Néhány apró Oracle specialitás, amikről érdemes egy-két szót ejteni.

A DUAL tabla

Van a SYS felhasználó tulajdonában egy DUAL nevű tábla, aminek egyetlen oszlopa van: `DUMMY VARCHAR2(1)` és a táblának egyetlen sora van. Mivel minden SELECT utasításnak legalább egy táblát le kell kérdeznie (vagyis valamit be kell írni a FROM után), a fenti tábla hasznos olyan beépített függvények hívása esetén, amelyeknél valójában nincs szükségünk táblára. Ennek segítségével lehet például lekérdezni a rendszeridőt, vagy az aktuális felhasználó azonosítóját.

```
SELECT SYSDATE FROM DUAL;
SELECT USER    FROM DUAL;
```

Ezt a táblát sokszor az oracle eszközök is használják, ezért ne változtassunk rajta.

Az oracle-ben ha valamilyen rendszeradatot szeretnénk lekérdezni akkor mindig ezt a DUAL táblát kell használnunk. Rendszer információkat a következő SQL függvényekkel kérhetünk:

```
SYSDATE, USER, UID, USERENV … stb.
```

Összesítő függvények

egyszeres mélységben egymásba ágyazhatók.

```
SELECT MAX(AVG(fizetes)) FROM dolgozo GROUP BY oazon;
```

Felső-N elemzés

Ha egy lekérdezésnek nem az összes sorát szeretnénk megkapni, hanem csak az első néhány sort, akkor használhatjuk a ROWNUM pszeudooszlopot. A ROWNUM-ot a lekérdezés eredményének előállítása közben rendeli az adatbázis-kezelő a sorokhoz, mint egy sorszámot.
Lekérdezhetjük egy megadott rendezés szerinti első néhány sort, ha a rendezést egy alkérdésbe tesszük, és a külső SELECT-ben szűrünk ROWNUM szerint.

```
SELECT fizetes FROM (SELECT dnev, fizetes FROM dolgozo ORDER BY fizetes)
WHERE ROWNUM <= 5;
```

Zárójelben kell megadni a rendezést, az alábbi nem a legkisebb fizetéseket adja vissza:

```
SELECT dnev, fizetes FROM dolgozo WHERE ROWNUM  <= 5 ORDER BY fizetes;
```

Az Oracle 12-es verziójától használható az alábbi szintaxis is:

```
SELECT dnev, fizetes FROM dolgozo ORDER BY fizetes FETCH FIRST 5 ROWS ONLY;
```

Ez utóbbinak a részletesebb szintaxisa az alábbi (lásd még a dokumentációban):

```
[ OFFSET offset { ROW | ROWS } ]
[ FETCH { FIRST | NEXT } [ { rowcount | percent PERCENT } ]
    { ROW | ROWS } { ONLY | WITH TIES } ]
```

A ROW és ROWS valamint a FIRST és NEXT kulcsszavak felcserélhetők.
Ha OFFSET részt nem adunk meg, a rendszer azt 0-nak tekinti.
Negatív offset, rowcount vagy percent értéket 0-nak tekinti.
Ha az offset, rowcount vagy percent értéke NULL, nem ad vissza egy sort sem.
Ha az offset, rowcount vagy percent értéke nem egész, csonkolja.
WITH TIES megadása esetén, a holtversenyben lévőket is visszaadja.

```
SELECT fizetes FROM dolgozo ORDER BY fizetes FETCH FIRST 4 ROWS WITH TIES;
```

UNIQUE
A DISTINCT szinonimájaként használható, a két kulcsszó teljesen egyenértékű.

Külső join művelet: (+)
Az Oracle elsőként vezette be a külső join műveletet, eredetileg egy másfajta jelöléssel, mégpedig egy zárójelbe tett „+” jellel. A „+” jelet fel lehet úgy is fogni, hogy a vele megjelölt táblát a rendszer kiegészíti egy plusz sorral, nevezzük ezt „Joker” sornak. Azért tekinthetjük ezt Jokernek, mert tetszőleges feltételt megadva a WHERE kulcsszó után, ha a táblának egyetlen sora sem elégíti ki ezt a feltételt, akkor a Joker sort úgy fogja tekinteni a rendszer, mint, ami kielégíti a feltételt, és beleveszi a lekérdezés eredményébe. E Joker sor az összes oszlopában NULL értékkel fog megjelenni a végeredményben. Ha egy join művelet esetén egy tábla egy sorának nem lenne párja a másik táblában, de a másik táblát kiegészítjük ezzel a Joker sorral, akkor ezután már minden sornak lesz párja, ha más nem, akkor a Joker sor.

A (+) csak a WHERE kulcsszó után használható. A műveletet megelőző oszlop lesz a külső join oszlop. Vagyis ezen oszlop értékei helyett fog NULL értékeket szerepeltetni az Oracle a lekérdezésben, amennyiben a másik táblabeli sornak nincs párja.
pl.
```
SELECT dnev, onev FROM dolgozo, osztaly
WHERE osztaly.oazon = dolgozo.oazon (+);
```

Ez vissza fogja adni azt az osztályt is, amelyiken nem dolgozik senki.
Ha a két táblát összekapcsoló JOIN feltétel több oszlopot is tartalmaz, akkor a "(+)" jelet minden oszlop után ki kell tenni.
A "(+)" jelet csak oszlopra lehet alkalmazni nem pedig tetszőleges kifejezésre, de egy kifejezésben szerepelhet "(+)" jellel megjelölt oszlop.
"(+)" jellel megjelölt oszlop nem szerepelhet IN összehasonlításban (kivéve ha literálok vannak a zárójelben), nem hasonlítható össze SUBSELECT-tel, és a rá vonatkozó feltételt nem lehet OR-ral összekapcsolni más feltétellel.
Egy SELECT utasításban egy táblát csak egy másikkal lehet külső join művelet segítségével összekötni. Nem lehet egyszerre mindkét táblára külső joint alkalmazni.

Nem megengedett utasítások az alábbiak!!!

```
SELECT dnev, onev FROM dolgozo, osztaly
WHERE osztaly.oazon = dolgozo.oazon (+) OR dolgozo.oazon=10

SELECT dnev FROM dolgozo
WHERE dolgozo.oazon (+) IN (SELECT oazon FROM osztaly);
```

De megengedett az alábbi utasítás:

```
SELECT dnev FROM dolgozo WHERE oazon (+) IN (10, 20);
```

Ennek az oka, hogy az oazon IN (10, 20)-t a következővé alakítja át a rendszer:
```
Oazon = 10 OR oazon = 20
```

JOIN műveletek
(Az OUTER kulcsszó opcionális)

SELECT emp.* FROM emp **JOIN** dept **ON** emp.deptno <= dept.deptno
SELECT emp.* FROM emp **CROSS JOIN** dept
SELECT emp.* FROM emp **NATURAL JOIN** dept -- azonos nevű oszlopok alapján
SELECT emp.* FROM emp **JOIN** dept **USING** (deptno) -- a megadott oszlopok alapján
SELECT emp.* FROM emp **NATURAL LEFT OUTER JOIN** dept

OUTER JOIN (sőt FULL OUTER JOIN is)

```
SELECT o.oazon, dnev, onev
FROM dolgozo d RIGHT OUTER JOIN osztaly o ON d.oazon = o.oazon;
```

A külső join-olt táblának fogja a rendszer az összes sorát visszaadni, vagyis mintha a másik táblát egészítenénk ki „Joker” sorral.

A fentivel ekvivalens az alábbi régi módszer:
```
SELECT o.oazon, dnev, onev FROM dolgozo d, osztaly o
WHERE d.oazon (+) = o.oazon;
```

Példa:

Van két táblánk A(O1, O2) és B(O1, O2). A táblák tartalma a következő:

A

2	B
4	D

B

3	C
4	D

```
create table a(o1 integer, o2 varchar2(20));
create table b(o1 integer, o2 varchar2(20));
insert into a values(2, 'B');
insert into a values(4, 'D');
insert into b values(3, 'C');
insert into b values(4, 'D');
```

2 ekvivalens lekérdezés:
```
SELECT * FROM A LEFT OUTER JOIN B ON A.O1 = B.O1;
SELECT * FROM A, B WHERE A.O1 = B.O1 (+);
```
Az eredmény:

4	D	4	D
2	B	**NULL**	**NULL**

Kicsit módosított 2 lekérdezés:
```
SELECT * FROM A LEFT OUTER JOIN B ON (A.O1 = B.O1 AND B.O1 > 5);
SELECT * FROM A, B WHERE A.O1 = B.O1 (+) AND B.O1 (+) > 5;
```
Az eredmény:

4	D	**NULL**	**NULL**
2	B	**NULL**	**NULL**

Tovább módosított 2 lekérdezés:

```
SELECT * FROM A LEFT OUTER JOIN B ON A.O1 = B.O1 WHERE B.O1 > 5;
SELECT * FROM A, B WHERE A.O1 = B.O1 (+) AND B.O1 > 5;
```

Az eredmény:

```
No rows selected
```

A fenti példák azt mutatják, hogy az adatbáziskezelő másképp értékeli ki a join feltételt, mint a kiválasztási feltételt **külső join esetén**. Ha egy sor a join feltételnek nem felel meg, attól még külső join esetén megtartja a rendszer a végeredményben. Ha egy kiválasztási feltételnek nem felel meg, akkor viszont nem tartja meg.

Egy lekérdezésen belül több táblára vonatkozóan is megadható külső join, pl. az alábbi lekérdezés visszaadja azokat a vevőket is, akik semmit sem rendeltek és azokat a rendeléseket is, amiknek meg egyetlen tételsora sincs. Az alábbi csupán egy példa, ilyen nevű tábláink jelenleg nincsenek.

```
SELECT * FROM vevo, rendeles, tetelek
WHERE vevo.vkod = rendeles.vkod (+)
AND rendeles.rkod = tetelek.rkod (+);
```

Érdemes megnézni az alábbi két ekvivalens lekérdezést is.

```
SELECT o.oazon, dnev, onev, fizetes
FROM dolgozo d, osztaly o
WHERE d.oazon (+) = o.oazon AND o.oazon > 20 AND fizetes(+) > 1500;

SELECT o.oazon, dnev, onev, fizetes
FROM dolgozo d RIGHT OUTER JOIN osztaly o
ON d.oazon = o.oazon AND fizetes > 1500
WHERE o.oazon > 20;
```

Vagyis a fentieket összefoglalva, a WHERE utáni feltételek közül azokat, amelyekben szerepel a (+) jel, illetve azokat, amelyek a JOIN ON kulcsszó után szerepelnek JOIN feltételnek tekinti a rendszer, a többi feltételt pedig hagyományos kiválasztási feltételként. Lásd még az alábbi két ekvivalens példát.

```
SELECT o.oazon, dnev FROM dolgozo d, osztaly o
WHERE d.oazon (+) = o.oazon AND dnev (+) LIKE '%A%'
AND o.oazon >= 30;

SELECT o.oazon, dnev FROM dolgozo d RIGHT OUTER JOIN osztaly o
ON (d.oazon = o.oazon AND dnev LIKE '%A%')
WHERE o.oazon >= 30
```

Az alábbi példa azt mutatja meg, hogy hogyan lehet függvényben használni a (+) jelet. Az utána következő a vele ekvivalens másik szintaxist mutatja.

```
SELECT o.oazon, dnev FROM dolgozo d, osztaly o
WHERE d.oazon (+) = o.oazon AND INSTR(dnev (+),'A') > 0
AND o.oazon >= 30 AND o.oazon <= 40;
```

30	ALLEN
30	WARD
30	MARTIN
30	BLAKE
30	JAMES
40	**NULL**

```
SELECT o.oazon, dnev FROM dolgozo d RIGHT OUTER JOIN osztaly o
ON (d.oazon = o.oazon AND INSTR(dnev, 'A') > 0)
WHERE o.oazon >= 30 AND o.oazon <= 40;
```

Az alábbit viszont már nem lehetne a (+)-os szintaxissal kifejezni

```
SELECT c.customer_id, o.order_id
FROM oe.customers c FULL OUTER JOIN oe.orders o
ON c.customer_id = o.customer_id
WHERE c.customer_id BETWEEN 113 and 116
ORDER BY o.order_id  NULLS FIRST;
```

Próbáljuk ki az alábbi lekérdezésben a (+) jelet áttenni a másik oldalra, illetve a LEFT|RIGHT|FULL kulcsszavakat.

```
create table outer1(o1 number, o2 VARCHAR2(10));
insert into outer1 values(1, 'Egy');
insert into outer1 values(2, 'Ketto');

create table outer2(o1 number, o2 VARCHAR2(10));
insert into outer2 values(1, 'Egy');
insert into outer2 values(3, 'Harom');

select * from outer1 t1, outer2 t2 where t1.o1 (+) = t2.o1;
```

1	Egy	1	Egy
NULL	**NULL**	3	Harom

```
select * from outer1 full outer join outer2 on outer1.o1 = outer2.o1;
```

1	Egy	1	Egy
NULL	**NULL**	3	Harom
2	Ketto	**NULL**	**NULL**