa:

Helysegek (HKod, HNev, ...)

Szallások (SzID, SzNev, Hkod, ..., Ar/Ej)

Turisták (TID, TNev, Email, HKod)

```
DELETE FROM Turisták
```

```
WHERE HKod IN (SELECT HKod FROM Szallások);
```

- Töröljük azokat a turistákat, akik olyan helységben laknak, ahol van szállás.

- Ha nem adunk meg feltételt \Rightarrow a reláció összes sora törlődik.

pl.

```
DELETE FROM Turisták
```

Törlés vagy csonkolás?

- **(Ism.) TRUNCATE TABLE** – reláció csonkolása
 - Törli a tábla összes sorát vagy a tábla egy bizonyos részét.
- Általános szintaxis: `TRUNCATE TABLE <reláció_név>;`
- Különbség a **DELETE** és **TRUNCATE** között:
 - **DELETE-DML, TRUNCATE-DDL (!!!)**
 - TRUNCATE esetén nincs lehetőség WHERE feltétel megadására.
 - IDENTITY – TRUNCATE - visszaállítódik a kezdeti értékre, DELETE esetén folytatja a sorszámozást.
 - TRUNCATE – törli a sorokat anélkül, hogy naplózná a sorok egyenkénti törlését, ellentétben a DELETE-tel. \Rightarrow TRUNCATE gyorsabb, mint a DELETE és kevesebb rendszer- és tranzakciós naplózási erőforrást használ.

Törlés vagy csonkolás?

■ **TRUNCATE TABLE** – reláció csonkolása

- Törli a tábla összes sorát vagy a tábla egy bizonyos részét.

Általános szintaxis: `TRUNCATE TABLE <reláció_név>;`

■ Nem használható olyan táblák esetén:

- Amelyekre van külső kulcsos hivatkozás. (Kivétel: ha saját magára hivatkozik a tábla.)
- Indexelt nézetben szerepel. (*lsd.későbbi kurzuson*)
- Ilyenkor: DELETE-tel dolgozzunk.

Adatok módosítása (UPDATE)

```
UPDATE R SET <új értékadások> WHERE <feltétel>;
```

- Az értékadásokat vesszővel választjuk el.
- **Művelet eredménye:** az összes olyan R-beli sorban, amelyek megfelelnek a feltételnek, az értékadáslistának megfelelően a komponensek (attribútumértékek) módosulnak.
- Példa:

```
UPDATE Alkalmazottak  
SET Név = 'Nagy Éva Mária', Fizetés = 450  
WHERE SzemSzám = 111111;
```

```
Alkalmazottak(SzemSzám, Név, Fizetés, Cím, RészlegID);
```

Adatok módosítása

- Több sort is módosíthatunk egy UPDATE parancs segítségével.

- **Példa:** UPDATE Alkalmazottak
SET Fizetés = Fizetés * 1.2
WHERE Fizetés < 600;

- A WHERE feltétel tartalmazhat alkérdéseket is:

- **Példa:** UPDATE Alkalmazottak
SET Fizetés = Fizetés * 1.5
WHERE SzemSzám IN
(SELECT SzemSzám FROM Managerek);

Alkalmazottak (SzemSzám, Név, Fizetés, Cím, RészlegID) ;
Managerek (SzemSzám) ;

Feladatok

Országok (OKod, ONev)

Helysegek (HKod, HNeve, OKod)

Szállástípusok (SzTID, SzTNeve)

Szállások (SzID, SzNeve, Hkod, SztID, Csillag, Ar/Ej)

Turisták (TID, TNeve, Email, HKod)

Foglalások (SzID, TID, KezdDatum, NapokSzama, Osszar)

***U1)** Növeljük minden 3csillagnál nagyobb értékeléssel rendelkező szállások árát 10%-kal, míg a többi szállásét 5%-kal!*

Feladatok

U1) Növeljük minden 3csillagnál nagyobb értékeléssel rendelkező szállások árát 10%-kal, míg a többi szállását 5%-kal!

1. megoldás: 2 UPDATE utasítással:

```
UPDATE Szallasok  
SET Ar/ej = Ar/ej*1.10  
WHERE Csillag>3;
```

```
UPDATE Szallasok  
SET Ar/ej = Ar/ej*1.05  
WHERE Csillag<=3;
```

CASE utasítás UPDATE műveletnél

U1) Növeljük minden 3csillagnál nagyobb értékeléssel rendelkező szállások árát 10%-kal, míg a többi szállásét 5%-kal!

2. megoldás: CASE utasítással:

```
UPDATE Szallasok
SET Ar/ej = CASE
    WHEN Csillag>3 THEN Ar/ej*1.10
    ELSE Ar/ej*1.05
END
```

UPDATE alapértelmezett értékek használatával

U2) Azon szállások árát állítsuk be az alapértelmezés szerinti értékre (25euro), amelyeknél Csillag ≤ 2 !

Szállások (SzID, SzNev, Hkod, SztID, Csillag, Ar/Ej)

UPDATE alapértelmezett értékek használatával

U2) Azon szállások árát állítsuk be az alapértelmezés szerinti értékre (25euro), amelyeknél Csillag ≤ 2 !

Szállások (SzID, SzNev, Hkod, SztID, Csillag, Ar/Ej)

```
UPDATE Szallasok  
SET Ar/ej = DEFAULT  
WHERE Csillag<=2
```

- Ha az adott mezőhöz nincs DEFAULT megszorítás rendelve, az alapértelmezett érték NULL lesz.

Problémák adatmódosító műveletek végrehajtásakor

- Problémák a törlés, beszűrés esetén akkor jelenhetnek meg, ha egy relációnak nincs egyedi kulcsa:
 - ugyanazt a sort többször is be tudjuk szűrní;
 - törlés esetén pedig egy parancs az összes azonos sort kitörli, nem tudjuk csak az egyiket azonosítani.
- **Példa:** Legyen a Szállít (SzállID, ÁruID, Ár) reláció és tegyük fel, hogy nem adtuk meg a (SzállID, ÁruID) párost elsődleges kulcsnak.

<i>SzállID</i>	<i>ÁruID</i>	<i>Ár</i>
111	45	2.5
222	45	2.6
111	67	1.7
111	56	2.2
222	67	1.8
222	56	2.2

```
INSERT INTO Szállít  
VALUES (111, 45, 2.5)
```

```
DELETE FROM Szállít  
WHERE SzállID = 111 AND ÁruID = 45
```