Baze de Date

Cap. 2. SQL - utilizare DDL, gestiune date, interogari active

Textbook: Ramakrishnan, Gehrke, “Database Management Systems”, McGraw Hill, Cap. 3

2023 UPT Conf.Dr. Dan Pescaru

Introducere

1. SQL – Structured Query Language
2. IBM SystemR project - SEQUEL (1970) (Structured English Query Language)
3. Oracle V2 (1979)
4. Standardul curent in industrie (ANSI/ISO)
5. Limbaj declarativ
6. SGBD-uri comerciale: dialecte SQL + extensii procedurale (Oracle PL/SQL, Ms T-SQL etc.)

Standardul SQL

1. ANSI/ISO:
   - SQL'1986 (SQL-87) – Propus de ANSI
   - SQL'1992 (SQL2) – revizie majoră (ISO 9075), cel mai folosit în zilele noastre în BD relaționale de uz general
   - SQL'1999 (SQL3) – adaugă interogări recursive și tipuri de date orientate pe obiecte
   - SQL'2003 (SQL/XML) – adaugă facilități XML
   - SQL'2011 – adaugă suport pentru date temporale
   - SQL'2016 – adaugă suport pentru tipare și JSON
   - SQL'2019 – în dezvoltare – suport pentru tablouri multidimensionale

Limbajul SQL

1. Data Definition Language (DDL)
   - Instrucțiuni pentru definirea unei baze de date și a obiectelor acesteia (tabele, indecși, constrângeri etc.)
2. Data Manipulation Language (DML)
   - Interogări pentru modificarea și extragerea datelor dintr-o BD
3. Data Control Language (DCL)
   - Instrucțiuni pentru control al BD precum administrare, privilegii și tranzacții

De ce SQL?

1. DDL
   - Creare/ștergere bază de date, tabele și vederi
   - Creare/ștergere de secvențe și indecși
   - Creare/ștergere proceduri stocate și triggere
2. DML
   - Inserare/ștergere/modificare înregistrări
   - Interogare date
3. DCL
   - Adăugare/ștergere utilizatori, roluri și privilegii
   - Include TCL (Transactions Control Language)

SQL DDL – Tipuri de date

1. Depind de SGBD
2. Esențiale pentru optimizarea cerințelor de spațiu de stocare
3. În cele mai multe cazuri include și un interval specific de valori posibile
4. Specifică operatorii asociați tipurilor
5. Obs: tipuri de date din limbajul de programare client necesită mapare

Tipuri de date ANSI SQL

1. Șiruri de caractere
   - CHARACTER(n) sau CHAR(n): șir de *n* caractere cu lungime fixă
   - VARCHAR(n): șir cu lungime variabilă cu dimensiune maximă de *n* caractere
2. Numere
   - INTEGER, SMALLINT și BIGINT
   - FLOAT, REAL și DOUBLE PRECISION
   - NUMERIC(nr. cifre, zecimale) și DECIMAL(nr. cifre, zecimale)
3. Dată calendaristică și timp
   - DATE: pt. date calendaristice (ex. 2023-01-02)
   - TIME: pentru momente de timp (ex. 13:50:12)
   - TIMESTAMP: compus din DATE și TIME (ex. 2023-01-02 13:50:12)

Ex. Tipuri de date Oracle SQL

1. Șiruri de caractere
   - CHAR(n), VARCHAR2(n) , NCHAR(n), NVARCHAR2(n) – max. 2000 de caractere (conversie la codarea locală - ex. ASCII pt. Windows)
   - RAW, LONG RAW – șiruri binare (32KB, 2GB – fără conversie)
2. Numere
   - BINARY_INTEGER (4B), BINARY_FLOAT (4B), BINARY_DOUBLE (8B) - suportă valorile infinit și NaN
   - NUMERIC(nr. cifre[, zecimale]), nr. cifre<=38
3. Dată calendaristică și timp
   - DATE: între 1 Ian 4712 BC și 31 Dec 9999 AD
   - TIMESTAMP: include fracțiuni de secundă - un număr între 0 și 9 cifre (implicit este 6)
4. Obiecte de mari dimensiuni – large objects (LOB)
   - BLOB, CLOB, NCLOB: pentru binar/car/Unicode până la 128TB

Ex. Tipuri de date MySQL

1. Șiruri de caractere
   - CHAR(n) max 255 caractere, VARCHAR2(n) max 255 caractere
   - TEXT – 64KB caractere, LONGTEXT – 4GB caractere
   - ENUM(x,y,z, …), SET(x,y,z, …) liste și mulțimi de indentificatori
2. Numere
   - TINYINT (1B), SMALLINT (2B), INT(n) (4B), BIGINT(n) (8B), FLOAT(nr.cifre, zecimale) (4B), DOUBLE(nr.cifre,zecimale) (8B)
   - DECIMAL(nr.cifre, zecimale) numere în virgulă fixă
3. Dată calendaristică și timp
   - DATE: dată în intervalul '1000-01-01' și '9999-12-31'
   - DATETIME: de la '1000-01-01 00:00:00' la '9999-12-31 23:59:59'
   - TIME: de la '-838:59:59' la '838:59:59', YEAR: de la 1901 la 2155
   - TIMESTAMP: de la '1970-01-01 00:00:01' la '2038-01-09 03:14:07' cu inițializare automată
4. Obiecte de mari dimensiuni – large objects (LOB)
   - BLOB, LONGBLOB: pentru 64KB / 4GB

SQL DDL– Crearea unei baze de date

1. Instrucțiunea CREATE DATABASE
2. Depinde de SGBD-ul considerat
3. Sisteme mici – operație simplă. Ex. MySQL:

```
CREATE {DATABASE | SCHEMA}
      [IF NOT EXISTS] numeBD
      [DEFAULT] CHARACTER SET=charset_name ]
      [DEFAULT] COLLATE=collation_name ]
```

- IF NOT EXISTS – nu dă eroare dacă deja există
- CHARACTER SET – set caractere - diacritice (ex. latin1, utf8, cp1250…)
- COLLATE – specifică cum sunt comparate caracterele (UNICODE sau fără diacritice)

- Sisteme mari = operație complexă
  - Presupune configurarea spațiului de stocare (ex. Oracle ASM - automatic storage management etc.)

Oracle – Crearea unei baze de date

1. Proces complex, o BD poate poate găzdui mai multe scheme de date

   http://docs.oracle.com/cd/B28359_01/server.111/b28310/create003.htm#ADMIN11073

2. Pași necesari
   - Pas 1: Specificarea unui identificator de instanță (SID)
   - Pas 2: Setarea variabilelor de mediu
   - Pas 3: Alegerea unei metode de autentificare administrator
   - Pas 4: Crearea unui fișier cu parametrii pentru inițializare
   - Pas 5: (pt. Windows) Crearea unei instanțe
   - Pas 6: Conectarea unei instanțe
   - Pas 7: Crearea unui fișier cu parametrii pentru configurare server
   - Pas 8: Pornire instanță
   - Pas 9: Rulare comandă CREATE DATABASE
   - Pas 10: Crearea de Tablespaces

Ex. Oracle – CREATE DATABASE

1. Folosind Oracle Managed Files:

```
CREATE DATABASE numeBD
    USER SYS IDENTIFIED BY sys_password
    USER SYSTEM IDENTIFIED BY system_password
    EXTENT MANAGEMENT LOCAL
    DEFAULT TEMPORARY TABLESPACE temp
    UNDO TABLESPACE undotbs1
    DEFAULT TABLESPACE users;
```

2. O BD Oracle poate conține mai multe SCHEME – echivalent cu BD MySQL

SQL DDL – Crearea unei tabele

1. Utilizată pentru a crea structura de tabele a unei BD
2. Sintaxă

```
CREATE TABLE nume_tabelă (
    coloană1 tip(dimensiune) constrângeri,
    coloană2 tip(dimensiune) constrângeri,
    ....  );
```

3. Constrângeri
   - NOT NULL –nu permite valori NULL în acea coloană
   - UNIQUE – coloana poate conține doar valori unice
   - PRIMARY KEY - combină NOT NULL și UNIQUE
   - FOREIGN KEY – pentru integritatea referențială
   - CHECK – verifică o condiție logică generică
   - DEFAULT – o valoare implicită pentru acea coloană

Ex. Oracle – CREATE TABLE

```
CREATE TABLE nume_tabelă [(
        coloană1 tip(dimensiune) [NULL | NOT NULL],
        coloană2 tip(dimensiune) [NULL | NOT NULL],
       ....
       CONSTRAINT nume_c PRIMARY KEY (coloană1, ... coloană_n)
       CONSTRAINT nume_c
                FOREIGN KEY (coloană1, ... coloană_n)
                REFERENCES tabelă părinte (coloană1, ... coloană_n)
                [ON DELETE CASCADE | ON DELETE SET NULL]
       CONSTRAINT nume_c UNIQUE (coloană1, ... coloană_n)
       CONSTRAINT nume_c CHECK (expresie logică) ); ]
| [AS (SELECT …)]
```

- Constrângeri suplimentare
  - O cheie primară poate conține cel mult 32 de coloane
  - Unele dintre câmpurile care fac parte dintr-o constrângere unică pot conține valori NULL atâta timp cât combinația este unică

SQL DDL – ALTER TABLE (I)

1. Utilizată pentru modificare tabele
2. Permite redenumire, adăugare/ștergere/modificare câmpuri

```
ALTER TABLE nume_tabela_vechi
   RENAME TO nume_tabela_nou;

ALTER TABLE tabela
   ADD nume_colana tip [constrangeri coloană];

ALTER TABLE tabela
   DROP COLUMN nume_colana; sau RENAME COLUMN nume_nou;

ALTER TABLE tabela
   MODIFY nume_colana tip [constrangeri coloană];
```

3. Constrângeri coloană: NOT NULL, UNIQUE, DEFAULT etc.

SQL DDL – ALTER TABLE (II)

1. Importantă pentru gestionarea constrângerilor

```
ALTER TABLE tabela
    ADD CONSTRAINT nume_constrangere { CHECK,
                                       PRIMARY KEY sau FOREIGN KEY }
ALTER TABLE tabela
    DROP CONSTRAINT nume_constrangere;
ALTER TABLE tabela
{ENABLE | DISABLE}  CONSTRAINT nume_constrangere;
```

2. Ex.

```
ALTER TABLE marinari
    ADD CONSTRAINT c_varsta_limita CHECK varsta<70;
```

SQL DDL – Ștergerea obiectelor

1. Ștergerea unei BD
   MySQL: DROP DATABASE [IF EXISTS] numeBD
   Oracle: DROP DATABASE numeBD [INCLUDING BACKUPS]
   [NOPROMPT] (via **RECOVERY MANAGER**)
2. Ștergere tabele
   MySQL: DROP [TEMPORARY] TABLE [IF EXISTS] nume_tabelă
   Oracle: DROP TABLE nume_tabelă [CASCADE CONSTRAINTS]
   [PURGE]
   - PURGE nu permite ROLLBACK
3. Ștergere indecși
   MySQL: DROP INDEX nume ON tabelă
   Oracle: DROP INDEX nume

SQL DML – Interogări active

1. Permite modificarea datelor din tabele unei BD:
   - Adăugarea de înregistrări noi folosind INSERT
   - Modificare/corecție unor date din anumite coloane folosind UPDATE
   - Ștergerea înregistrărilor care nu mai sunt necesare cu DELETE

BD folosită în exemple (BD Port)

- **Tabelă Marinar**

mid	nume	rang	varsta
22	Ion	7	45
31	Horatiu	1	33
58	Oana	8	54
71	Constantin	9	55

- **Tabelă Barcă**

bid	nume	culoare
101	Cleo	Albastra
102	Triton	Rosie
103	Poseidon	Verde

- **Tabelă Rezervare**

rid	Mid	Bid	dată
22	31	101	10/03/2022
231	58	103	18/10/2022
71	31	101	22/10/2022

SQL DML - INSERT

1. Adăugare înregistrări:

INSERT INTO tabelă(listă câmpuri)
VALUES (listă valori);

INSERT INTO tabelă
VALUES (listă valori 1:1);

INSERT INTO tabelă
SET câmp1=exp1, ... ;

Obs: - Poziție aleatoare după inserare – nici o garanție că va fi la sfârșitul tabelei

- valorile lipsă = NULL (eroare dacă a fost specificată o constrângere *not null*)

2. Ex: INSERT INTO Marinari VALUES (58, ”Dinu”, 8, 54);

SQL DML – INSERT + SELECT

1. INSERT se poate combina cu SELECT pentru a copia datele dintr-o tabelă în alta:

```
INSERT INTO tabelă2
        SELECT * FROM tabelă1 WHERE condiție;
INSERT INTO tabelă2 (coloană1, coloană1, ...)
        SELECT coloană1, coloană1, ...
        FROM tabelă1 WHERE condiție;
```

Obs: - câmpurile din cele două tabele trebuie să corespundă ca și tip

2. Ex:

```
INSERT INTO Pensionari(nume, varsta)
        SELECT nume, varsta
        FROM Marinari WHERE varsta >=65;
```

SQL DML - UPDATE

1. Modificare/ corecție date:

```
UPDATE tabelă
    SET coloană1=exp1, ...
    WHERE condiție logică;
```

2. Ex:

```
UPDATE Marinari
    SET rang = rang+1
    WHERE rang<9 AND varsta>40;
```

3. Obs: pentru revenire: SET rang = rang – 1?

Nu toate expresiile sunt REVERSIBILE !!!

SQL DML - DELETE

1. Ștergere înregistrări:

   DELETE FROM tabelă

   WHERE condiție logică;

2. Ex:

   DELETE FROM Marinari

   WHERE varsta>65;

   !!! DELETE FROM Marinari;

3. Obs: nu există UNDO. Singura posibilitate este utilizarea unei tranzacții și ROLLBACK!

Ex. MySQL - DELETE

DELETE [LOW_PRIORITY] [IGNORE]
FROM tabelă
[WHERE condiție] [ORDER BY ...]
[LIMIT nr.rânduri];

- Parametrii
  - ORDER BY permite efectuarea într-o ordine prestabilită
  - LIMIT permite limitarea la un anumit număr de înregistrări
  - Pot fi folosite împreună
  - Ex. DELETE FROM log WHERE user = 'Joe'
    ORDER BY entry_time LIMIT 1;
    // șterge doar cea mai veche înregistrare din tabela **log**