II. ZH szintaxis összefoglaló

Csoportfüggvények

```
AVG, COUNT, SUM, MIN, MAX
```

pl.

```
SELECT AVG(salary) FROM EMPLOYEES;
SELECT COUNT(*) FROM DEPARTMENTS;
```

Csoportosítás

Mindig két részből áll:

1. Valami, amit kiszámolunk a csoportra - pl. az átlagot (lásd a fenti függvényeket)
2. Mi az, ami alapján kosarakba gyűjtjük a dolgokat

pl.

```
SELECT department_id, MAX(salary)
FROM EMPLOYEES
GROUP BY department_id;
```

Having

```
SELECT
  MAX(SALARY),
  DEPARTMENT_ID
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID
HAVING MAX(SALARY) > 10000;
```

Allekérdezések

Gyakorlatilag annyi, hogy amit egy VIEW-ba tettem volna, azt beleírhatom zárójelbe is inline. Fujj.

```sql
SELECT
    e.department_id,
    last_name,
    legkisebb
FROM
    employees e,
    -- allekérdezés innentől:
    (
        SELECT
            department_id,
            MIN(salary) legkisebb
        FROM employees
        GROUP BY department_id
    ) min
WHERE
    e.salary=min.legkisebb
        AND
    e.department_id=min.department_id;
```

IN, ANY/SOME, ALL, EXISTS

IN

```sql
SELECT
    department_name Részlegnév,
    city Város
FROM
    departments
        NATURAL JOIN
    locations
WHERE
    department_id
        IN
    (
        SELECT department_id
            FROM employees NATURAL JOIN jobs
            WHERE job_title = 'Programmer'
    )
ORDER BY Részlegnév;
```

NOT IN ("Antijoin")

```
SELECT *
FROM employees
WHERE
    department_id
        NOT IN
    (
        SELECT department_id
        FROM departments
        WHERE location_id = 1700
    )
ORDER BY last_name;
```

ANY

```
SELECT *
FROM employees
WHERE
    salary =
        ANY(
            SELECT salary FROM employees
            WHERE department_id = 30
        );
```

IN és ANY viszonya

- Az `IN(query)` az ugyanaz, mint az `= ANY(query)`

EXISTS

```
WHERE EXISTS ( subquery );
```

Ahol a `subquery` legalább egy értékkel visszatér, itt igaz lesz.

```
-- Azon részlegek listázása, ahol van 15.000 dollárnál magasabb fizetésű dolgozó

SELECT *
FROM departments d
WHERE
        EXISTS (
        SELECT *
        FROM employees e
        WHERE
                        d.department_id=e.department_id
                                AND
                e.salary > 15000
    )
ORDER BY
        department_name;
```

Halmazműveletek

```
UNION, INTERSECT, MINUS
```

```
SELECT region_id FROM regions
        UNION
SELECT region_id FROM countries;
```

ROWNUM / LIMIT

```
SELECT * FROM
(
    SELECT *
    FROM employees
        ORDER BY salary DESC
)
WHERE ROWNUM <= 5;
```

```
FETCH FIRST n ROWS ONLY
--vagy
OFFSET n ROWS
[FETCH NEXT m ROWS ONLY]
```

PL/SQL

```
SET serveroutput ON
BEGIN
        dbms_output.put_line('Hello World!');
END;
```

```
ACCEPT nev PROMPT 'Kérem a neved:'
BEGIN
        dbms_output.put_line('Hello ' || '&nev');
END;
```

```
BEGIN
        dbms_output.put_line('Hello '||'&n');
END;
```

```
ACCEPT x PROMPT 'Kérem a számot:'
DECLARE
        szam NUMBER;
BEGIN
        szam := &x;
        dbms_output.put_line(szam ||
' négyzete: ' ||szam*szam);
END;
```

```
IF feltétel THEN
                ezt csináljuk
ELSIF másikfeltétel THEN
                mást csinálunk
ELSE
                 különben ezt csináljuk
END IF;
```

```
DECLARE
        tol NUMBER :=1;
        ig  NUMBER :=10;
BEGIN
        LOOP
                dbms_output.put_line(tol);
                EXIT WHEN tol=ig;
                tol:=tol+1;
        END LOOP;
END;
```

```
WHILE feltétel
LOOP
        ....
END LOOP;
```

```
FOR i IN tól..ig
LOOP
        ....
END LOOP
```

```
DECLARE
        tol NUMBER;
        ig  NUMBER;
BEGIN
        tol:=&mettol;
        ig:=&meddig;
        FOR i in tol..ig
        LOOP
                IF MOD(i,2)=1 THEN
                        dbms_output.put(i || ', ');
                END IF;
```

```
        END LOOP;
        dbms_output.put_line('');
END;
```

```
DECLARE
        egysor employees%ROWTYPE;
BEGIN
  FOR egysor IN (SELECT * FROM employees)
  LOOP
    dbms_output.put('Név: ' || egysor.last_name);
    dbms_output.put_line(',  Fizetés: ' || egysor.salary);
  END LOOP;
END;
```

```
DECLARE
  CURSOR kurzor IS select * from employees;
  rekord employees%ROWTYPE;
BEGIN
  FOR rekord IN kurzor
  LOOP
    IF rekord.salary BETWEEN 10000 AND 15000
    THEN
                dbms_output.put_line(rekord.last_name|| ',  ' || rekord.salary);
    END IF;
  END LOOP;
END;
```

```
DECLARE
  CURSOR kurzor IS SELECT * FROM employees FOR UPDATE OF salary NOWAIT;
  rekord employees%ROWTYPE;
  fizetes NUMBER;
BEGIN
        FOR rekord IN kurzor
        LOOP
                fizetes := rekord.salary * 1.2;

                UPDATE employees
                SET salary = fizetes
                WHERE CURRENT OF kurzor;
        END LOOP;
END;
```

```
CREATE PROCEDURE osszead (a NUMBER, b NUMBER) IS
BEGIN
        dbms_output.put_line('Összegük: '|| (a+b));
END;
```

```
CREATE OR REPLACE FUNCTION maxfizu
(dn NUMBER) RETURN NUMBER IS
m NUMBER;
BEGIN
SELECT MAX(salary) INTO m
FROM employees WHERE department_id=dn;
RETURN m;
END;
```

```
CREATE OR REPLACE TRIGGER  {név}
[BEFORE | AFTER | INSTEAD OF]
{esemény} [OR {esemény} …]
ON {tábla}
[FOR EACH ROW [WHEN {feltétel}]]
[DECLARE {változók}]
BEGIN
        {utasítások}
[EXCEPTION …]
END;

--pl.

CREATE TRIGGER Jelzo
AFTER INSERT ON departments
BEGIN
        dbms_output.put_line('Új részleg beszúrva!');
END;
```

```
CREATE OR REPLACE TRIGGER Naplozo
BEFORE DELETE OR INSERT ON employees
FOR EACH ROW
BEGIN
  IF INSERTING THEN
                dbms_output.put_line('Uj: '||   :NEW.last_name);
  ELSIF DELETING THEN
                dbms_output.put_line('Töröl: '||
                :OLD.last_name);
        END IF;
END;
```

User management

```
ALTER USER géza
IDENTIFIED BY géza123;
```

```
ALTER USER géza
QUOTA UNLIMITED ON users;
```

```
CREATE USER <felhasználónév>
IDENTIFIED BY <jelszó>
QUOTA <mennyiség> ON <táblatér>
```

```
DROP  USER  felhasználónév  [CASCADE]
```

```
GRANT CREATE TABLE,CREATE VIEW
TO géza;

GRANT alkalmazottak TO  géza;
```

```
REVOKE CREATE TABLE
FROM géza;

REVOKE alkalmazottak
FROM géza;
```

```
INSERT, UPDATE, DELETE, SELECT
```

```
SAVEPOINT ittmentettünk;
ROLLBACK TO ittmentettünk;
ROLLBACK;
COMMIT;
```