Изпитна тема № 10: Бази данни

Агрегиращи функции. Групиране на данни. Скаларни функции, транзакции, съхранени процедури, тригери.

Агрегиращите функции позволяват извличането на обобщена информация от базата данни. Най използваните агрегиращи функции са следните:

* COUNT – Изчислява броя на ненулевите елементи
* SUM – Изчислява сумата на изразите в дадена колона
* AVG – Изчислява средноаритметично в дадена колона
* MIN – Намира минимална НМС в дадена колона
* MAX – Намира максималната НГС в дадена колона

В следния пример е създадена заявка, която връща средната стойност на колоната Salary от таблицата Employees.

**SELECT** AVG(SALARY)

**FROM** Employees;

Групирането на данни се използва, когато е необходимо да се използва агрегатна функция, не върху цялата таблица, а само върху дадени групи от нея. За групиране се използва клаузата **GROUP BY.** Целта е агрегатната функция да се приложи върху множество по-малки групи, които в последствие ще се обединят в една обща. В следния пример се демонстрира заявка, която ще извлече средната стойност на всяка заплата във всеки отделен град.

**SELECT** City,

AVG(Salary)

**FROM** Cities

**GROUP BY** City;

Скаларните функции в базите данни са функции, които приемат параметри и връщат една стойност. Те се използват за обработка на данни в колони или редове. Примери за скаларно функции са:

* UPPER – Приема низ, връща низа само с главни букви.
* LOWER – Приема низ, връща низа само с малки букви.
* LEN – Приема низ, връща неговата дължина.
* CONCAT – Конкатенира два или повече низа
* DATEPART – Приема частта от датата, която трябва да се вземи, самата дата и връща само тази част, която е поискана.

В следния пример е създадена заявка която връща месеца при подадена дата.

**SELECT** DATEPART(year, ‘2022-05-10 08:30:00’) **AS** Year

Транзакциите в SQL се използват за групиране на една или повече операции в едно цяло, където трябва да бъдат изпълнени успешно всички операции, за да се приложат промените в базата данни. В случай на грешка дори при една от операции се отменят всички. Транзакции се използват за да се постигне автоматизиране, систематизиране, изолация и устойчивост. За да се създаде транзакция се използва ключовата дума **BEGIN** **TRANSACTION**, с която се поставя началото на транзакцията. След това следват командите, а накрая се използва **COMMIT** за успешно завършване на транзакцията или **ROLLBACK** при неуспешно завършване, съответно отменяне на всички промени. Транзакции са необходими, когато се работи с множество сложни операции и трябва консистентност на данните.

Съхранените процедури (stored procedures) в SQL са предварително компилирани блокове от код, които се съхраняват в базата данни и могат да бъдат използвани повторно. Те представляват набор от SQL операции, който се изпълнява от базата данни при извикване. Еднo от основните предимства на съхранените процедури е оптимизацията на изпълнението. Тъй като се компилират предварително, те могат да бъдат изпълнявани по-ефективно и по-бързо от заявки, които се изпращат от клиента до сървъра.

В следния пример е декларирана процедура, която приема един параметър от тип INT и връща всички елементи, които отговарят на дадено условие.

**CREATE PROCEDURE** GetCustomerOrders

@CustomerID **INT**

**AS**

**BEGIN**

**SELECT** \* **FROM** Orders

**WHERE** CustomerID **=** @CustomerID**;**

**END;**

Тригерите (triggers) в SQL са обекти, които се активират автоматично от базата данни при определени събития или действия в таблиците. Те са асоциирани с конкретна таблица и се изпълняват автоматично при добавяне, промяна или изтриване на данни в тази таблица. Тригерите позволяват вмъкване на допълнителна логика, валидации или манипулации на данните, преди или след извършването на съответните операции в таблицата.

Обобщен план (Опорни думи)

1. **Агрегиращи функции**
   * COUNT
   * SUM
   * AVG
   * MIN
   * MAX
2. **Групиране на данни**
   * Когато се използва агрегатна функция върху група от елементи в цялата таблица.
   * GROUP BY
   * В последствие малките групи се обединяват
3. **Скаларни функции** 
   * UPPER
   * LOWER
   * LEN
   * CONCAT
   * DATEPART
4. **Транзакции** – групиране на една или повече команди. Или са успешни всички **COMMIT**, или нито една **ROLLBACK, BEGIN TRANSACTION** за начало
5. **Съхранени процедури** – предварително компилирани блокове от код, съхраняват се в базата данни. Използват се при извикване **EXECUTE.** Оптимизация на изпълнението. CREATE PROCEDURE, PARAMS, AS, BEGIN, QUERY, END;
6. **Тригери** – обекти, които се активират и изпълняват автоматично при възникване на определени събития

Казус

Напишете заявка, която показва най-ниската заплата за всеки отдел. Списъкът да е подреден по заплата във низходящ ред и по длъжност в азбучен.

**SELECT** e.JobTitle,

e.Salary

**FROM** Employees **AS** e

**JOIN** Departments **AS** d

**ON** e.DepartmentID **=** d.DepartmentID

**WHERE** e.Salary **= (**

**SELECT MIN(**emp.Salary**)**

**FROM** Employees **AS** emp

**WHERE** e.DepartmentID **=**

emp.DepartmentID

**)**

**ORDER BY** e**.**Salary **DESC,** e.JobTitle **ASC**

ЗАДАЧА 2:

**SELECT** e.FirstName,

e.LastName,

e.Salary

**FROM** Employees e

**WHERE** e.Salary = (

**SELECT** **MAX**(emp.Salary)

**FROM** Employees **AS** emp

**WHERE** e.DepartmentID =

emp.DepartmentID

)

**ORDER** **BY** e.FirstName **DESC**

ЗАДАЧА 3:

**SELECT** d.[Name]

**FROM** Employees **AS** e

**JOIN** Departments **AS** d

**ON** e.DepartmentID = d.DepartmentID

**GROUP** **BY** d.[Name]

**HAVING** **COUNT**(\*) > 10