**Notiuni fundamentale despre bazele de date si SQL (1)**

* Învatati modul in care este structurata o baza de date
* Învatati modul de proiectare a unei baze de date
* Învatati sa creati baze de date MySQL si tabele
* Învatati sa stergeti si sa modificati tabele dintr-o baza de date MySQL

Acesta lectie explica bazele de date relationale si modul de utilizare a acestora.  
Spre deosebire de fisiere, bazele de date prezinta multe avantaje, inclusiv o mai mare protectie a integritatii datelor si asigurarea partajarii datelor.  
In aceasta lectie va fi prezentat SQL, limbajul standard pentru crearea, accesul si manipularea bazelor de date relationale. Detaliile (cu precadere sintaxa SQL sunt cele referitoare la MySQL, cel mai popular limbaj de baze de date folosit cu PHP.

**1. Structura unei baze de date relationale**

O baza de date relationale stocheaza datele în tabele, iar fiecare tabel stocheaza in coloane informatii despre un anumit tip de element.   
In continuare puteti vedea un tabel caracteristic dintr-o baza de date relationala care prezinta personalitati istorice ale Americii.

|  |  |  |
| --- | --- | --- |
| PersoanaID | Nume | AnNastere |
| 001 | George Washington | 1732 |
| 002 | John Adams | 1735 |
| 003 | Thomas Jefferson | 1743 |

Primul rând al tabelului atribuie nume pentru fiecare coloana.  
Fiecare rând al tabelului, altul decât primul rând, descrie o singura personalitate. De exemplu, al doilea rând descrie o personalitati pe nume George Washington.  
Fiecare coloana, descrie un anumit atribut al personalitatii De exemplu, a doua coloana contine numele personalitatilor, iar a treia coloana contine anii în care s-au nascut acestia.  
Pentru a se putea face referire, mai usor, la un anumit rând al tabelului, se obisnuieste ca fiecare tabel sa contina o coloana care identifica în mod unic fiecare rând. Aceasta coloana se numeste "cheia primara" a tabelului. In exemplul prezentat, coloana numita "PersoanaID" serveste drept cheie primara.  
Daca nici o coloana nu contine o valoare unica pentru fiecare rând, se pot combina valorile mai multor coloane pentru a crea o "cheie primara compusa".  
O baza de date relationala se numeste astfel datorita capacitatii sale de a stabili relatii între date din mai multe tabele.   
Urmatorul exemplu prezinta doua tabele si o relatie între acestea.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | PersoanaID | Nume | AnNastere | | 001 | George Washington | 1732 | | 002 | John Adams | 1735 | | 003 | Thomas Jefferson | 1743 | | |  |  | | --- | --- | | Meserie | PersoanaID | | Arhitect | 003 | | General | 001 | | Filozof | 002 | |

Al doilea tabel contine informatii despre meseriile caracteristice ale persoanelor prezentate in exemplul anterior. Mai precis, tabelul identifica persoana cu o anumita meserie. Numele meseriei serveste drept cheie primara a tabelului, care mai contine, în afara de aceasta, o singura coloana.  
Coloana respectiva stocheaza atributul "PersoanaID" din primul tabel, al persoanei care cunoaste meseria descrisa de un anumit rând. De exemplu, angajatul cu numarul 003 este arhitect.   
Retineti ca "PersoanaID" este atât cheia primara a tabelului original, dar si o coloana din noul tabel. Coloana "PersoanaID" a noului tabel se numeste "cheie externa"; desi nu este cheia primara a noului tabel, este cheia primara a unui alt tabel.  
  
Aplicatia software care gazduieste o baza de date se numeste "sistem de gestiune a bazelor de date" (SGBD). Exista multe sisteme de gestiune a bazelor de date, printre cele mai populare asemenea sisteme se numara: DB2 , Interbase , MySQL , Oracle , Postgresql , SQL Server , Sybase .  
MySQL este cel mai popular sistem de gestiune a bazelor de date destinat utilizarii cu PHP, în mare masura deoarece este gratuit. Totusi, prin intermediul PHP este posibil accesul la aproape orice SGBD modern.  
  
Bazele de date relationale înteleg SQL (Structured Query Language), un limbaj relativ simplu, folosit pentru solicitarea datelor. In ciuda simplitatii sale, SQL este un limbaj foarte puternic, care poate obtine accesul la date stocate în mai multe tabele, poate filtra datele dorite si poate sorta, rezuma si afisa rezultatele.  
În general, bazele de date relationale îsi stocheaza datele într-un singur fisier sau catalog. Aceasta caracteristica de organizare faciliteaza administrarea datelor, deoarece executarea copiei de siguranta, respectiv restaurarea unui singur fisier sau catalog se realizeaza mai usor.  
Sistemele de gestiune a bazelor de date relationale necesita mai multe cicluri de procesor pentru a satisface o cerere de date decât cele necesare pentru accesul la un fisier normal, dar ofera protectie sporita a datelor, iar pentru accesul la distanta usureaza traficul deoarece majoritatea operatiilor sunt efectuate de programul SQL, astfel singurele date transmise in retea ar fi rezultatul cerut.

2. Proiectarea unei baze de date

Cand creati un tabel intr-o baza de date, este important sa tineti cont de "cheia primara" si sa va bazati pe aceasta. Coloanele dintr-un tabel trebuie sa se bazeze pe cheia primara în totalitatea sa.  
O alta operatie importanta este specificarea unui tip de date pentru fiecare coloana. Majoritatea bazelor de date relationale accepta urmatoarele tipuri de date generale:

Caracter

Întreg

Zecimal

Data si ora

Binar

Tabelul urmator rezuma tipurile de date cele mai frecvent utilizate, acceptate de MySQL si de majoritatea bazelor de date relationale. MySQL accepta multe alte tipuri de date, dar acestea sunt cele mai folosite.

|  |  |
| --- | --- |
| Principalele tipuri de date din MySQL | |
| **Tip de date** | **Descriere** |
| **BLOB** | - Date binare arbitrare, cu o lungime maxima de 65535 octeti |
| **CHAR(m)** | - Un sir de caractere de lungime fixa, cu un maxim de m caractere, unde m este mai mic decât 256. Pentru obtinerea lungimii dorite, se insereaza spatii finale |
| **DATE** | - O data în format an-luna-zi; de exemplu 2008-10-15 |
| **DECIMAL  DECIMAL(m,d)** | - Un numar zecimal, reprezentat sub forma de sir cu "m" cifre, din care "d" se afla la dreapta punctului zecimal. Daca "m" si "d" sunt omise, în mod prestabilit se vor utiliza valorile 10 si 0 |
| **DOUBLE  DOUBLE (m, d)** | - Un numar cu virgula mobila, cu dubla precizie, având o latime de afisare egala cu "m" si un numar de "d" cifre dupa virgula. |
| **FLOAT(m,d)** | - Un numar cu virgula mobila, cu simpla precizie, având o latime de afisare egala cu "m" si un numar de "d" cifre dupa virgula |
| **INTEGER  INTEGER UNSIGNED** | - Un întreg pe 32 de biti. Daca se specifica atributul UNSIGNED, domeniul de valori este cuprins între 0 si 4294967295; în caz contrar, domeniul este cuprins între valorile -2147483648 si 2147483647 |
| **NUMERIC  NUMERIC (m, d)** | - Similar cu DECIMAL |
| **REAL REAL(m, d)** | - Similar cu DOUBLE |
| **SMALLINT  SMALLINT UNSIGNED** | - Un întreg pe 16 biti. Daca se specifica atributul UNSIGNED, domeniul de valori este cuprins între 0 si 65535; în caz contrar, domeniul este cuprins între valorile –32768 si 32767 |
| **TIME  TIMESTAMP  TIMESTAMP(m)** | - Ora în format ora-minut-secunda; de exemplu, 08-20-00. O valoare de tip data si ora, în format an-luna-zi ora-minut-secunda; de exemplu, 1976-01-05 00:00:00. Aceasta reprezentare este similara celei returnate de functiile UNIX. Pt. stocare valoarea este transformata din timpul curent in UTC si transformata invers la solicitarea datei. |
| **VARCHAR(m)** | - Un sir caracter de lungime variabila, cu un maximum de "m" caractere, unde m este mai mic decât 256 pentru versiuni MySQL anterioare 5.0.3, iar pt. cele mai noi limita este 65 535 bytes. Spatiile finale au fost eliminate |

Iata cateva indicatii pentru selectarea tipurilor de date:

* - Alegeti BLOB ca tip pentru datele pe care nu trebuie sa le manipulati si la care nu veti obtine acces prin intermediul limbajului SQL.
* - Alegeti un tip data sau ora adecvat pentru coloanele care contin date calendaristice sau ore.
* - Alegeti un tip numeric pentru coloanele folosite pentru numere sau în calcule.
  + - Pentru cantitati foarte mari sau foarte mici, alegeti DOUBLE ca tip de date.
  + - Pentru coloane care contin numere fara parte zecimala de dimensiuni medii, alegeti SMALLINT sau INTEGER ca tip de date.
  + - Pentru alte coloane care contin date numerice, alegeti DECIMAL ca tip de date.
* - Alegeti CHAR sau VARCHAR ca tip pentru celelalte coloane, chiar si pentru cele care contin cifre, cum ar fi un cod postal.

**3. Crearea unei baze de date MySQL si a unui tabel**

Dupa atata teorie, sa trecem la lucruri mai practice.  
La început, o baza de date nu contine tabele. Pentru a crea o baza de date si apoi un tabel într-o baza de date, folositi un sub-limbaj SQL special, cunoscut sub numele de Data Definition Language (DDL).   
Puteti emite comenzi DDL si alte comenzi SQL prin intermediul unui interpretor SQL sau prin intermediul PHP. Pentru inceput va fi prezentat modul de emitere a comenzilor SQL folosind un interpretor SQL. In lectiile urmatoare va fi prezentat modul de emitere a comenzilor SQL utilizând PHP.  
Dupa ce aveti instalat MySQL, deschideti iterpretorul "MySQL Command Line Client" (asemanator cu, "Command Promt") in care scrieti comenzile SQL (*În general, SQL nu este sensibil la diferenta între majuscule si minuscule. Deci, puteti scrie comensile cum preferati, cu majuscule sau minuscule*).  
Ca regula, dupa scrierea comenzilor, la sfarsit trebue sa fie caracterul "punct si virgula" (;)  
Pentru a crea o baza de date, folositi urmatoarea comanda:

* **CREATE DATABASE nume\_db;**

Unde "nume\_db" este numele bazei de date care va fi creata.  
Inainte de a crea un tabel sau a lucra cu alte comenzi intr-o baza de date, trebuie sa intrati in baza de date existenta, acest lucru se face folosind comanda:

* **USE nume\_db;**

Unde "nume\_db" este numele bazei de date. Cu aceeasi comanda "USE nume\_db" se schimba si baza de date in care vrem sa lucram.  
Pentru a crea un tabel într-o baza de date, emiteti comanda "CREATE TABLE", care are urmatoarea forma:

* **CREATE TABLE tabel (coloana tip, coloana tip, …);**

Unde "tabel" este numele tabelului, "coloana" este numele unei coloane, "tip" este tipul datelor incluse în coloana, se poate specifica un numar nedefinit de coloane.  
De exemplu, comanda urmatoare creeaza un tabel numit "carte", care contine coloanele "carteid", "titlu" si "pret":

CREATE TABLE carte (carteid CHAR(10), titlu VARCHAR(255), pret decimal(5,2));

- În afara tipului de date, intre paranteze, puteti specifica si alte atribute optionale ale unei coloane:

* **NOT NULL** - Fiecare rând trebuie sa contina o valoare a coloanei asociate; valorile nule nu sunt permise.
* **DEFAULT valoare** - Daca nu este data o valoare a coloanei asociate, se va presupune valoarea specificata.
* **AUTO INCREMENT** - MySQL va repartiza în mod automat un numar de serie ca valoare a coloanei asociate.
* **PRIMARY KEY** - Coloana asociata este cheia primara a tabelului care o contine.

Iata un exemplu ceva mai complicat de creare a unui tabel, care foloseste unele din aceste atribute optionale:

CREATE TABLE carte (carteid CHAR(10) PRIMARY KEY,  
titlu VARCHAR(255) NOT NULL,  
pret DECIMAL(5,2) DEFAULT 50.00);

**4. Stergerea si modificarea unui tabel**

Stergerea unui tabel sau a unei coloane este un act irevocabil, datele fiind definitiv sterse.  
Pentru a sterge un tabel, scrieti urmatoarea comanda:

* **DROP TABLE tabel;**

Unde "tabel" este numele tabelului care va fi sters.  
Dupa crearea unui tabel, îl puteti modifica prin emiterea unei comenzi "ALTER TABLE", care are mai multe forme.  
Urmatoarea forma a comenzii va permite sa stergeti o coloana din tabel:

* **ALTER TABLE tabel DROP coloana;**

Unde "tabel" este numele tabelului care va fi modificat, iar "coloana" este numele coloanei care va fi stearsa.  
O alta forma a comenzii va permite sa adaugati o noua coloana în tabel:

* **ALTER TABLE tabel ADD coloana tip [optiuni];**

Unde "tabel" este numele tabelului care va fi modificat, "coloana" este numele coloanei care va fi adaugata, "tip" este tipul noii coloane, iar "[optiuni]" constituie toate optiunile dorite, precum PRIMARY KEY.  
De exemplu, pentru a adauga coloana "pretnou" la tabelul "carte", scrieti comanda:

ALTER TABLE carte ADD pretnou DECIMAL(5,2) DEFAULT 50.00;

Pentru a vedea toate bazele de date create in MySQL, folositi urmatoarea comanda:

* **SHOW DATABASES;**

Pentru a vedea toate tabelele dintr-o baza de date din MySQL, folositi urmatoarea comanda:

* **SHOW TABLES;**

Pentru a vedea toate coloanele dintr-un tabel, si atributele acestora, folositi urmatoarea comanda:

* **SHOW COLUMNS FROM tabel;**

**2. Accesul la datele dintr-o baza de date, interogarile SQL**

În afara de Data Definition Language, SQL include Data Manipulation Language (DML). DML va permite sa formati interogari, care obtin accesul la datele aflate într-o baza de date relationala si raporteaza aceste date. De asemenea, puteti folosi DML pentru a insera, actualiza si sterge rândurile dintr-un tabel.  
Cea mai elementara forma de interogare este comanda "SELECT" simpla.  
Cea mai simpla interogare raporteaza toate coloanele din toate rândurile unui tabel. Interogarea are urmatoarea forma:

* **SELECT \* FROM tabel;**

- Unde "tabel" este numele tabelului la care se va obtine accesul.  
Aceasta comanda va afisa un tabel cu fiecare rând pe o linie separata. Datele de iesire includ numele coloanelor si linii care separa coloanele.  
Daca doriti sa selectati numai anumite coloane sau sa raportati coloanele într-o anumita ordine, puteti folosi urmatoarea forma a comenzii "SELECT":

* **SELECT coloana1, coloana2 FROM tabel;**

- Unde "tabel" este numele tabelului, iar "coloana1" si "coloana2" sunt coloanele la care se va obtine accesul si al caror continut va fi afisat.   
Puteti specifica mai multe coloane, separate prin virgula.  
Daca vreti sa fie selectate numai acele rânduri care satisfac un anumit criteriu, clauza "WHERE" va permite sa specificati o conditie; rândurile care nu satisfac conditia nu sunt raportate.  
De exemplu, iata o interogare care raporteaza un singur rând:

SELECT titlu, pret FROM carte WHERE carteid=1;

Forma conditiilor folosite în sub-limbajul DML al limbajului SQL este similara cu aceea a conditiilor PHP. Puteti folosi oricare din urmatorii operatori relationali:

* **=**       Egalitate
* **<>**       Inegalitate
* **!=**       Inegalitate, diferit de
* **<**       Mai mic decat
* **>**       Mai mare decat
* **<=**       Mai mic sau egal cu
* **=>**       Mai mare sau egal cu

Puteti compara valoarea unei coloane cu aceea a altei coloane, respectiv valoarea unei coloane cu o valoare string. Valorile string SQL (sirurile) trebuie sa fie incluse între ghilimele simple, nu între ghilimelele duble.

*Când unei coloane nu i-a fost repartizata nici o valoare, SQL îi atribuie valoarea speciala NULL. De asemenea, programatorii pot atribui în mod explicit valoarea NULL unei coloane.   
Comparatiile obisnuite cu valori NULL, care folosesc operatorii de (in)egalitate, vor returna un rezultat fals. Totusi, puteti folosi operatorul special "****<=>****". care compara valorile tinând cont de valoarea NULL. Daca folositi acest operator pentru a compara doua valori NULL, se obtine un rezultat adevarat.*

SQL include si numerosi operatori de comparatie non-algebrici:

|  |  |
| --- | --- |
| **Operator** | **Descriere** |
| **x BETWEEN y AND z** | Adevarat, daca valoarea lui x este cuprinsa între valorile lui y si z |
| **x LIKE y** | Adevarat daca valoarea lui x este echivalenta cu modelul y |
| **x NOT LIKE y** | Adevarat daca valoarea lui x nu este echivalenta cu modelul y |
| **x IN (y1, y2)** | Adevarat daca valoarea lui x este un membru al listei y1, y2. Lista poate contine unul, doi sau mai multi membri |
| **x NOT IN (y1, y2)** | Adevarat daca valoarea lui x nu este un membru al listei y1, y2. Lista poate contine unul, doi sau mai multi membri |
| **x IS NULL** | Adevarat daca x are valoarea NULL |
| **x IS NOT NULL** | Adevarat daca x nu are valoarea NULL |

Sub-limbajul folosit pentru specificarea modelelor asociate operatorului "LIKE" este diferit de cel folosit de PHP. Meta-caracterul "**%**" corespunde unui numar de zero sau mai multe caractere, iar meta-caracterul "**\_**" corespunde unui singur caracter.  
Modelele, precum sirurile, sunt incluse între ghilimele simple.  
- De exemplu, modelul '**%ar%**' corespunde oricarui sir care contine sub-sirul 'ar', inclusiv siruri precum 'ar', 'arc' si 'un zar'.  
Pentru a plasa un caracter "%" sau "\_" într-un sir test, folositi secventa "\%" sau "\\_"; ca în PHP.  
Ca si PHP, sub-limbajul DML din SQL va permite sa formati expresii logice care combina expresiile relationale. Puteti folosi oricare din urmatorii operatori logici:

* **AND , &&** - SI, adevarat daca ambii operanzi sunt adevarati
* **OR , ||** - SAU, adevarat daca un operand este adevarat
* **NOT , !** - NU, adevarat daca operandul este fals

De exemplu, urmatoarea interogare raporteaza rândurile care au pretul de carte mai mare decât 11 sau al caror titlu include sub-sirul 'Curs':

SELECT titlu, pret FROM carte  
    WHERE pret>11 OR titlu LIKE ’%Curs%’;

*Pentru a compara o valoare cu o expresie regulata în MySQL, folositi urmatoarea forma:****x REGEXP y****, unde "x" este valoarea care va fi testata, iar "y" este o expresie regulata, delimitata prin ghilimele simple.*

**3. Modificarea datelor dintr-o baza de date**

Sub-limbajul SQL, Data Manipulation Language, include comenzi care va permit sa inserati rânduri noi într-un tabel, sa actualizati una sau mai multe coloane ale rândurilor existente, respectiv sa stergeri rânduri dintr-un tabel.  
Pentru a insera un nou rând într-un tabel, folositi comanda "INSERT", care are urmatoarea forma:

* **INSERT INTO tabel VALUES (valoare1, valoare2);**

- Unde "tabel" este numele tabelului la care se va adauga rândul.  
- "valoare1" este valoarea pentru prima coloana din tabel, iar "valoare2" este valoarea celei de-a doua coloane din tabel. Se pot da mai mult sau mai putin de doua valori; numarul valorilor date ar trebui sa fie egal cu acela al coloanelor din tabel.  
O coloana poate primi si valoarea "NULL", cu exceptia situatiilor când definitia coloanei nu permite.  
O forma mai buna (si recomandata) a comenzii INSERT specifica numele coloanelor carora le sunt atribuite valorile:

* **INSERT INTO tabel (coloana1, coloana2) VALUES (valoare1, valoare2);**

Unde coloana denumita "coloana1" primeste valoarea "valoare1", iar coloana denumita "coloana2" primeste valoarea "valoare2".  
Pot fi specificate mai mult sau mai putin de doua coloane si valori. Numarul coloanelor specificate trebuie sa fie egal cu numarul valorilor specificate.  
Coloanele care nu sunt denumite în comanda "INSERT" si care nu au o valoare prestabilita (DEFAULT) primesc valoarea NULL, cu exceptia situatiilor când valoarea respectiva nu este permisa (în acest caz, comanda INSERT esueaza).  
Daca doriti sa adaugati mai multe randuri intr-un tabel, intr-o singura inserare, puteti folosi urmatoarea forma a comenzii INSERT:

* **INSERT INTO tabel (coloana1, coloana2)   
      VALUES (valoare1, valoare2),   
          (valoare3, valoare4),   
          (valoare5, valoare6);**

Pentru a modifica valoarea unui rând sau mai multor rânduri existente într-un tabel, folositi o comanda "UPDATE", care are urmatoarea forma:

* **UPDATE tabel SET coloana1=valoare1, coloana2=valoare2   
      WHERE conditie;**

- Unde "tabel" este numele tabelului ale carui rânduri urmeaza sa fie modificate, "coloana1" este numele primei coloane care urmeaza a fi modificata, "valoare1" este valoarea care va fi repartizata în "coloana1", "coloana2" este numele celei de-a doua coloane care urmeaza a fi modificata, "valoare2" este valoarea care va fi repartizata în "coloana2", iar "conditie" identifica rândul sau rândurile care urmeaza a fi actualizate, dupa o anumita conditie.  
*Poate fi actualizat un numar mai mare sau mai mic de coloane. Daca se doreste ca toate randurile coloanei sa fie actualizate, clauza "WHERE" poate fi omisa.*  
De exemplu, comanda urmatoare modifica din coloana "titlu" numele asociat cartii in "Curs2", in randul unde coloana "pret" are valoarea 5.

UPDATE carte SET titlu=’Curs2’  
    WHERE pret=5;

Pentru a sterge un rând dintr-un tabel, folositi comanda "DELETE", care are urmatoarea forma:

* **DELETE FROM tabel WHERE conditie;**

Daca se vrea a fi sterse toate rândurile tabelului, clauza WHERE poate fi omisa.  
De exemplu, comanda urmatoare sterge rândul din tabelul "carte" asociat titlului "Curs2":

DELETE FROM carte WHERE titlu='Curs2';

**4. Sortarea si gruparea**

Deseori, este important ca datele sa fie raportate într-o anumita secventa. Puteti specifica ordinea de raportare a rezultatelor interogarii folosind clauza "ORDER BY", care are urmatoarea forma:

* **ORDER BY valoare**

Daca se vor folosi mai multe câmpuri de sortare, acestea vor fi separate prin virgula.  
Daca doriti sa indicati o sortare descendenta, în locul uneia ascendente, specificati "DESC" dupa valoare.  
De exemplu, pentru a ordona toate cartile în functie de pret, de la cel mai mare la cel mai mic, respectiv dupa titlu pentru un pret egal, puteti folosi urmatoarea interogare:

SELECT pret, titlu FROM carte   
    ORDER BY pret DESC, titlu;

Pentru a include numai titlurile care au un pret mai mare de 50, adaugati o clauza WHERE la comanda SELECT:

SELECT pret, titlu FROM carte   
    WHERE pret>50   
    ORDER BY pret DESC, titlu;

SQL include functii care va permit sa raportati valori diferite, valori utile, precum un numar al rândurilor tabelului. Iata cele mai importante functii in acest sens:

|  |  |
| --- | --- |
| **Functie** | **Descriere** |
| **count(\*)** | - Numarul rândurilor din tabel |
| **count(coloana)** | - Numarul rândurilor din tabel care contin o valoare diferita de NULL în coloana specificata |
| **count(distinct coloana)** | - Numarul valorilor distincte diferite de NULL care apar în coloana specificata |
| **avg(coloana)** | - Valoarea mijlocie (medie) a coloanei numerice specificate |
| **min(coloana)** | - Valoarea minima din coloana specificata |
| **max(coloana)** | - Valoarea maxima din coloana specificata |
| **sum(coloana)** | - Suma valorilor din coloana specificata |

De exemplu, interogarea urmatoare raporteaza numarul cartilor si pretul mediu al acestora:

SELECT count(\*), avg(pret) FROM carte;

- Datele de iesire vor fi afisate pe o singura linie.  
Clauza SQL "AS" specifica un nume nou pentru o coloana sau expresie. Numele specificat este folosit ca titlu în rapoartele SQL. Clauza "AS" este utila în lucrul cu functiile din tabelul anterior, va ajuta sa identificati mai usor datele de iesire.  
De exemplu, interogarea precedenta poate fi rescrisa pentru a include o clauza "AS", astfel:

SELECT count(\*) AS Carte\_Numar,   
    avg(pret) AS Pret\_Mediu   
    FROM carte;

O alta clauza utila este "GROUP BY", aceasta are urmatoarea forma:

* **GROUP BY coloana-sortare**

- Unde "coloana-sortare" este numele sau valoarea unei coloane, specificata într-o clauza "ORDER BY", care trebuie sa urmeze dupa clauza "GROUP BY".  
De exemplu, avem un tabel "carte" cu 4 coloane (carteid, titlu, anul si pret), interogarea urmatoare raporteaza numarul cartilor si pretul mediu pentru fiecare an (anul) în parte:

SELECT count(\*), avg(pret) FROM carte   
    GROUP BY anul   
    ORDER BY anul;

Pentru a include în datele de iesire numai grupurile selectate, specificati clauza "HAVING" imediat dupa clauza "GROUP BY". Clauza "HAVING" are urmatoarea forma:

* **HAVING conditie**

De exemplu, pentru a include numai cartile al caror atribut "anul" are valoare mai mare decât 1980, srieti urmatoarea interogare:

SELECT count(\*), avg(pret) FROM carte   
    GROUP BY anul   
    HAVING anul>1980   
    ORDER BY anul;

Mai jos sunt afisati operatorii matematici folositi in SQL:

* **+**       Adunare   
  **-**      Scadere   
  **\***      Inmultire   
  **/**      Impartire   
  **|**       SAU, la nivel de bit   
  **&**      SI la nivel de bit   
  **>>**       Deplasare la dreapta   
  **<<**       Deplasare la stanga   
  **\_**       Complement la nivel de bit