SQL – Structured Query Language

- IBM fejlesztette ki 70-es években (SEQUEL)
- 1986-ban lett szabvány (SQL86)
- SQL86, SQL89, SQL92, SQL99, SQL3, SQL2003
- **DDL**, **DML**, **QL**, **DCL** (jogosultságkezelő és tranzakció-kezelő nyelv), …
- Nyelvjárások (Oracle, Sybase, DB2, Progress, MSSQL, mySQL, SQL Server, PostgreSQL, Access,…)
- Részben procedurális, részben deklaratív

Adatdefiniáló nyelv

- **CREATE**
- *Adatbázis objektum létrehozása.*
- *Példa adatbázistábla definíciójára:*

```
create table ACCOUNTS (
AC_ID_PK number primary key,
AC_STATUS number,
AC_COUNTRY_ID number default 44,
AC_CREATED date default sysdate,
AC_ACCOUNT varchar2(50)
);
```

- *Példa index definiálására:*

```
create index idx_case_ins on my_table
(UPPER(empname));
```

Adatdefiniáló nyelv

- **ALTER**
- *Adatbázis-objektum módosítása. Példa:*
- ALTER TABLE STAFF_OPTIONS **ADD** SO_INSURANCE_PROVIDER Varchar2(35);
- ALTER TABLE STAFF_OPTIONS **MODIFY** SO_INSURANCE_PROVIDER Varchar2(35) DEFAULT 'ABC Ins';

Adatdefiniáló nyelv

- **DROP**
- *Egy adatbázisbeli objektum megszüntetése.*
- *Példa:*

```
DROP table ACCOUNTS;
DROP view BP_ACCOUNTS;
```

Lekérdezőnyelv

- A lekérdezés általános formája:

  SELECT [DISTINCT] *select_list*

  FROM *table_list*

  [WHERE *conditions*]

  [GROUP BY *group_by_list*]

  [HAVING *search_conditions*]

  [ORDER BY *order_list* [ASC | DESC]]
  [FOR UPDATE *for_update_options*]

Lekérdezőnyelv

- Relációs algebrai műveletek kifejezhetők SQL-ben
- Szorzás:

```
SELECT * FROM Beteg, Kezeles;
```

- Kiválasztás:

```
SELECT *
FROM suppliers
WHERE (city = 'New York' and name = 'IBM')
or (city = 'Newark');
```

- Vetítés:

```
SELECT DISTINCT city, state
FROM suppliers;
```

Lekérdezőnyelv

- Egyesítés:

```
select supplier_id
from suppliers
UNION
select supplier_id
from orders;
```

- Kivonás (például r(a,b)-s(a,b)):

```
SELECT distinct r.a, r.b
FROM r WHERE not exists (select * from s
where r.a = s.a and r.b = s.b);
```

Lekérdezőnyelv

- A származtatott műveletek, és minden relációs algebrai kifejezés felírható SQL-ben.
- Összekapcsolás (equi-join):

```
SELECT suppliers.supplier_id,
suppliers.supplier_name, orders.order_date
FROM suppliers, orders
WHERE suppliers.supplier_id = orders.supplier_id;
```

- Külső összekapcsolás:

```
select suppliers.supplier_id,
suppliers.supplier_name, orders.order_date
from suppliers, orders
where suppliers.supplier_id = orders.supplier_id(+);
```

Lekérdezőnyelv

- $r(a,b) \div s(b)$ hányados kifejezése SQL-ben:
- $r(a,b) \div s(b) = \Pi_a(r) - \Pi_a(\Pi_a(r) \times s - r)$
- $\Pi_a(r) \times s = \Pi_{r.a,s.b}(r \times s)$
- select distinct r.a,s.b from r,s;
- $\Pi_a(r) \times s - r$
- select distinct r.a,s.b from r r1, s s1
  where **not exists**
  (select * from r r2
  where r2.a=r1.a and s1.b=r2.b);
- $\Pi_a(\Pi_a(r) \times s - r)$
- select distinct r.a from r r1, s s1
  where **not exists**
  (select * from r r2
  where r2.a=r1.a and s1.b=r2.b);
- $\Pi_a(r) - \Pi_a(\Pi_a(r) \times s - r)$
- select distinct r2.a from r r2
  where **not exists**
  (select * from r r1, s s1
  where r2.a=r1.a and
  not exists
  (select * from r r3
  where r3.a=r1.a
  and s1.b=r3.b));

Lekérdezőnyelv

- Speciális kiválasztási feltételek:
- Mintaillesztés:

```
SELECT * FROM suppliers
WHERE supplier_name like '%bob%';
SELECT * FROM suppliers
WHERE supplier_name not like 'T%';
SELECT * FROM suppliers
WHERE supplier_name like 'Sm_th';
SELECT * FROM suppliers
WHERE supplier_name LIKE 'H%!%'
escape '!';
```

Lekérdezőnyelv

- Nullérték lekérdezése:

```
select * from suppliers
where supplier_name IS NULL;
```

- Zárt intervallumhoz tartozás:

```
SELECT *
FROM suppliers
WHERE supplier_id between 5000 AND 5010;
```

```
SELECT *
FROM orders
WHERE order_date between
to_date('2003/01/01', 'yyyy/mm/dd')
AND to_date('2003/12/31', 'yyyy/mm/dd');
```

Lekérdezőnyelv

- Alkérdések használata Where részben
- IN – (multi)halmaznak eleme

```
SELECT *
FROM suppliers
WHERE supplier_name in ( 'IBM', 'Hewlett
Packard', 'Microsoft');

select * from all_tables tabs
where tabs.table_name in (select
cols.table_name from all_tab_columns cols
where cols.column_name = 'SUPPLIER_ID');
```

Lekérdezőnyelv

- Maximum kiválasztása alkérdéssel

  select b1.fizetes from ber b1

  where b1.fizetes >= all (select b2.fizetes from ber b2);

- *Használható*:

  all – az alkérdés minden értékére teljesül a feltétel (alapértelmezés, nem kell kiírni)

  any – az alkérdés legalább egy értékére teljesül a feltétel

Lekérdezőnyelv

- Aggregátorfüggvények
  - count, avg, max, min, sum
- SELECT COUNT(*) as "Number of employees"
  FROM employees
  WHERE salary > 25000;
- SELECT COUNT(DISTINCT department) as "Unique departments"
  FROM employees
  WHERE salary > 25000;
- SELECT SUM(DISTINCT salary) as "Total Salary"
  FROM employees
  WHERE salary > 25000;
- SELECT SUM(income - expenses) as "Net Income"
  FROM gl_transactions;

Lekérdezőnyelv

- SELECT MIN(salary) as "Lowest salary",
  MAX(salary) as "Highest salary"
  FROM employees;
- SELECT Avg(salary) as "Avg Salary"
  FROM employees
  WHERE salary > 25000;

Lekérdezőnyelv

- Csoportosítás, csoportok szűrése
- SELECT department, COUNT(*) as "Number of employees"
  FROM employees
  WHERE salary > 25000
  GROUP BY department
  HAVING COUNT(*) > 10;
- SELECT department, MIN(salary) as "Lowest salary"
  FROM employees
  GROUP BY department
  HAVING MIN(salary) = 35000;

Lekérdezőnyelv

- Eredmény rendezése
- SELECT supplier_city, supplier_state
  FROM suppliers
  WHERE supplier_name = 'IBM'
  ORDER BY supplier_city DESC,
  supplier_state ASC;
- SELECT supplier_city
  FROM suppliers
  WHERE supplier_name = 'IBM'
  ORDER BY 1 DESC;

Lekérdezőnyelv

- Nézettábla (VIEW) használata
- áttekinthetőség (összetett lekérdezés felbontása egyszerűbb kérdésekre)
- adatvédelem (oszlopok, sorok korlátozása)
- ellenőrzött adatbevitel
- fizikai helyet nem foglal (csak a lekérdezés definícióját tároljuk)
- mindenhol használható, ahol közönséges tábla használható
- CREATE VIEW sup_orders AS
  SELECT suppliers.supplier_id, orders.quantity, orders.price
  FROM suppliers, orders
  WHERE suppliers.supplier_id = orders.supplier_id
  and suppliers.supplier_name = 'IBM';

Lekérdezőnyelv

- CREATE OR REPLACE VIEW high_salary AS SELECT name, salary FROM emp WHERE salary > 10000 WITH CHECK OPTION;
- Az OR REPLACE rész miatt lecseréli a definíciót, ha már korábban létrehoztunk ilyen nevű nézettáblát.
- INSERT INTO high_salary VALUES ('Smith',50000); /* **a sor bekerül az emp táblába** */
- INSERT INTO high_salary VALUES ('Smith',200); /* **a sor nem kerül be az emp táblába, hibaüzenet***/

Adatkezelő nyelv

- **INSERT**
- *Adatokat ad hozzá egy táblához.*
- CREATE TABLE People (surname VARCHAR2(25), firstname VARCHAR2(25));
- INSERT INTO People VALUES ('McKay','Alice');
- INSERT INTO People (surname,firstname) VALUES ('Smith','John');
- INSERT INTO People (surname) VALUES ('Ullman'); /* **A firstname nullérték lesz!** */
- INSERT INTO People (firstname, surname) (select emp_first, emp_surname from Employees);

Adatkezelő nyelv

- **UPDATE**
- *Módosítást hajt végre az adatokon.*
- UPDATE suppliers
  SET name = 'HP'
  WHERE name = 'IBM';
- UPDATE suppliers
  SET supplier_name = (SELECT customers.name
  FROM customers
  WHERE customers.customer_id
  = suppliers.supplier_id)
  WHERE EXISTS
  (SELECT customers.name
  FROM customers
  WHERE customers.customer_id =
  suppliers.supplier_id);

Adatkezelő nyelv

- **DELETE**
- Adatokat töröl egy táblából.
- DELETE FROM suppliers
  WHERE supplier_name = 'IBM';
- DELETE FROM suppliers
  WHERE EXISTS
  ```
  ( select customers.name
  from customers
  where customers.customer_id =
          suppliers.supplier_id
  and customers.customer_name = 'IBM' );
  ```

Jogosultságkezelő nyelv

- Jogok: select, insert, update, delete, alter, reference (megszorítás hivatkozhat a táblára), index
- grant select, insert, update, delete on suppliers to smith;
- grant **all** on suppliers to smith;
- grant select on suppliers to **public**;
- revoke delete on suppliers from anderson;
- revoke **all** on suppliers from **public**;
- Jogok továbbadásának joga:
  GRANT SELECT ON emp TO scott **WITH GRANT OPTION;**
- Visszavonáskor a továbbadott jogok is visszavonásra kerülnek

Kulcsok, megszorítások

- Elsődleges kulcs = azonosító (nullértéket nem tartalmazhat)
- **1** elsődleges kulcsot lehet defininiálni táblánként
- több oszlopból is állhat
- CREATE TABLE supplier (supplier_id numeric(10) not null, supplier_name varchar2(50) not null, contact_name varchar2(50), CONSTRAINT supplier_pk PRIMARY KEY (supplier_id, supplier_name));
- Megszűntethető, vagy ideiglenes ki/be kapcsolható
- ALTER TABLE supplier **drop** CONSTRAINT supplier_pk;
- ALTER TABLE supplier **disable** CONSTRAINT supplier_pk;

Idegen kulcsok

- A sorban szereplő adat egy másik táblában szereplő sor azonosítója (gyerek-szülő tábla). Idegen kulcs.
- Léteznie kell a hozzátartozó sornak a másik táblában.
- CREATE TABLE products
  (product_id numeric(10) not null, supplier_id numeric(10) not null, supplier_name varchar2(50) not null,
  CONSTRAINT fk_supplier_comp
  FOREIGN KEY (supplier_id, supplier_name)
  REFERENCES supplier (supplier_id, supplier_name));

- **Szülő tábla (szállító) sorának törlésével az azonosítót tartalmazó sorok a gyerek táblából (termékek) is törlődnek a következő definíció esetén:**
- CREATE TABLE products
  (product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id)
  REFERENCES supplier (supplier_id)
  ON DELETE CASCADE);

- Törölhető, ki/bekapcsolható
- ALTER TABLE products
  enable CONSTRAINT fk_supplier;

Másodlagos kulcsok

- Több oszlopcsoport is megadható azonosítás céljára
- CREATE TABLE supplier
  (supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_unique UNIQUE
  (supplier_id, supplier_name));

- Törölhető, ki/bekapcsolható

Függőségek ellenőrzése

- CREATE TABLE suppliers (supplier_id numeric(4), supplier_name varchar2(50), CONSTRAINT check_supplier_id CHECK (supplier_id BETWEEN 100 and 9999));
- CREATE TABLE suppliers (supplier_id numeric(4), supplier_name varchar2(50), CONSTRAINT check_supplier_name CHECK (supplier_name = upper(supplier_name)));

SQL programok készítése

- **PL/SQL – programszerkezetek használata**
- Feldolgozandó sorhalmazokhoz **sormutatókat, kurzor**okat definiálunk

- CREATE OR REPLACE Function FindCourse

```
  ( name_in IN varchar2 )
  RETURN number
IS
   cnumber number;
   CURSOR c1
   IS
     SELECT course_number
      from courses_tbl
      where course_name = name_in;

BEGIN
open c1;
fetch c1 into cnumber;

if c1%notfound then
    cnumber := 9999;
end if;

close c1;
RETURN cnumber;
END;
```

SQL programok készítése

- Kurzorok attribútumai: %ISOPEN, %FOUND, %NOTFOUND, %ROWCOUNT
- Kurzorhoz tartozó sorok zárolhatók:
- CURSOR c1
  IS
  SELECT course_number, instructor
  from courses_tbl
  FOR UPDATE OF instructor;
- *A kurzor aktuális értéke módosítható, törölhető*
- UPDATE courses_tbl
  SET instructor = 'SMITH'
  WHERE CURRENT OF c1;
- DELETE FROM courses_tbl
  WHERE CURRENT OF c1;

SQL programok készítése

- **Vezérlő szerkezetek PL/SQL-ben**

-
```
CREATE OR REPLACE Function IncomeLevel
    ( name_in IN varchar2 )
    RETURN varchar2
IS
    monthly_value number(6);
    lLevel varchar2(20);
    cursor c1 is
      select monthly_income
      from employees
      where name = name_in;
  BEGIN
open c1;
fetch c1 into monthly_value;
close c1;
IF monthly_value <= 4000 THEN
    lLevel := 'Low Income';
ELSIF monthly_value > 4000 and monthly_value <= 7000 THEN
    lLevel := 'Avg Income';
ELSIF monthly_value > 7000 and monthly_value <= 15000 THEN
    lLevel := 'Moderate Income';
ELSE
    lLevel := 'High Income';
END IF;
  RETURN lLevel;
END;
```

SQL programok készítése

- select table_name,
  CASE owner
  WHEN 'SYS' THEN 'The owner is SYS'
  WHEN 'SYSTEM' THEN 'The owner is SYSTEM'
  ELSE 'The owner is another value' END
  from all_tables;
- A fenti CASE ekvivalens a következővel:

```
IF owner = 'SYS' THEN
    result := 'The owner is SYS';
ELSIF owner = 'SYSTEM' THEN
   result := 'The owner is SYSTEM'';
ELSE
   result := 'The owner is another value';
END IF;
```

SQL programok készítése

- **Ciklusszervezés**
- LOOP

```
    ....
    monthly_value := daily_value * 31;
    ....
    EXIT WHEN monthly_value > 4000;
END LOOP;
```

- FOR Lcntr IN REVERSE 1..15

```
LOOP
    ....
    LCalc := Lcntr * 31;
    ....
END LOOP;
```

- WHILE monthly_value <= 4000

```
LOOP
    ....
    monthly_value := daily_value * 31;
    ....
END LOOP;
```

SQL programok készítése

- **Kurzor sorain végighaladó ciklus**

- CREATE OR REPLACE Function TotalIncome
```
   ( name_in IN varchar2 )
   RETURN varchar2
IS
   total_val number(6);

   cursor c1 is
     select monthly_income
     from employees
     where name = name_in;
BEGIN
total_val := 0;
FOR employee_rec in c1
LOOP
   total_val := total_val + employee_rec.monthly_income;
END LOOP;

RETURN total_val;
END;
```

SQL programok készítése

- Ugrás egy programrészre

  GOTO label_name;

  Label_name: {*statements*}
- Konstansok megadása

  LTotal constant numeric(8,1) := 8363934.1;
- Változók definiálása kezdeti értékkel

  LType varchar2(10) := 'Example';

SQL programok készítése

- **Függvények megadása** (paraméter lehet IN, OUT, IN OUT)
- CREATE [OR REPLACE] FUNCTION function_name
  [(parameter [,parameter])]
  RETURN return_datatype
  IS | AS
  [declaration_section]
  BEGIN
  executable_section
  [EXCEPTION
  exception_section]
  END [function_name];

- **Eljárások megadása**
- CREATE [OR REPLACE] PROCEDURE procedure_name
  [(parameter [,parameter])]
  IS | AS
  [declaration_section]
  BEGIN
  executable_section
  [EXCEPTION
  exception_section]
  END [procedure_name];

SQL programok készítése

- PL/SQL eljárások, függvények végrehajtása:
- EXEC statement;
- EXEC package.procedure;
- EXEC package.function(parameters);

SQL programok készítése

- **Beépített függvények kategóriái**
- Karakterfüggvények
- Matematikai függvények
- Dátumfüggvények
- Konverziós függvények
- Hibakezelő függvények
- Egyéb függvények

SQL programok készítése

- Karakterfüggvények:

Ascii	Convert	Lower	Soundex
AsciiStr	Decompose	Lpad	Substr
Chr	Dump	Ltrim	Translate
Compose	Initcap	Replace	Trim
Concat	Instr	Rpad	Upper
Concat with ||	Length	Rtrim	VSize

SQL programok készítése

- Matematikai függvények

Abs	Covar_pop	Max	Sqrt
Acos	Covar_samp	Median	StdDev
Asin	Count	Min	Sum
Atan	Cume_Dist	Mod	Tan
Atan2	Dense_Rank	Power	Tanh
Avg	Exp	Rank	Trunc (1)
Bin_To_Num	Extract	Remainder	Trunc (2)
BitAnd	Floor	Round (1)	Var_pop
Ceil	Greatest	Round (2)	Var_samp
Corr	Least	Sign	Variance
Cos	Ln	Sin	
Cosh	Log	Sinh	

(1) - numbers, (2) - dates)

SQL programok készítése

- Dátumfüggvények

Add_Months	Last_Day	Round	To_Date
Current_Date	LocalTimestamp	SessionTimeZone	Trunc
Current_Timestamp	Months_Between	Sysdate	Tz_Offset
DbTimeZone	New_Time	Sys	Timestamp
From_Tz	Next_Day	To_Char	

- Hónap utolsó napja:
- last_day(to_date('2004/02/03', 'yyyy/mm/dd'))

  értéke: Feb 29, 2004
- Dátum karakterré alakítása:
- to_char(sysdate, 'Month DD, YYYY')

  értéke: 'March 05, 2007'

SQL programok készítése

- Konverziós függvények

Bin_To_Num	NumToDSInterval	To_DSInterval	To_Single_Byte
Cast	NumToYMInterval	To_Lob	To_Timestamp
CharToRowid	To_Char	To_Multi_Byte	To_Timestamp_Tz
From_Tz	To_Clob	To_NClob	To_YMInterval
HexToRaw	To_Date	To_Number	

- Általános konverziós függvény:

  cast('22-Aug-2003' AS varchar2(30))

SQL programok készítése

- Hibakezelő függvények

  SQLCODE

  SQLERRM

- Egyéb függvények

BFilename	Decode	NVL2	UserEnv
Cardinality	Group_ID	Sys_Context	
Case Statement	NANVL	Uid	
Coalesce	NVLUser		

SQL programok készítése

- **Néhány fontosabb függvény**:
- Nullértékek helyettesítése:
- select NVL(supplier_city, 'n/a')
  from suppliers;
- Értékek cseréje, attól függően, hogy nullérték-e vagy nem:
- select NVL2(supplier_city, 'Completed', 'n/a')
  from suppliers;
- A lista első nem nullértéke:
- SELECT coalesce(address1, address2, address3) result
  FROM suppliers;

SQL programok készítése

- Fontosabb Oracle rendszertáblák
- ALL_CATALOG
- ALL_CONSTRAINTS
- ALL_TABLES
- ALL_TAB_COLUMNS
- ALL_TRIGGERS
- ALL_USERS
- ALL_VIEWS
- TABLE_PRIVILEGES

SQL programok készítése

- **Triggerek használata**
- Adatbázis események bekövetkezésekor automatikusan elinduló eljárások

- **A Triggerek típusai:**
- Insert Trigger:
  - BEFORE INSERT Trigger
  - AFTER INSERT Trigger
- Update Trigger:
  - BEFORE UPDATE Trigger
  - AFTER UPDATE Trigger
- Delete Trigger:
  - BEFORE DELETE Trigger
  - AFTER DELETE Trigger

- *A triggerek ki/bekapcsolhatók, megszüntethetők:*
- ALTER TRIGGER orders_before_insert **DISABLE**;
- DROP TRIGGER orders_before_insert;

SQL programok készítése

- *Például After Insert trigger megadása:*
- CREATE or REPLACE TRIGGER *trigger_name*
  AFTER INSERT
  ON *table_name*
  [FOR EACH ROW]
  [WHEN feltétel]
- DECLARE
  -- változók deklarálása
  BEGIN
  -- a trigger utasításai
  EXCEPTION
  WHEN ...
  -- kivételkezelés
  END;

SQL programok készítése

```
CREATE OR REPLACE TRIGGER orders_after_update
AFTER UPDATE
   ON orders
   FOR EACH ROW
DECLARE
   v_username varchar2(10);
BEGIN
   SELECT user INTO v_username
   FROM dual;
   INSERT INTO orders_audit
    ( order_id,
      quantity_before,
      quantity_after,
      username )
   VALUES
    ( :new.order_id,
      :old.quantity,
      :new.quantity,
      v_username );
END;
```

Tranzakciókezelés SQL-ben

- Set Transaction
  SET TRANSACTION READ ONLY;
  SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
  SET TRANSACTION USE ROLLBACK SEGMENT name;
- Commit
  A változtatások véglegesítése, mások számára láthatóvá tétele.
- Rollback
  Az összes változtatás érvénytelenítése egy adott pontig.
  ROLLBACK [WORK] [TO [SAVEPOINT] savepoint_name];
- Lock Table
  Táblák különböző finomságú zárolása
  LOCK TABLE tables IN lock_mode MODE [NOWAIT];
  Zárolási módok:
  ROW SHARE
  ROW EXCLUSIVE
  SHARE UPDATE
  SHARE
  SHARE ROW EXCLUSIVE
  EXCLUSIVE