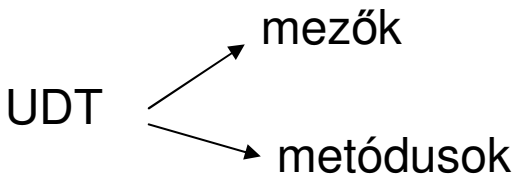
Adatbázisok MSc

7. téma

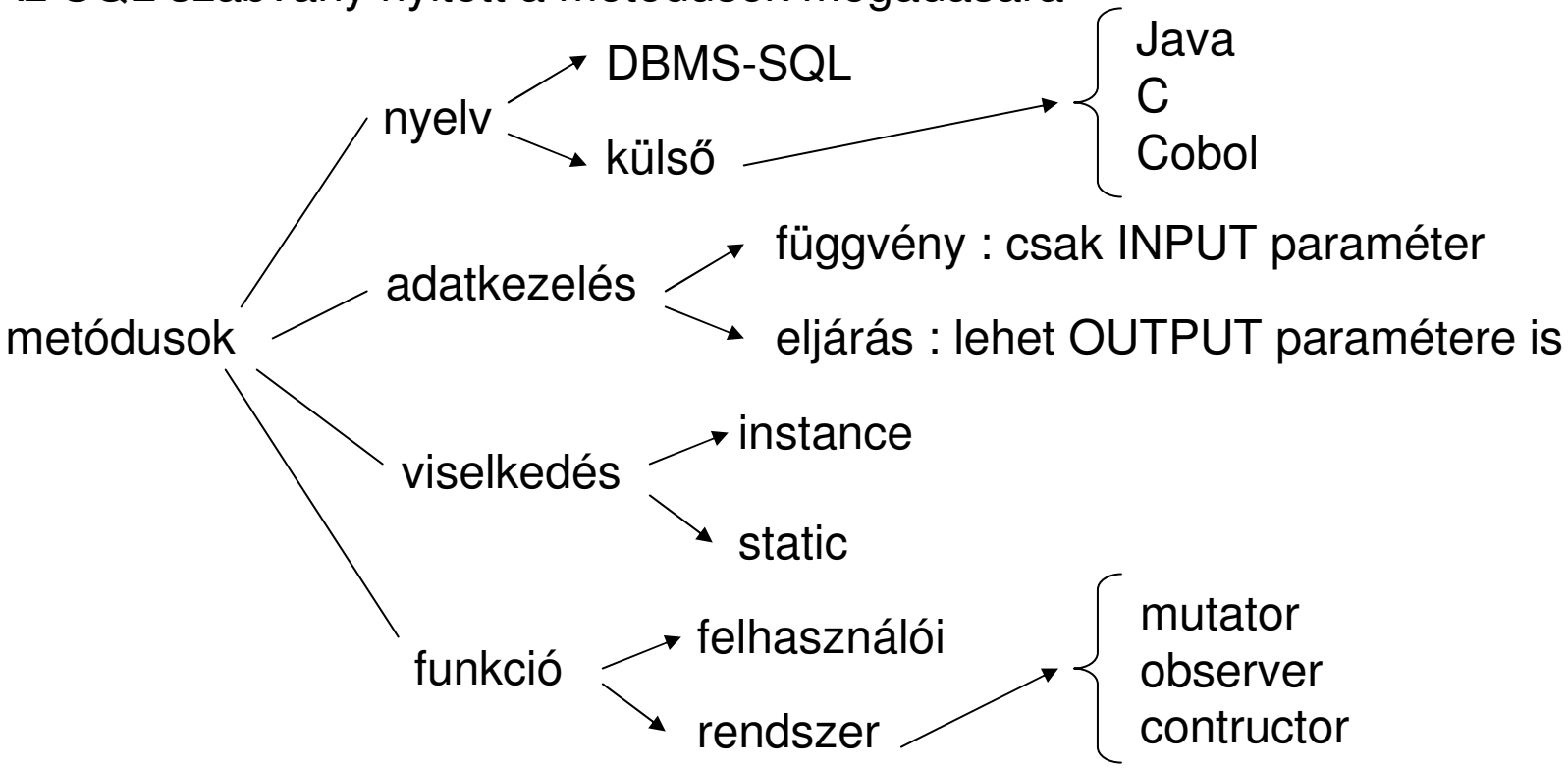
SQL OO elemei – aktív komponensek

UDT aktív elemek

Az UDT egyik fontos jellemzője az egységbezárás, mely az állapot és viselkedés együttesét jelenti



Az SQL szabvány nyitott a metódusok megadására



UDT aktív elemek

Metódusok definiálása az UDT-hez kötötten történik

metódus megadása: → interfész az UDT-ben
metódus megadása: → implementáció az UDT-n kívül

```
CREATE TYPE tnev AS (
…
)
INSTANCE | STATIC | CONSTRUCTOR METHOD mnev (paraméterlista)
  RETURNS tip;


CREATE INSTANCE | STATIC | CONSTRUCTOR METHOD
  mnev (paraméterlista)  RETURNS tip  FOR tnev
  törzs;
```

A törzsben az aktuális obejktumra a SELF kulcsszóval hivatkozhatunk.

UDT aktív elemek

```
CREATE TYPE dolgozo_t AS (
    nev CHAR(25),
    statusz CHAR(1)
)
INSTANTIABLE
REF IS SYSTEM GENERATED
INSTANCE METHOD szstatusz () RETURNS CHAR(15);


CREATE INSTANCE METHOD szstatusz() RETURNS CHAR(15)
FOR dolgozo_t
RETURN
   CASE
     WHEN SELF.statusz = 'A'  THEN 'aktiv'
     WHEN SELF.statusz = 'N'  THEN 'nyugdijas'
     WHEN SELF.statusz = 'D'  THEN 'diak'
     ELSE 'ismeretlen'
   END;
```

UDT aktív elemek

Metódusok meghívása

A metódusra a () operátorral lehet hivatkozni

```
CREATE TABLE dolgozok OF dolgozo_t (
   REF IS id SYSTEM GENERATED
)

SELECT a.nev, a.szstatusz() FROM dolgozok a
WHERE a.szstatusz() NOT LIKE ‘N%’;

SELECT a.nev, a.uzem->nev() FROM dolgozok a
WHERE a.szstatusz() NOT LIKE ‘N%’;

SELECT sysdate, dolgozo_t.letszam(13) FROM dual;
```

UDT aktív elemek

Speciális metódusok

Az SQL szabványban a objektumok adattagjai nem érhetők el közvetlenül

Minden hivatkozás az UDT observer és mutator metódusán keresztül történik

A felhasználó számára ezen metódusok transzparensek

Nagyfokú függetlenséget biztosít

x dolgozo;

z = x.nev; ⟶ z = x.nev() : observer

x.nev = ‘Peter’ ⟶ z.nev(‘Peter’) : mutator

A mutator metódus a teljes objektumot adja vissza

A metódusok nem írhatók át

UDT aktív elemek

Speciális metódusok

Konstruktor függvény

Itt a konstruktor függvény csak az adattagok kezdőértékét állítja be, de nem allokál helyet
Neve megegyezik az UDT nevével
A konstruktor értéket ad vissza és nem objektumot

```
CREATE TYPE dolgozo_t AS (
    nev CHAR(25),
    statusz CHAR(1)
)
CONSTRUCTOR METHOD dolgozo_t (n CHAR, s CHAR)
RETURNS dolgozo_t;

CREATE CONSTRUCTOR METHOD dolgozo_t (n CHAR, s CHAR)
RETURNS dolgozo_t FOR dolgozo_t
BEGIN
    SET SELF.nev = n;  SET SELF.statusz = s;
    RETURN SELF;
END
```

UDT aktív elemek

Objektum létrehozása

Új objektumot a NEW operátorral lehet létrehozni.

Obj = NEW tipus (paraméterlista)

Tipus: az UDT típusa
Paraméterlista: a konstrukrra illő szignatúra

```
CREATE CONSTRUCTOR METHOD dolgozo_t (n CHAR, s CHAR)
RETURNS dolgozo_t FOR dolgozo_t
BEGIN
    SET SELF.nev = n;  SET SELF.statusz = s;
    RETURN SELF;
END

CREATE TABLE uzemek (nev CHAR(25), fonok dolgozo_t)

UPDATE dolgozok SET fonok = NEW dolgozo_t('ZZ','A') WHERE
nev LIKE 'P%';
```

UDT aktív elemek

Objektum kezelése

Az objektum tartalma rendszerint közvetlenül nem reprezentálható

Szükség lehet az objektum különböző formátumban történő reprezentálására

```
CREATE CAST (tipus1 TO tipus2) WITH konverzios_fuggveny;
```

A konverziós rutin egy önálló adatbázis elem, nem az UDT közvetlen része

```
CREATE TYPE dolgozo_t AS (
    nev CHAR(25),
    statusz CHAR(1)
)

CREATE FUNCTION konv (be dolgozo_t) RETURNS CHAR(25)
BEGIN
  RETURN be.nev || be.szstatusz();
END;

CREATE CAST (dolgozo_t TO CHAR) WITH konv;
```

UDT aktív elemek

Objektum kezelése

Az objektum értékek összehasonlítása sem triviális feladat a szerkezet miatt

SQL megoldás: objektum állapot leképzése elemi típusra, s ott történik meg az összehasonlítás (egyenlőség, rendezés)

```
CREATE ORDERING FOR tipus ORDER FULL BY MAP
WITH FUNCTION függvény;
```

```
CREATE FUNCTION konv (be dolgozo_t) RETURNS CHAR(25)
BEGIN
  RETURN be.nev || be.szstatusz();
END;

CREATE ORDERING FOR dolgozo_t ORDER FULL BY MAP
WITH FUNCTION konv (dolgozo_t)
```

Ekkor a dolgozo_t értékeket szöveges alakjuk alapján hasonlítja össze

UDT aktív elemek

Polimorfizmus

Az SQL támogatja az szignatúra alapján történő függvény, eljárás és metódus kiválasztást: azonos név, de eltérő szignatúra

```
CREATE TYPE tip AS (

)
INSTANCE METHOD f1 (db INT) …;
INSTANCE METHOD f1 (db INT, min INT)…;
```

A származtatott típusokban felülírható az azonos szignatúrájú metódusok értelmezése is

```
CREATE TYPE altip UNDER tip AS (

)
OVERRIDE INSTANCE METHOD f1 (db INT) …;
```

A metódusok, eljárások törzsének megírása

Az üzleti logika procedurális nyelven definiált.

Alkalmazható programozási nyelvek:
- külső (pl. java, C)
- saját
- SQL/kiegészítés

Külső: alkalmazzák (pl. Postgres)
+: ismert, sok funkció
-: laza integráció,védelem

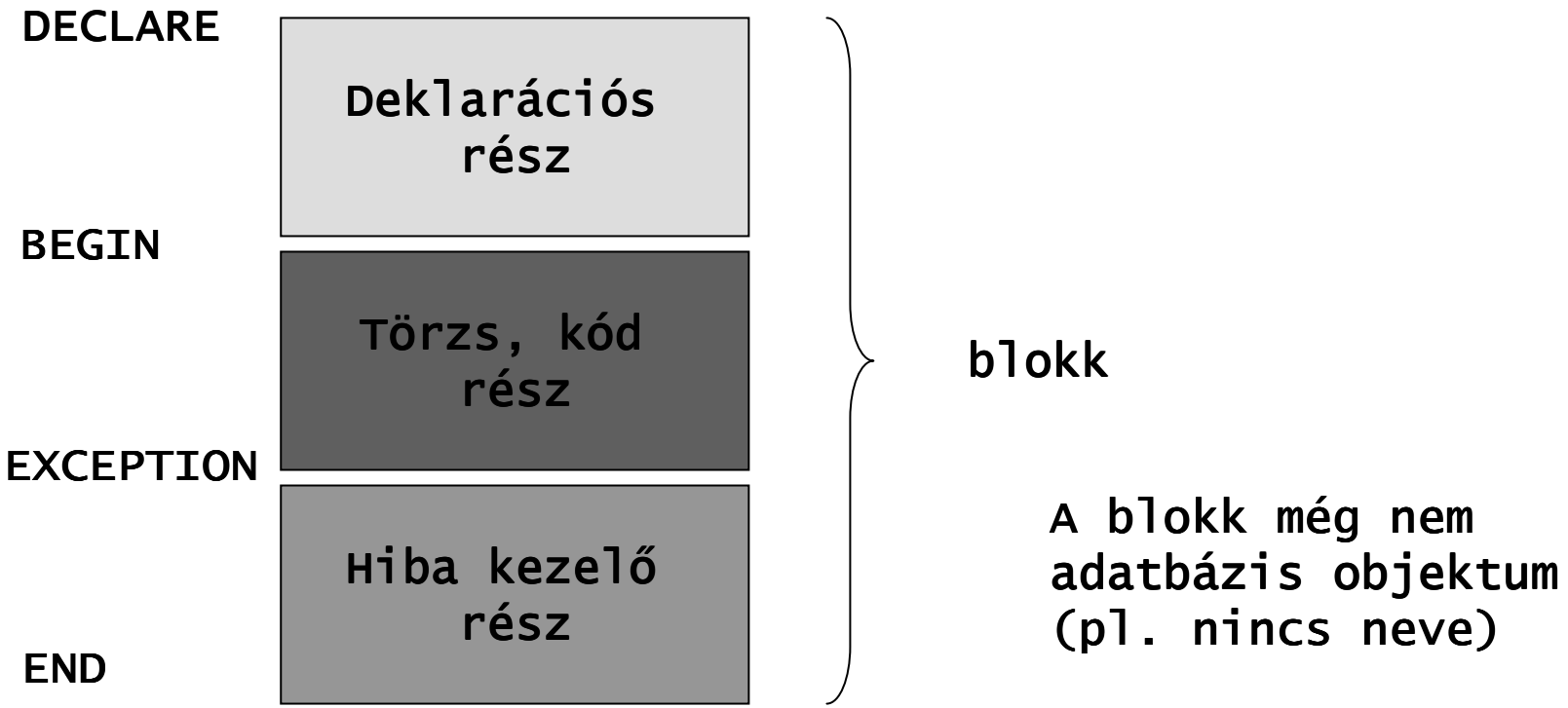
Saját: ritkán alkalmazzák (pl. VFP)
+: testre szabható
-: új ismereteket igényel, zártság

SQL kiegészítés: sokan alkalmazzák (pl. Oracle, SQLServer)
+: szoros integráció
-: sok új elem van a nyelvben, zártság

PL/SQL alapjai

PL/SQL: az Oracle SQL kiegészítése a procedurális elemeket definiálására

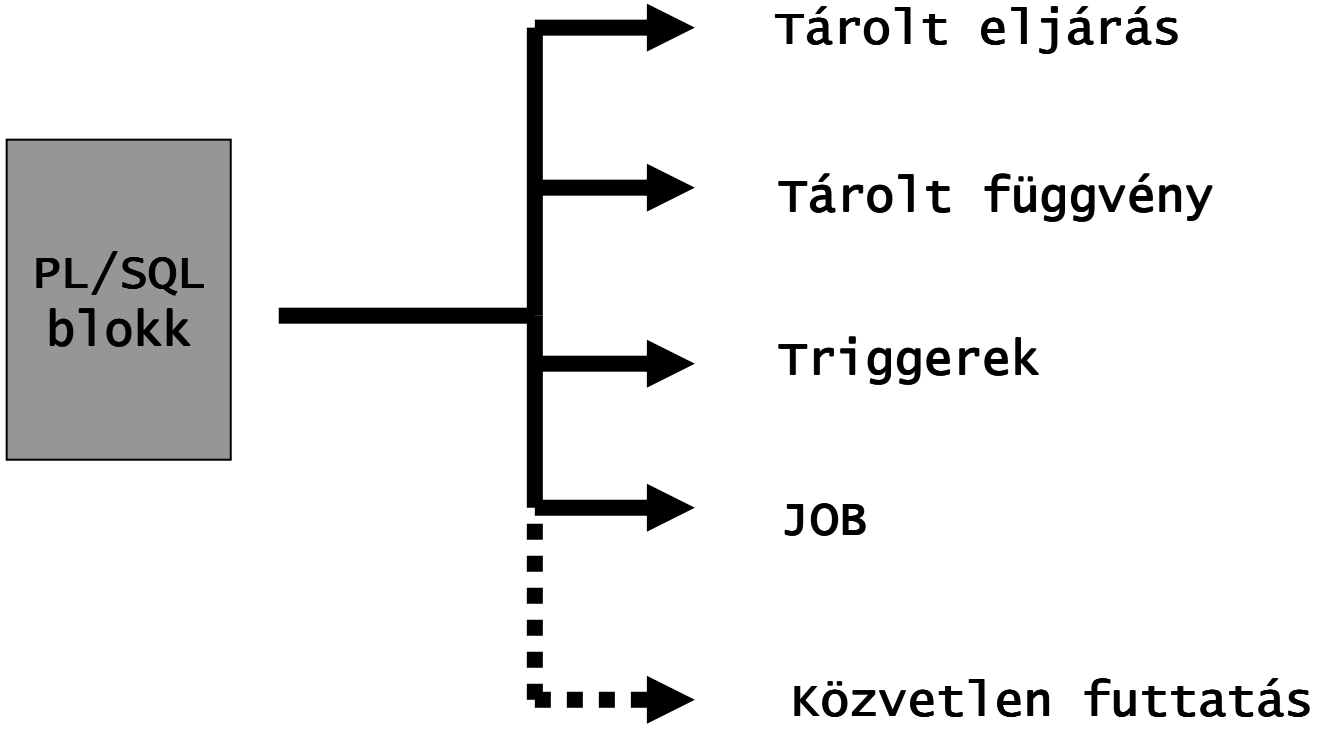
A PL/SQL kódok építő köve a PL/SQL blokk



A blokk még nem adatbázis objektum (pl. nincs neve)

PL/SQL alapjai

A PL/SQL blokk felhasználása



Tárolt eljárások, PL/SQL elemei

A tárolt eljárások DB objektumként védhető, karbantartható

```
CREATE OR REPLACE PROCEDURE
  név (pnev1 IN | OUT típus1,…   )
AS
 PL/SQL blokk
```

Paramétereknél:

IN: bemenő paraméter

OUT: értéket kap, amit kinn is látni kellene

Elindítása:

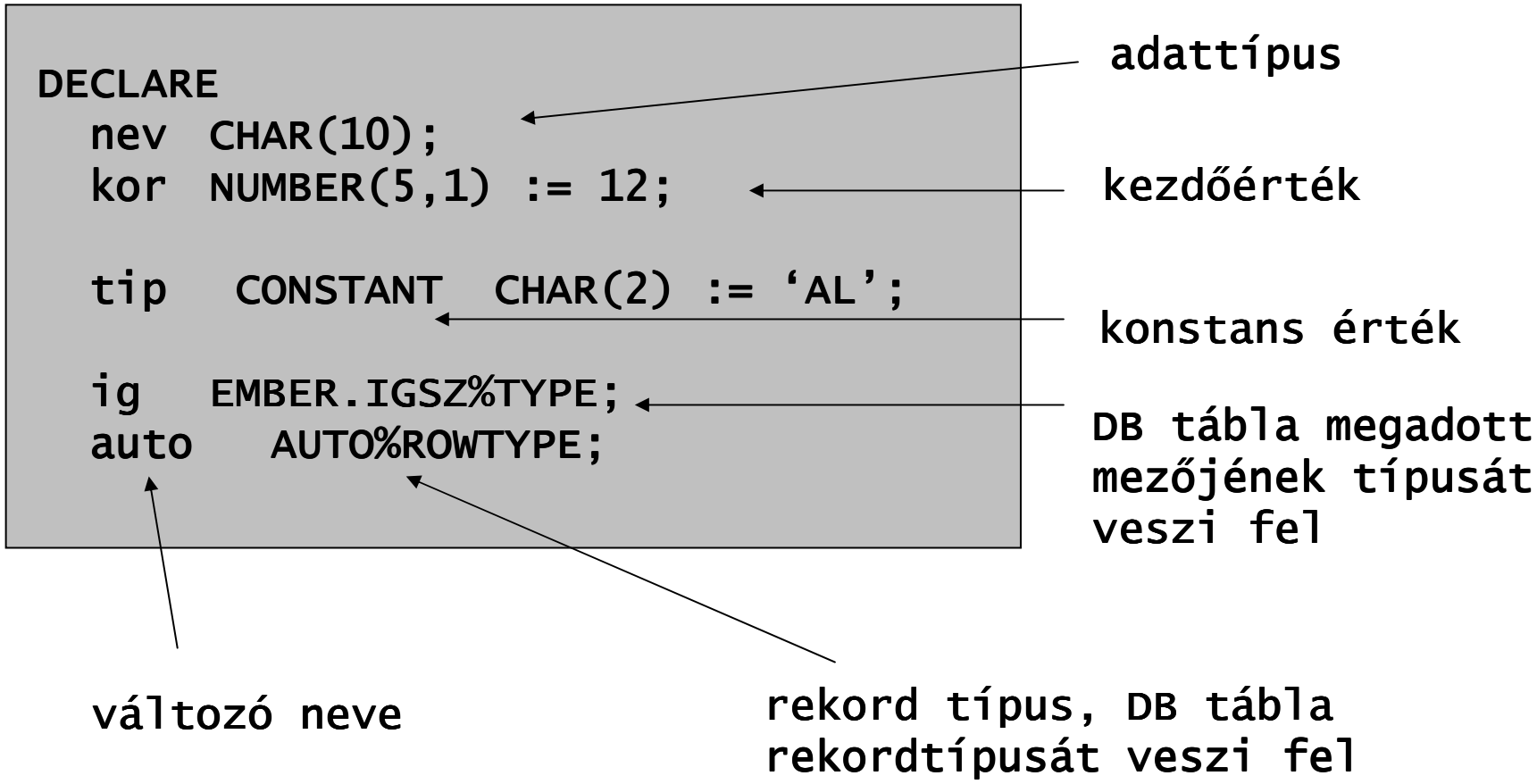
```
EXECUTE név (param1, ..)
```

Megszüntetése:

```
DROP PROCEDURE név
```

Programozási elemek

A deklarációs rész felépítése



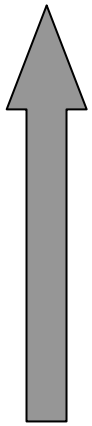
Alap adattípusok

BINARY_INTEGER: egész, bináris, +/-214783647-ig
NUMBER(n,m): valós, számjegyes
CHAR(n): szöveg
VARCHAR2(n): változó méretű szöveg
BOOLEAN: logikai
DATE: dátum, idő
RAW: bináris
LONG: hosszú szöveg(2GB)
LONG RAW: hosszú bináris(2GB)

```
declare
s1 char(5) := '12';
s2 char(5) := '12 ';
=> s1 = s2 ist TRUE
declare
s1 varchar2(5) := '12';
s2 varchar2(5) := '12 ';
=> s1 = s2 ist FALSE
```

Alap operátorok

Operátorok prioritási táblázata



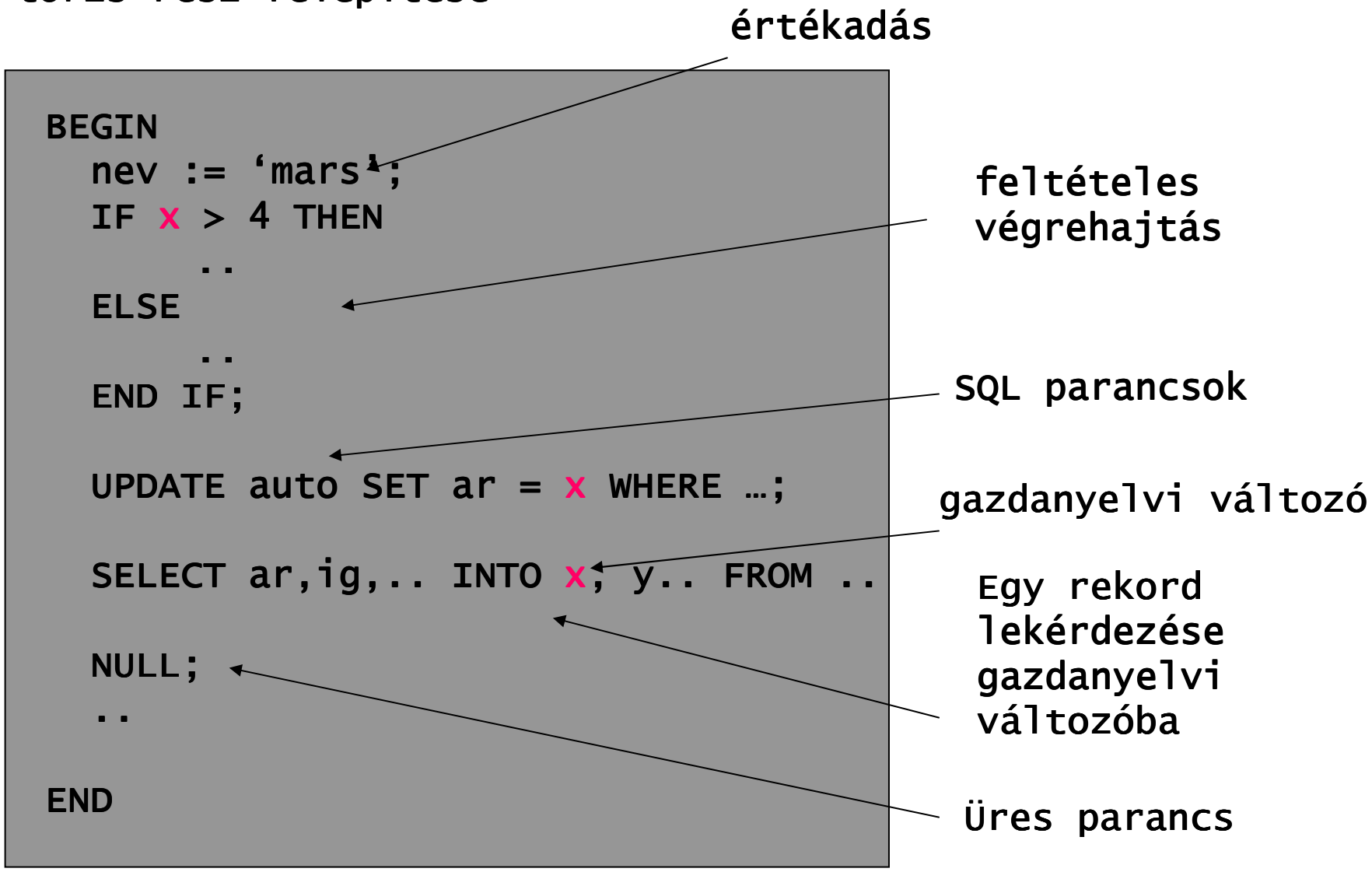
NOT, **
+, - (előjelek)
/, *
+, -, ||
=, !=, <, >, <=, >=, IS NULL,LIKE,BETWEEN,IN
AND
OR

Néhány függvény

```
Szöveg     : upper(), initcap(), substr(), length(), rtrim()
dátum      : sysdate, to_date(), to_char(), to_number()
numerikus  : abs(), mod(), round()
...
```

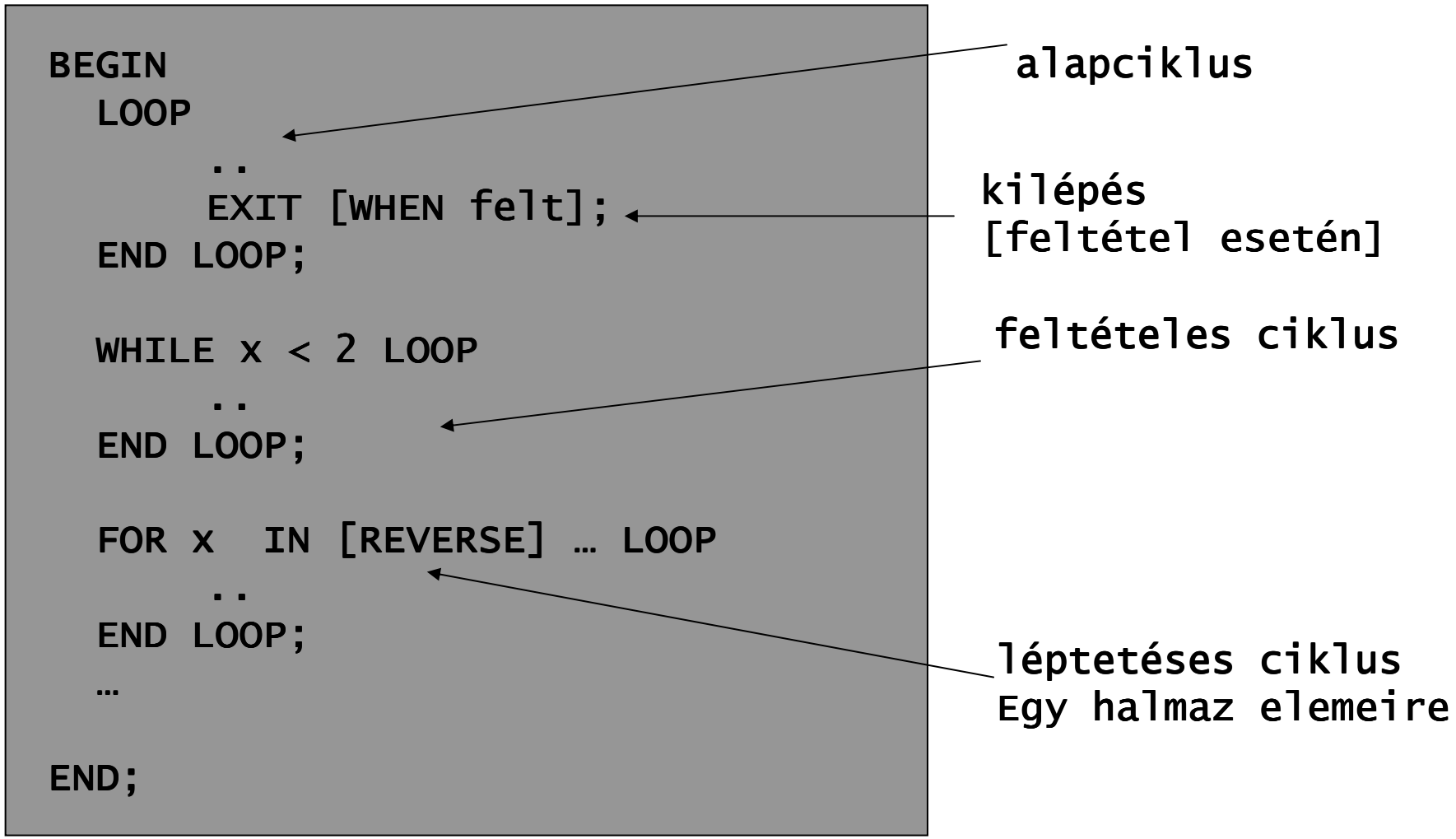
Programozási elemek

A törzs rész felépítése



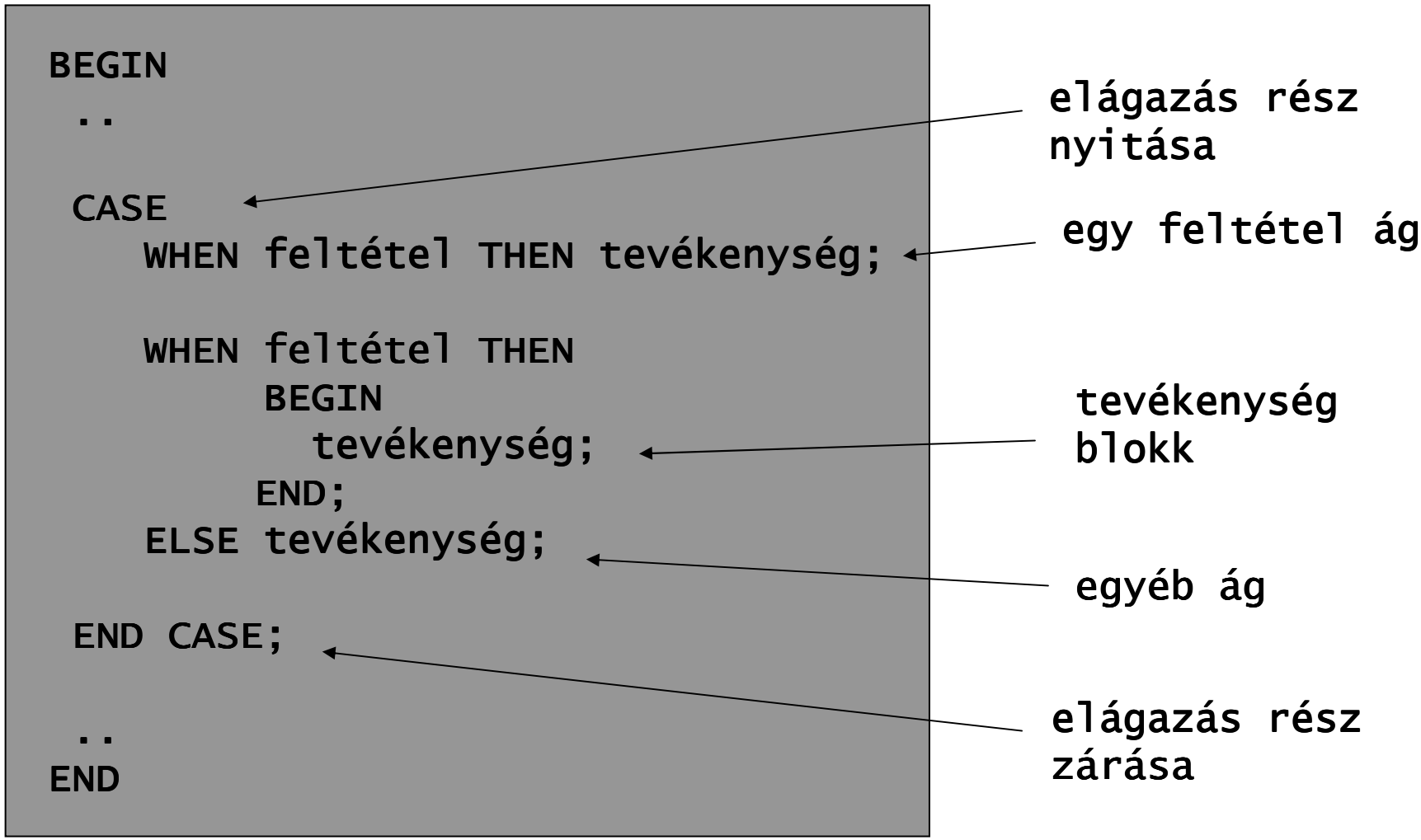
Programozási elemek

A törzs rész felépítése



Programozási elemek

Többszörös elágazás



Programozási elemek

Kurzor szerkezet, több rekord beolvasása a DB-ből

```
DECLARE
     CURSOR cnev(param) IS SELECT ..;
BEGIN
      OPEN cnev(aparam);
      LOOP
          FETCH cnev INTO vlista;
          EXIT WHEN cnev%NOTFOUND;
      END LOOP;
      CLOSE cnev;
```

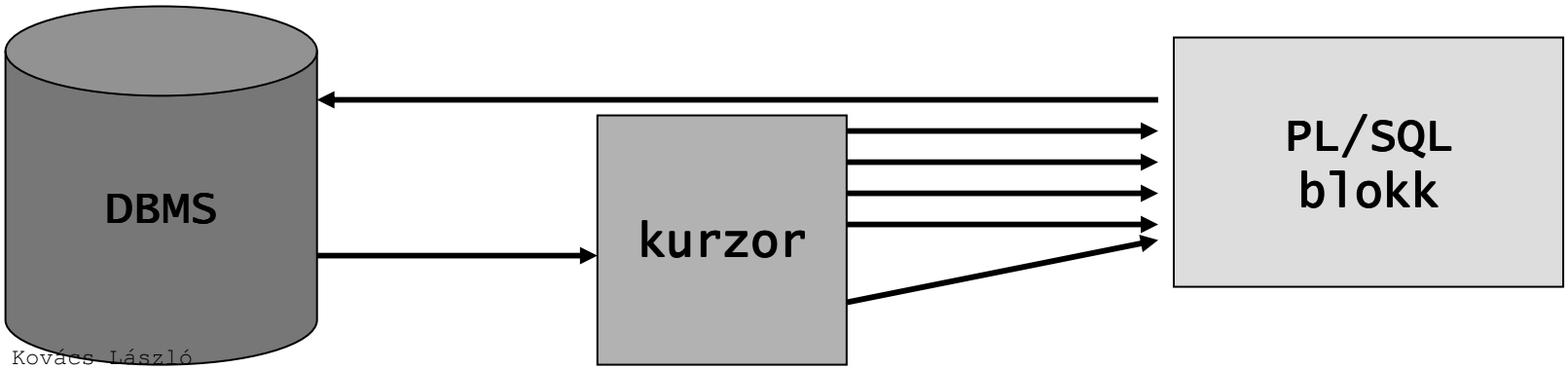
deklaráció, paraméterrel

nyitás

rekord olvasás

kilépés a ciklusból

lezárás



Programozási elemek

Közvetlen kurzorkezelési ciklus

```
DECLARE
     CURSOR cnev IS SELECT ..;
BEGIN
      OPEN cnev;
      FOR c1 IN cnec LOOP
         x := c1.mezo1;
         ...
      END LOOP;
      CLOSE cnev;
```

deklaráció, paraméterrel

nyitás

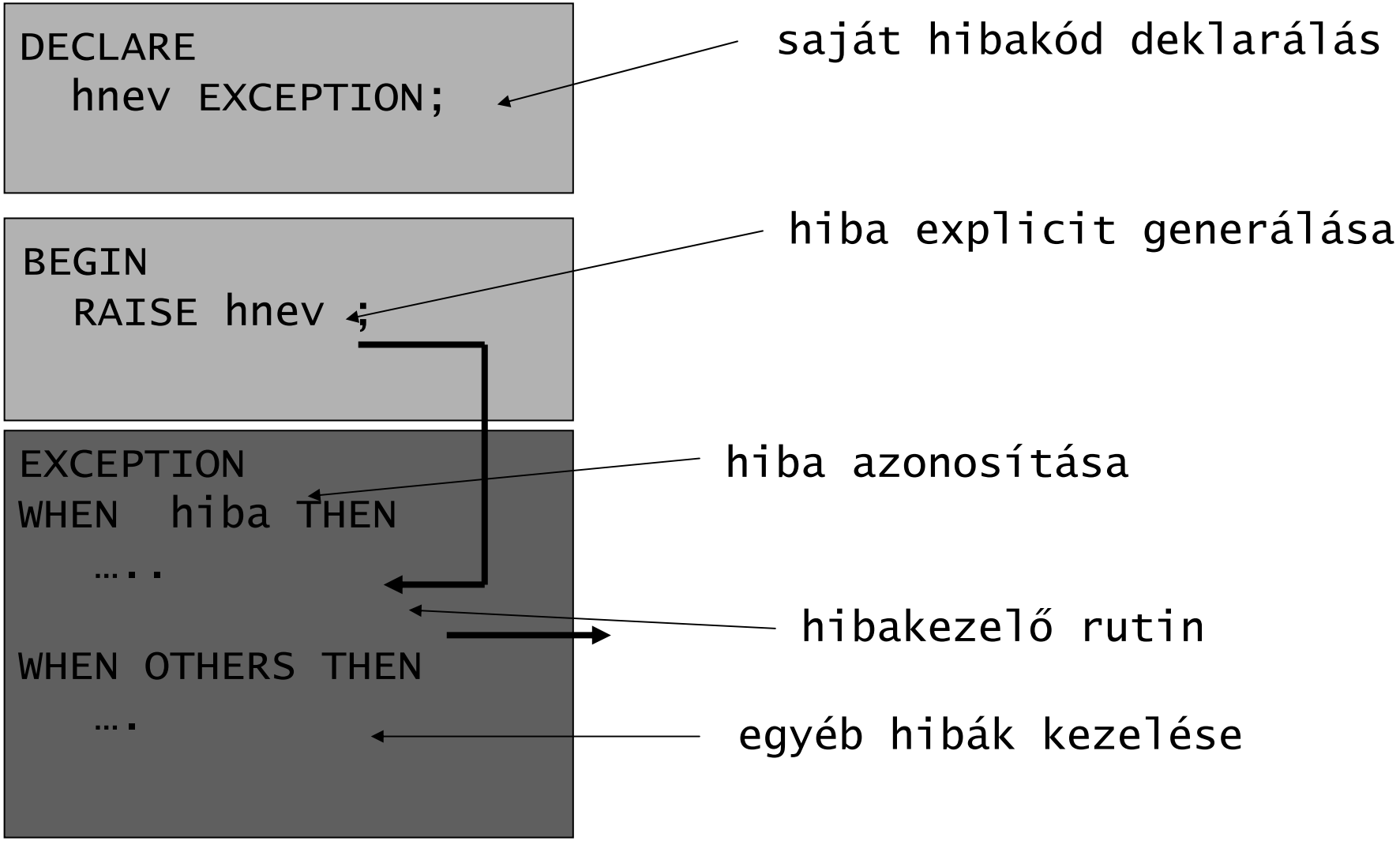
ciklus

Hivatkozás egy mezőre

lezárás

Programozási elemek

Hibakezelés



Hibakódok

no-data-found	: nincs több adat
Cursor-already-open	: már nyitott a kurzor
Dup-val-On-index	: kulcs dupplikálás
Storage-error	: memória hiba
Program-error	: pl/SQL motor hiba
Zero-divide	: nullával való osztás
invalid-cursor	: nem élő kurzor
Login-denied	: érvénytelen bejelentkezés
Invalid-number	: adattípus hiba
Too-many-rows	: több mint egy eredményrekord

Aktuális hibakód lekérdezése:

SQLCODE()

A hibakódhoz tartozó üzenet lekérdezése:

SQLERRM(kód)

Minta kód

```
DECLARE
  v_einheit_kurz varchar2(10);
  v_bezeichnung varchar2(40);
BEGIN
  v_einheit_kurz := 'kg';
  v_bezeichnung := 'Kilogramm';
  insert into einheit (einheit_kurz, bezeichnung)
  values (v_einheit_kurz, v_bezeichnung);
EXCEPTION when DUP_VAL_ON_INDEX then
  update einheit set
  bezeichnung = v_bezeichnung
  where einheit_kurz = v_einheit_kurz;
END;
```

Minta kód

```
declare
function einheit_existiert (p_einheit_kurz in varchar2)
return boolean is
v_anzahl pls_integer;
begin
select count(*)into v_anzahl
from einheit
where einheit_kurz = p_einheit_kurz;
return v_anzahl > 0;
end;
begin
if not einheit_existiert ('m') then
insert into einheit
(einheit_kurz, bezeichnung) values (m', 'Meter');
end if;
end;
```

Oracle DBMS: Metódusok

implementáció : PL/SQL, Java, C++,..
típus : objektum, osztály szint
megadás : deklaráció, definíció

```
SQL> CREATE TYPE EMBER AS OBJECT (
        NEV CHAR(20),
        MEMBER FUNCTION GET_NEV RETURN CHAR
       );

SQL> CREATE TYPE BODY EMBER AS
        MEMBER FUNCTION GET_NEV RETURN CHAR IS
          BEGIN
            RETURN SELF.NEV;
          END;
       END;
```

Metódusok

```
SQL> CREATE TABLE T1 OF EMBER;
SQL> INSERT INTO T1 VALUES('PETER');
SQL> SELECT P.GET_NEV() FROM T1 P;
          P.GET_NEV()
          ---------------------------------
          PETER

SQL>  CREATE TYPE EMBER AS OBJECT (
  2     NEV CHAR(20),
  3     MEMBER FUNCTION GET_NEV RETURN CHAR,
  4     MEMBER PROCEDURE SET_NEV(UN IN CHAR)
  5   );
```

Metódusok

```
SQL> CREATE TYPE BODY EMBER AS
        MEMBER FUNCTION GET_NEV RETURN CHAR IS
          BEGIN
            RETURN SELF.NEV;
          END;
        MEMBER PROCEDURE SET_NEV (UN IN CHAR) IS
          BEGIN
            SELF.NEV := UN;
          END;
      END;
SQL> CREATE TABLE T1 OF EMBER;
SQL> INSERT INTO T1 VALUES('GABOR');
SQL> INSERT INTO T1 VALUES(‘ANNA');
```

Metódusok

```
SQL> SELECT P.* FROM T1 P;
        NEV
        ------------------
        ANNA
        GABOR
SQL> SELECT REF(P) FROM T1 P;
        REF(P)
        ----------------------------------------------------------
        000028020992A57F97C14B4425A22249F…..
        000028020914FF76ACCFC4428592784D7….
SQL> SELECT DEREF(REF(P)) FROM T1 P;
        DEREF(REF(P))(NEV)
        ------------------------------------
        EMBER('ANNA              ')
        EMBER('GABOR              ')
```

```
DECLARE
  CURSOR C1 IS SELECT REF(P) FROM T1 P;
  E1 REF EMBER;
  E EMBER;
BEGIN
  OPEN C1;
  LOOP
    FETCH C1 INTO E1;
    EXIT WHEN C1%NOTFOUND;
    SELECT DEREF(E1) INTO E FROM DUAL;
    DBMS_OUTPUT.PUT_LINE(‘ NEV = ' || E.GET_NEV());
    --  DEREF(E1).SET_NEV(‘ZOLI’)  -- hiba
  END LOOP;
  CLOSE C1;
END;

SQL> SET SERVEROUTPUT ON
```

Metódusok

```
SQL> UPDATE T1 T SET T = EMBER('GABI') WHERE
          NEV = 'GABOR';
DECLARE
  CURSOR C1 IS SELECT DEREF(REF(P)) FROM T1 P
     FOR UPDATE;
  E EMBER;
BEGIN
  OPEN C1;
  LOOP
    FETCH C1 INTO E;
    EXIT WHEN C1%NOTFOUND;
    E.SET_NEV('ZOLI');
    UPDATE T1 T SET T = E WHERE CURRENT OF C1 ;
  END LOOP;
  CLOSE C1;
END;
```

Osztály metódusok

```
SQL> CREATE TYPE DOBOZ AS OBJECT (
        ELHOSSZ NUMBER(3),
        SZIN CHAR(20),
        STATIC FUNCTION DARAB(SZI IN CHAR)
          RETURN NUMBER );
SQL> CREATE TABLE DOBOZOK OF DOBOZ;
SQL> CREATE TYPE BODY DOBOZ AS
        STATIC FUNCTION DARAB (SZI IN CHAR)
           RETURN NUMBER IS
           DB NUMBER;
        BEGIN
           SELECT COUNT(*) INTO DB FROM DOBOZOK
             WHERE SZIN = SZI;
           RETURN DB;
        END;
      END;
```

```
SQL> INSERT INTO DOBOZOK VALUES(DOBOZ(12,'KEK'));
SQL> INSERT INTO DOBOZOK VALUES(DOBOZ(43,‘ZOLD’));
SQL> INSERT INTO DOBOZOK VALUES(DOBOZ(22,'KEK'));

SQL> SELECT DOBOZ.DARAB('KEK') FROM DUAL;
        DOBOZ.DARAB('KEK')
        ------------------
                         2
```

```
DROP TABLE KONYVEK1;
DROP TABLE KIADOK1;

CREATE TABLE KIADOK1 (KKOD NUMBER(3) PRIMARY
KEY, NEV CHAR(20));
CREATE TABLE KONYVEK1 (KOD NUMBER(5) PRIMARY
KEY, CIM CHAR(20), AR NUMBER(3), KIAD
REFERENCES KIADOK1);

DECLARE
  I NUMBER(5);
BEGIN
  FOR I IN 1..1000 LOOP
    INSERT INTO KIADOK1 VALUES(I,'KIADO' || TO_CHAR(I));
  END LOOP;
END;
COMMIT;
```

```
DECLARE
  I NUMBER(5);
BEGIN
  FOR I IN 1..100000 LOOP
    INSERT INTO KONYVEK1 VALUES(I,'CIM' || TO_CHAR(I),
    NULL,MOD(I,100)+1);
  END LOOP;
END;
COMMIT;
CREATE TYPE KIADO AS OBJECT (
  KKOD NUMBER(3),
  NEV CHAR(20));
CREATE TYPE KONYV AS OBJECT (
  KOD NUMBER(5),
  CIM CHAR(20),
  AR NUMBER(3),
  KIAD REF KIADO);
```

```
DECLARE
  I NUMBER(5);
BEGIN
  FOR I IN 1..1000 LOOP
    INSERT INTO KIADOK2 VALUES(KIADO(I,'KIADO' ||
    TO_CHAR(I)));
  END LOOP;
END;
DECLARE
  I NUMBER(5);  E REF KIADO;
BEGIN
  FOR I IN 1..100000 LOOP
     SELECT REF(T) INTO E FROM KIADOK2 T WHERE
     T.KKOD = MOD(I,100)+1;
     INSERT INTO KONYVEK2 VALUES(KONYV(I,
     'CIM' || TO_CHAR(I),NULL,E));
  END LOOP;
END;
```

```
SELECT TO_CHAR(SYSDATE,'HH:MI:SS') FROM DUAL;
CREATE VIEW V1 AS SELECT A.NEV, B.CIM FROM
  KIADOK1 A, KONYVEK1 B WHERE KIAD = KKOD;
SELECT COUNT(*) FROM V1;
SELECT TO_CHAR(SYSDATE,'HH:MI:SS') FROM DUAL;
CREATE VIEW V2 AS SELECT A.CIM CIM ,A.KIAD.NEV NEV
  FROM KONYVEK2 A;
SELECT COUNT(*) FROM V2;
SELECT TO_CHAR(SYSDATE,'HH:MI:SS') FROM DUAL;
```

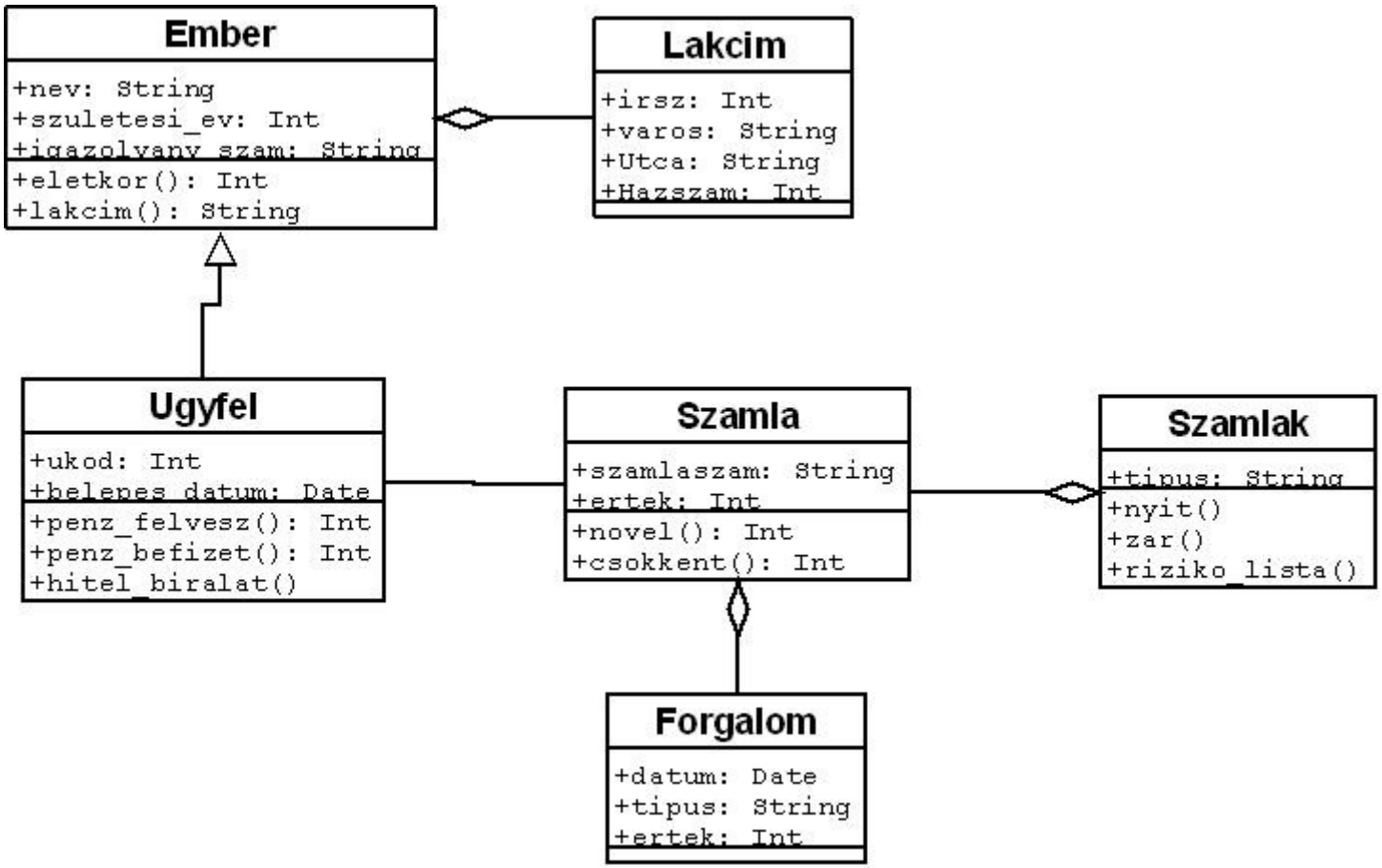
HASONLÓ VÉGREHAJTÁSI IDŐK
KB 1 SEC

Minta példa: banki információs rendszer

- Ügyfelek
- számlák
- mozgások

Mit lehet kihozni az ORDBMS modellből?

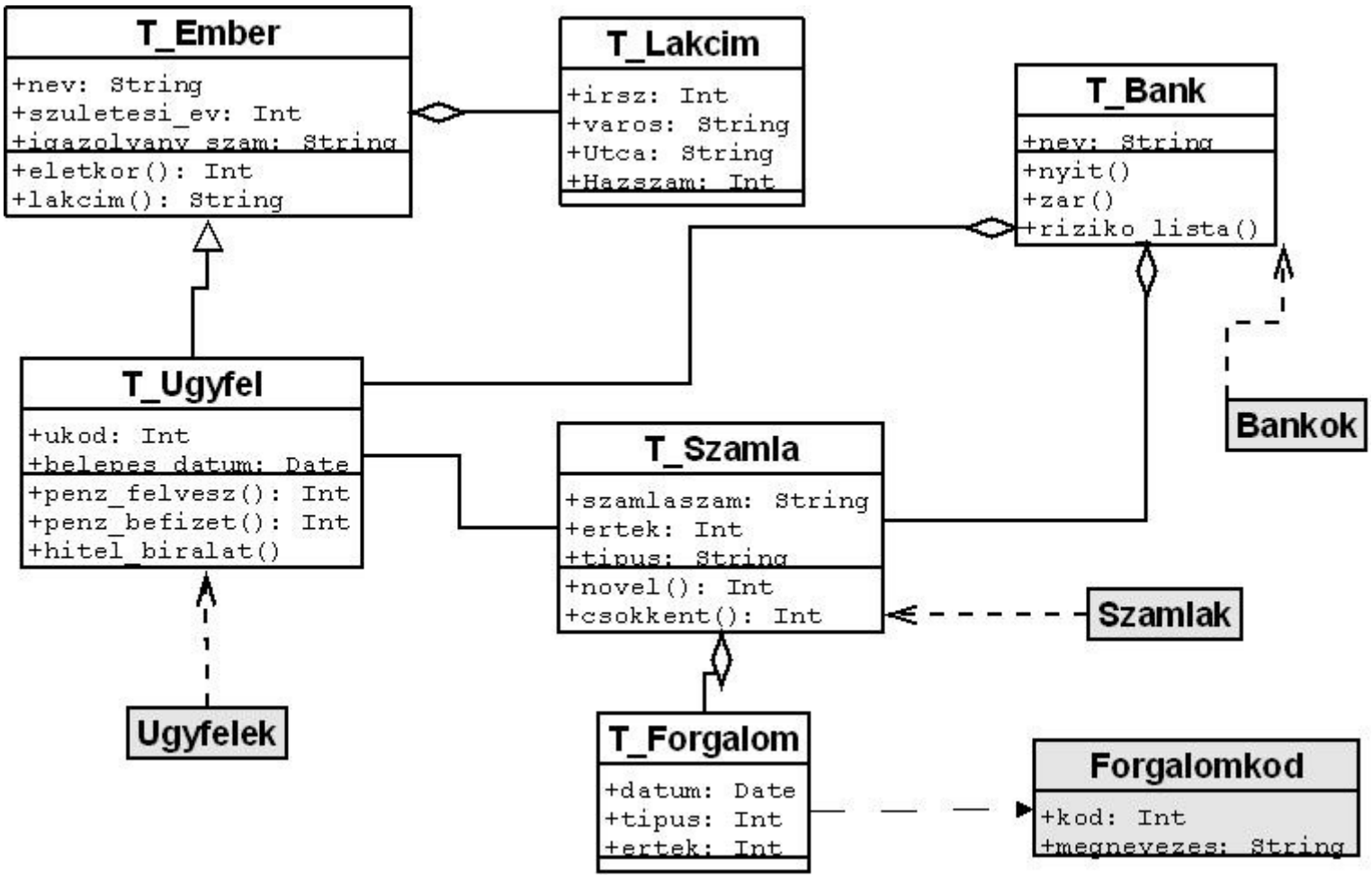
Minta példa: banki információs rendszer



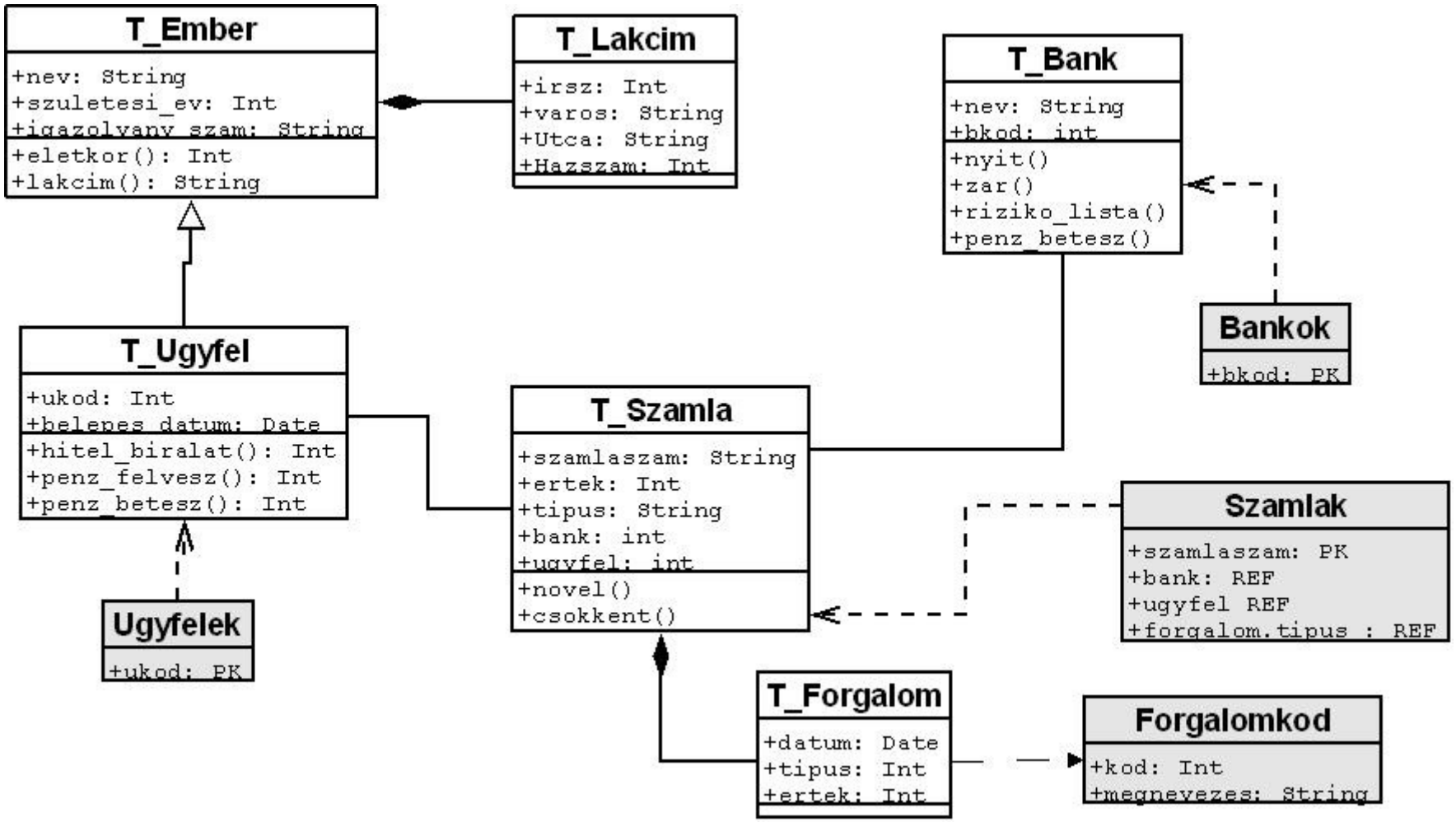
UML OOP orientált megközelítés

UML OOP orientált megközelítés problémái

- Táblák létrehozása,
- UDT és tábla kapcsolata
- SQL szemlélet vegyítése az OOP megközelítéssel
- OID kezelése
- kapcsolatok kezelése, REF vagy FOREIGN KEY vagy beépülés
- összetettség
- metódusok elérése



Relációs és OOP elvek összefésülése



Igazított, relációs és OOP elvek összefésülése,
Kapcsolatok egységesítése

```
CREATE OR REPLACE TYPE T_LAKCIM AS OBJECT (
  IRSZ NUMBER(4),
  VAROS CHAR(20),
  UTCA CHAR(20),
  HAZSZAM NUMBER(3)
);
.
/
SHOW ERRORS
CREATE OR REPLACE TYPE T_EMBER AS OBJECT (
  NEV CHAR(20),
  SZULETESI_EV NUMBER(4),
  IGAZOLVANY_SZAM CHAR(10),
  CIM  T_LAKCIM,
  MEMBER FUNCTION ELETKOR  RETURN NUMBER,
  MEMBER FUNCTION LAKCIM RETURN CHAR
) NOT FINAL;

CREATE OR REPLACE TYPE BODY T_EMBER AS
  MEMBER FUNCTION ELETKOR RETURN NUMBER AS
  BEGIN
     RETURN TO_NUMBER(TO_CHAR(SYSDATE,'YYYY')) - SELF.SZULETESI_EV;
  END;
  MEMBER FUNCTION LAKCIM RETURN CHAR AS
  BEGIN
     RETURN TRIM(SELF.CIM.VAROS) || ' ' || TRIM(SELF.CIM.UTCA) || ' ' ||
       TRIM(TO_CHAR(SELF.CIM.HAZSZAM));
  END;
END;
```

```
CREATE OR REPLACE TYPE T_UGYFEL UNDER T_EMBER  (
   UKOD NUMBER(4),
   BELEPES_DATUM DATE,
   MEMBER FUNCTION PENZ_FELVESZ (ERTEK IN NUMBER) RETURN NUMBER,
   MEMBER FUNCTION PENZ_BETESZ (ERTEK IN NUMBER) RETURN NUMBER,
   MEMBER FUNCTION HITEL_BIRALAT(ERTEK IN NUMBER) RETURN NUMBER
);
CREATE TYPE T_FORGALOM AS OBJECT (
  DATUM DATE,
  TIPUS NUMBER(3),
  ERTEK NUMBER(8)
);
CREATE or replace TYPE T_NAPLO AS TABLE OF T_FORGALOM;
CREATE OR REPLACE TYPE T_SZAMLA AS OBJECT (
  SZAMLASZAM CHAR(10),
  ERTEK NUMBER(8),
  TIPUS CHAR(3),
  FORGALOM T_NAPLO,
  UGYFEL number(8),
  BANK NUMBER(8),
  MEMBER PROCEDURE NOVEL (ERTEKA IN NUMBER),
  MEMBER PROCEDURE CSOKKENT (ERTEKA IN NUMBER)
);
CREATE TABLE UGYFELEK OF T_UGYFEL (UKOD PRIMARY KEY, NEV NOT NULL)
  OBJECT IDENTIFIER IS PRIMARY KEY;

CREATE TABLE FORGALOMKOD (KOD NUMBER(3) PRIMARY KEY,MEGNEVEZES CHAR(20));
```

```
CREATE OR REPLACE TYPE T_BANK AS OBJECT (
   NEV CHAR(20),
   BKOD NUMBER(4),
   STATIC PROCEDURE PENZ_BETESZ (ERTEK IN NUMBER, UGYF IN NUMBER),
   MEMBER FUNCTION NYIT (NNEV IN NUMBER, NSZEV IN NUMBER, NSZIG IN CHAR) RETURN CHAR,
   MEMBER FUNCTION RIZIKO_LISTA RETURN INT
);
CREATE TABLE BANKOK OF T_BANK (BKOD PRIMARY KEY);


CREATE TABLE SZAMLAK OF T_SZAMLA (SZAMLASZAM PRIMARY KEY,
      UGYFEL REFERENCES UGYFELEK, BANK REFERENCES BANKOK)
 OBJECT IDENTIFIER IS PRIMARY KEY
 NESTED TABLE FORGALOM STORE AS SZFORGALOM;


CREATE OR REPLACE TYPE BODY T_SZAMLA AS
 MEMBER PROCEDURE NOVEL (ERTEKA IN NUMBER) AS
 BEGIN
   SELF.ERTEK := SELF.ERTEK + ERTEKA;
   INSERT INTO TABLE(SELECT T.FORGALOM FROM SZAMLAK T WHERE
        T.SZAMLASZAM=SELF.SZAMLASZAM)
     VALUES (T_FORGALOM(SYSDATE,1,ERTEKA));
 END;
```

```
 MEMBER PROCEDURE CSOKKENT (ERTEKA IN NUMBER) AS
    ERTEKU NUMBER;
   BEGIN
    IF ERTEKA > SELF.ERTEK THEN
      ERTEKU := SELF.ERTEK;
      SELF.ERTEK := 0;
      INSERT INTO TABLE(SELECT T.FORGALOM FROM SZAMLAK T WHERE
          T.SZAMLASZAM=SELF.SZAMLASZAM) VALUES (T_FORGALOM(SYSDATE,2,ERTEKU));
    ELSE
      SELF.ERTEK := SELF.ERTEK - ERTEKA;
      INSERT INTO TABLE(SELECT T.FORGALOM FROM SZAMLAK T WHERE
         T.SZAMLASZAM=SELF.SZAMLASZAM)  VALUES (T_FORGALOM(SYSDATE,2,ERTEKA));
    END IF;
  END;
END;

CREATE OR REPLACE TYPE BODY T_UGYFEL AS
  MEMBER FUNCTION PENZ_BETESZ (ERTEK IN NUMBER) RETURN NUMBER AS
    SZ T_SZAMLA;
    BEGIN
     SELECT DEREF(REF(T)) INTO SZ FROM SZAMLAK T WHERE T.UGYFEL = SELF.UKOD;
     SZ.NOVEL(ERTEK);
    RETURN 0;
  END;
END;
.
```

```
CREATE OR REPLACE TYPE BODY T_BANK AS
  STATIC PROCEDURE PENZ_BETESZ (ERTEK IN NUMBER, UGYF IN NUMBER) AS
      UUF T_UGYFEL;
      EE NUMBER;
     BEGIN
      SELECT DEREF(REF(T)) INTO UUF FROM UGYFELEK T WHERE T.UKOD = UGYF;
      EE := UUF.PENZ_BETESZ(ERTEK);
     END;
END;

---------------

INSERT INTO BANKOK VALUES ('BANK1',1);

INSERT INTO UGYFELEK VALUES ('PETER',1969,'IG11',T_LAKCIM(11,'MISKOLC','PETOFI',23),
   11,SYSDATE);

SELECT NEV FROM UGYFELEK;

INSERT INTO SZAMLAK VALUES ('S1',0,'A',T_NAPLO(T_FORGALOM(SYSDATE,0,0)),11,1);
INSERT INTO SZAMLAK VALUES ('S2',0,'A',T_NAPLO(T_FORGALOM(SYSDATE,0,0)),12,1);

SELECT * FROM SZAMLAK WHERE SZAMLASZAM='S1';

SELECT * FROM (SELECT FORGALOM FROM SZAMLAK WHERE SZAMLASZAM='S1');

EXEC T_BANK.PENZ_BETESZ(30,11);
```

Tapasztalatok

- SELF nem mindig módosítható (FUNCTION, PROCEDURE)
- REF(SELF) nem megy
- PROCEDURE csak másik eljárásból vagy EXEC révén hívható meg
- FV-ek SELECT-ben meghívva nem tartalmazhatnak DML parancsokat
- SQL és OOP illesztése nehézkes