Gyakorlati óravázlat 05 – NULL, DML, tranzakciók

1. NULL kezelése
2. UNION, UNION ALL, INTERSECT, MINUS
3. egyszerű INSERT, UPDATE, DELETE
4. COMMIT, ROLLBACK

\_\_\_\_\_\_\_\_\_\_\_\_

**NULL érték kezelése**

NULL nem konkrét érték, háromállapotú logikát eredményez. Ha konkrét értékként, konstansként kezelnénk – ahogy sok programnyelv teszi, akkor a NULL = NULL igaz lenne, például 0 = 0 fordulna le, vagy "" = "" -ként. Viszont, ha a NULL = NULL igaz lenne, akkor például a nem ismert életkorúak szükségszerűen kor szerint megegyeznének, ami nyilván nem lenne helyes. És a példában látható, hogy a tagadás sem lehet helyes, hiszen attól még, hogy valami nem ismert, attól még nem feltétlenül mindenkinek különböző az életkora, tehát a NULL <> NULL sem lehet igaz. Ebben az esetben az állapot mindig UNKNOWN, vagy logikailag a harmadik állapot. A harmadik állapot leellenőrzése az IS és az IS NOT operátorral lehet, amelynek **a jobboldai operandusa kizárólag** NULL lehet. 1 IS 2 nem helyes.

* + Szintaktikai hiba
    - 1 IS 2
    - 'valami' IS 'valami'
  + UNKNOWN esetek
    - 1 = NULL, NULL = 1, NULL <> 1, 1 <> NULL
    - NULL = NULL
    - NULL <> NULL
  + Igaz esetek
    - NULL IS NULL

**Üres karaktersorozat az NULL**

Az üres szövegről, azaz a '' karaktersorozat megegyezik a NULL fogalmával ('' IS NULL az igaz igazságértékű), ami „zavar az erőben“, hiszen azt mondtUK korábban, hogy a NULL nem konstans érték – az üres szöveg pedig az. Itt egy logikai csavarás van, a pragmatika oltárán feláldozva a matematikai korrektséget, mert itt kihasználja az Oracle, hogy bár a NULL jelentése sokféle lehet, de ebből egyet preferál. Nevezetesen a kényszerek ellenőrzését. INSERT még nem volt eddig, de NOT NULL kényszer igen. Ha valamire azt írnánk be, hogy NOT NULL tulajdonságú, és az üres szöveget elfogadnánk mint NULL-tól különböző érték, akkor mennyire ellenőrizzük le ténylegesen, hogy üresen hagytuk a mezőt, vagy maga az „üresség“ jelzi azt, hogy valami NULL értékű? A kettő között nem lenne érdemi különbség, így a NULL értelmezése szövegek esetében annyi, hogy „üres mező“.

Csak jelzem, hogy számok esetében az üres mező az implemetációban szinte mindig a „0“ érték (minden biten 0 van) eltárolása, ami bajosan megkülönböztethető a 0 értéktől. Nagyon ritka, hogy ettől egy-egy programnyelv eltérjen, mert fizikailag egyetlen bájtot sem lehet „üresen“ hagyni – bináris számrendszert használunk. 0 IS NULL azonban hamis. A szövegek esetében ne feledjük, hogy van termináló karakter, a \0 – ez jelzi a szöveg végét, ami nyilvánvalóan állhat a szöveg legelején is. A trükk tehát a pragmatikában keresendő, a NOT NULL tényleg törekedjen arra szövegeknél, hogy az nem lehet üres mező. Erre érdemes odafigyelni.

**Feladatok**

1. Fejtsük meg az alábbi utasításokat! (Lehet kérni őket, hogy még futtatás előtt tippeljék meg, hogy melyiknek mi lesz az eredménye és csak utána futtassák le. Aztán közösen megbeszéljük.)

SELECT 'Lefutott' AS megmi FROM DUAL WHERE NULL = NULL;

SELECT 'Lefutott' AS megmi FROM DUAL WHERE NULL <> NULL;

SELECT 'Lefutott' AS megmi FROM DUAL WHERE NULL IS NULL;

SELECT 'Lefutott' AS megmi FROM DUAL WHERE '' = NULL;

SELECT 'Lefutott' AS megmi FROM DUAL WHERE NULL LIKE '%';

SELECT 'Lefutott' AS megmi FROM DUAL WHERE NULL LIKE NULL;

SELECT 'Lefutott' AS megmi FROM DUAL WHERE '' LIKE '%';

SELECT 'Lefutott' AS megmi FROM DUAL WHERE '' IS NULL;

SELECT 'Lefutott' AS megmi FROM DUAL WHERE NULL IS '';

1. Kérdezzük le a MINTA séma orszagadatok táblájából azokat az országokat, akiknek az autójelük nincs megadva!
2. Kérdezzük le a MINTA séma orszagadatok táblájából azokat az országokat, akiknek az ünnepnapjuk nincs megadva, de az autójelük igen!

\_\_\_\_\_\_\_\_\_

**Feltételes függvények és UNION**

A feltételes függvények nagyban segítik az SQL nyelven leírt mondatok hatékonyságát. Ezekre amúgy nincs szükség, enélkül is meg lehet oldani a feladatokat, pl. UNION alkalmazásával. A segédfüggvényekkel sokkal tömörebb kódot érhetünk el, amelyben kevesebb a hiba, és gyorsabban elkészülünk, azaz hatékonyak vagyunk. Csak megmutatás szintjén, hogy pl. az NVL-re sincs szükség álljon az alábbi példa:

SELECT NVL(attrib, 'valami') FROM tabla;

--- ekvivalens

SELECT 'valami' FROM tabla WHERE attrib IS NULL

UNION

SELECT attrib FROM tabla WHERE attrib IS NOT NULL;

Természetesen, minden alábbira is hasonló megoldás adható.

* NVL függvény, a null érték helyett megadhatunk más (azonos adattípusú!) helyettesítőt. – Ezt kell tudni fejből is, a többit csak megnézzük, dokumentációból találják meg szükség esetén.
* DECODE a legáltalánosabb és legrégibb feltételes függvény, de elég fapados – lényegében egy C/C++/C# világból ismert switch-case eljárás SQL függvényként. A paraméterei DECODE( attribútum, érték1, válasz1, [érték2, válasz2, ... ], egyébválasz ). Az egyéb válasz kötelező, így a DECODE alapértelmezésben legalább 4 paraméterrel bír (és mindig páros számú paramétere van). Például:

SELECT DECODE( 1 + MOD(TO\_CHAR(SYSDATE,'J'), 7),

1, 'Hétfő',

2, 'Kedd',

3, 'Szerda',

4, 'Csütörtök',

5, 'Péntek',

6, 'Szombat',

'Vasárnap' ) AS megmi FROM DUAL;

* CASE egy kicsit többet, de legalábbis másképp tudja ugyanezt. Valakinek ez olvasható. Kétféle módja van: DECODE és összehasonlítási üzemmódja is, aminek a szintaxisa eltér az előbbitől egy picit. Előbbi esetre példa:

SELECT CASE 1 + MOD(TO\_CHAR(SYSDATE,'J'), 7)

WHEN 1 THEN 'Hétfő'

WHEN 2 THEN 'Kedd'

WHEN 3 THEN 'Szerda'

WHEN 4 THEN 'Csütörtök'

WHEN 5 THEN 'Péntek'

WHEN 6 THEN 'Szombat

ELSE 'Vasárnap' END AS megmi FROM DUAL;

* A második esetben a CASE után nem szerepelhet jellemző, attribútum, érték, kizárólag a WHEN után állhat logikai kifejezés:

SELECT CASE WHEN 1 + MOD(TO\_CHAR(SYSDATE,'J'), 7) < 6

THEN 'Hétköznap'

ELSE 'Hétvége' END AS megmi FROM inventories;

* A COALESCE egy NVL-klón és kiterjesztés. Annyival több az NVL függvénynél, hogy tetszőlegesen hosszú listát adhatunk meg neki, ami néha jól jön, de nem prioritás, hogy ezt tudjuk. Itt is fontos, hogy az adattípusoknak stimmelni kell!

SELECT COALESCE(NULL, NULL, NULL, 'végre valami' ) AS megmi

FROM DUAL;

* NULLIF egy speciális függvény, ritkán használt, de jól jöhet: „NULLIF( a, b )“ megegyezik pontosan a „CASE WHEN a = b THEN NULL ELSE a“ kifejezéssel.

A feltételes függvények helyettesíthetőek UNION-nal. Az UNION, MINUS és INTERSECT műveletek csak úniókompatibilis relációk között működnek (azonos oszlopszám és sorrendben azonos adattípus). Az oszlopneveket az először megadott táblból fogja venni. Mindegyik halmaz szemantikájú (azaz az eredeti relációkban lévő duplikátumokat is megszüntetik). Egyedül az UNION-nál van lehetőség az ALL szó használatával zsákszemantikát alkalmazni. Ténylegesen a feltételes függvények helyett az UNION ALL használható.

**Feladatok**

1. Fejtsük meg a lekérdezéseket!

SELECT nvl('2', 'Na, ez mi?') AS megmi FROM dual;

SELECT nvl(NULL, 'Na, ez mi?') AS megmi FROM dual;

SELECT nvl('', 'Na, ez mi?') AS megmi FROM dual;

1. Kérdezzük le a MINTA séma CÉGEK táblájából a cégek üzleti évének végét évszám nélkül! Akiknél nincs megadva, ott december 31-et írjunk ki! Rendezzük az eredményt az üzleti év vége szerinti sorrendbe! Elhagyhatnánk-e a 2. megoldásból az ALL-t?
2. Fejtsük meg az alábbi lekérdezéseket! (Irogassuk át őket menet közben, hogy megértsük a működésüket!)

SELECT COALESCE( NULL,NULL, 500 ) AS megmi FROM dual;

SELECT to\_char( SYSDATE, 'D' ) AS megmi FROM dual;

--DECODE függvény: mindig páros számú argumentum, mert az egyéb megadása a végén kötelező

SELECT decode(to\_char( SYSDATE, 'D' ) ,

1, 'Hétfő',

2, 'Kedd',

3, 'Szerda',

4, 'Csütörtök',

5, 'Péntek',

6, 'Szombat',

'Vasárnap' ) AS megmi FROM dual;

--CASE WHEN: feltételhez kötött elágazás

SELECT CASE WHEN to\_char( SYSDATE, 'D' ) < 6

THEN 'Hétköznap'

ELSE 'Hétvége' END AS megmi FROM dual;

--ugyanez decode-dal

SELECT decode (to\_char( SYSDATE, 'D' ),

6, 'Hétvége',

7, 'Hétvége',

'Hétköznap') AS megmi FROM dual;

-- „NULLIF( a, b )“ megegyezik pontosan a „CASE WHEN a = b THEN NULL ELSE a“ kifejezéssel.

SELECT nullif( 'valami', 'valami' ) AS megmi FROM dual;

SELECT nullif( 'valami', 'más' ) AS megmi FROM dual;

**DML utasítások, tranzakciók**

Adatbevitel, módosítás, törlés: egyszerűbb variációk. Lehet NULL-os és feltételes példa is.

Commit-rollback használata. (Sajnos a tranzakciók láthatósága és a SAVEPOINT kipróbálása nem fog beleférni ebbe az órába. A SET AUTOCOMMIT utasítás is majd csak később lesz.)

Importáláskor már láttuk az INSERT INTO utasítást, és a múlt heti előadáson is volt róla szó. Szintén az előadáson részletesen volt szó az UPDATE és DELETE utasításokról is és a COMMIT ROLLBACK utasításokról is. Ezért merem beletenni ebbe az órába. Van benne egy pici create table ismétlés és kiegészítés is. A default eddig csak egy gyakorló feladatban volt, elsikkadhatott. Fontos, hogy ha default is van meg constraint is, akkor a default van előbb.

**Feladatok**

1. Hozzuk létre az UGYFEL táblát az alábbiak szerint!  
   - UGYFELID: 3 jegyű azonosító szám, a tábla elsődleges kulcsa  
   - NEV: az ügyfél teljes neve, kötelező megadni  
   - NEM: egy karakteres szöveg, F vagy N  
   - MIKORTOL: az első szerződéskötés dátuma , alapértelmezetten az aktuális dátum idő nélkül  
   - VIP: egy számjegy, csak 1 és 0 lehet, alapértelmezetten 0.
2. Vigyünk be a táblába egy személyt, 0-s kóddal úgy, hogy minden adatot megadunk! Figyeljünk az adattípusokra!

INSERT INTO ugyfel  
VALUES (  
 0 --szám, aposztrófok nélkül!  
, 'Bőrönd Ödön' --szöveg, aposztrófok között  
, 'F' –szöveg  
, to\_date('10.05.01','YY.MM.DD') –dátum  
, 1 –szám  
);

1. Nézzünk meg a további adatbeviteleket! (Kérdezzük is le, hogy mi lett a vége! Az lesz a tanulság, hogy ha explicit bevisszük a NULL-t akkor NULL lesz, ha viszont kihagyjuk az adott oszlopokat, akkor alapértelmezett lesz.)

INSERT INTO ugyfel  
VALUES (  
 100  
, 'Kiss Piroska'  
, null  
, null  
, null  
);  
  
select \* from ugyfel;  
  
INSERT INTO ugyfel (ugyfelid, nev)  
VALUES (  
 200  
, 'Fekete Péter'  
);  
  
select \* from ugyfel;

1. Véglegesítsük az adatbevitelt a COMMIT utasítással!
2. Írjuk be az ugyfel táblába a minta.alkalmazott tábla adatai közül az azonosítót és a nevet! A hiányzó értékeknél hagyjuk az alapértelmezettet! Véglegesítsük a bevitelt!
3. Írjuk be az ugyfel táblába a minta.dolgozo tábla adatait az alábbiak szerint:   
   - az azonosítókat toljuk el 100-zal  
   - a neveket fűzzük össze  
   - a nemeket kódoljuk át  
   - a mikortól egyezzen meg a felv\_nap értékével  
   - a VIP maradjon üresen (ne írja be az alapértelmezést!)  
   Ilyen feladatoknál mindig érdemes előbb csak a select-et megírni, lefuttatni, hogy lássuk, hogy tényleg azt kaptuk, amit elképzeltünk, majd ha már jónak ítéljük, akkor írjuk elé az elejét. (Jelen esetben töröljük a kommentelést)
4. Kérdezzük le az ügyfél tábla adatait, töröljünk minden adatot, kérdezzük le újra, majd vonjuk vissza a törlést és kérdezzük le újra!
5. Töröljük az ugyfel táblából a 100-zal osztható azonosítójú ügyfeleket! Véglegesítsük!
6. Töröljük az ugyfel táblából azokat, akiknek nem tudjuk a nemét! Véglegesítsük!
7. Módosítsuk az adatokat: mindenkinél írjuk be a VIP oszlopba a 0 értéket! Véglegesítsük!
8. Módosítsuk az adatokat: mindenkinek az azonosítójából vonjunk ki 100-at! Véglegesítsük!
9. Módosítsuk az adatokat: mindenki, aki 2010 előtt kötött szerződést, legyen VIP ügyfél! Véglegesítsük!
10. Másoljuk le az ugyfel táblánakt ugyfel2 néven! Töröljük a tartalmát visszavonhatatlanul! A TRUNCATE TABLE utasítással a tábla tartalma úgy törlődik, hogy azt nem tudjuk visszavonni, mivel a TRUNCATE TABLE nem DML hanem DDL parancs.

**Önálló feladatok**

1. Kérdezzük le a MINTA.IRODALOM táblából azokat az írókat, akiknek nincs megadva eredeti nevük!
2. Kérdezzük le a MINTA.IRODALOM táblából az írók nevét és eredeti nevét úgy, hogy akinek nincs megadva az eredeti neve, azoknál az eredeti névhez is az ismert nevüket írjuk ki!
3. Minden országhoz listázzuk ki az ünnepnapokat is! Ahol NULL érték van, ott írjuk be a „nem ismert” szöveget! Rendezzük az eredményt az ünnepnapok szerinti sorrendbe!
4. Kérdezzük le a MINTA séma CEGEK táblájából az Rt-ket úgy, hogy az Rt után legyen pont, de ha volt pont eleve, akkor ne legyen két pont utána!
5. Hozzuk létre az alábbi HALLGATO táblát!  
   HAZ: hallgató azonosítója, 6 karakteres szöveg, elsődleges kulcs  
   HVNEV: hallgató vezetékneve, kötelező  
   HKNEV: hallgató keresztneve, kötelező  
   HSZDAT: hallgató születési dátuma  
   EVF: évfolyam, 1 jegyű egész szám, 1 és 6 között lehet, alapértelmezetten 1
6. Vigyünk be sql utasítással két rekordot, az egyik hallgatónak ne legyen évfolyama, a másiknak engedjük az alapértelmezett évfolyamot beírni!
7. A Moodle-ben található Excel táblázatból hozzuk létre az adatok beírására alkalmas sql utasításokat! Másoljuk be őket és futtassuk le! Véglegesítsük az adatbevitelt! (Csak az első utasítást másoljuk be a Moodle-be)
8. Töröljük a hallgatókat! Vonjuk vissza! Töröljük csak a 6. éves hallgatókat! Véglegesítsük!
9. Töröljük azokat a hallgatókat, akik 2016. szeptember 1-n még nem voltak 18 évesek! Véglegesítsük!
10. Módosítsuk mindenkinek az évfolyamát eggyel nagyobbra! Vonjuk vissza!
11. Egy udasítással módosítsuk a hallgatók évfolyamát az alábbiak szerint: a null-ból legyen 1, mindenkinek az évfolyama legyen eggyel nagyobb, az 5-ből pedig legyen null!