ИИКС

Лабораторная работа №2: «Работа с данными. Простые запросы на выборку»

Грущин Илья

Б21-515

2024 г

Задание

1. Подготовить данные заполнения таблиц, разработанных в рамках лабораторной работы 1-1. Занести их в базу данных при помощи SQL-инструкции INSERT. Можно использовать клиент sqlite3.exe или любой другой на Ваше усмотрение;
2. Разработать и реализовать не менее 8 SQL-запросов в соответствии с легендой в Вашей предметной области. Какие данные из таблиц могут пригодиться Вашим сотрудникам, и при каких условиях? При реализации запросов необходимо использовать возможности, описанные в следующем разделе;
3. Выполнить эти запросы в реализованной и заполненной базе данных, сохранить результаты выполнения; Реализовать разработанную схему данных средствами языка SQL в системе управления базами данных SQLite3.

Список выполненных простых запросов SQL с описаниями их смысла и ожидаемых результатов, а также результатами их выполнения на хранящихся в БД данных

Запрос 1:

Запрос: SELECT \* FROM Games;

Значение: Полная выборка всех игр

Ожидаемый результат: записи обо всех играх из БД

Результат запроса, преобразованный в таблицу средствами MS Word:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| "game\_id" | "title" | "release\_year" | "developer\_id" | "publisher\_id" | "age\_rating" | "peak\_players" |
| 1 | "Fortnite" | 2017 | 1 | 5 | "13+" | 12000000 |
| 2 | "Half-Life: Alyx" | 2020 | 2 | 4 | "17+" | 16000 |
| 3 | "The Witcher 3" | 2015 | 3 | 6 | "18+" | 103000 |
| 4 | "Assassin`s Creed Odyssey" | 2018 | 4 | 3 | "17+" | 62000 |
| 5 | "Call of Duty: Modern Warfare" | 2019 | 7 | 2 | "18+" | 2000000 |
| 6 | "Call of Duty: Black Ops" | 2010 | 8 | 2 | "18+" | 900000 |
| 7 | "Dragon Age: Inquisition" | 2014 | 5 | 1 | "17+" | 4000 |
| 8 | "Crusader Kings III" | 2020 | 6 | 7 | "16+" | 100000 |
| 9 | "Helldivers" | 2015 | 9 | 8 | "16+" | 50000 |
| 10 | "Spider-Man" | 2018 | 10 | 8 | "16+" | 500000 |

Запрос 2:

Запрос: SELECT title, release\_year FROM Games WHERE release\_year > 2016;

Значение: Выберем все игры, выпущенные после 2016 года

Ожидаемый результат: для каждой подходящей игры: название и год выпуска

Результат запроса, преобразованный в таблицу средствами MS Word:

|  |  |
| --- | --- |
| "title" | "release\_year" |
| "Fortnite" | 2017 |
| "Half-Life: Alyx" | 2020 |
| "Assassin`s Creed Odyssey" | 2018 |
| "Call of Duty: Modern Warfare" | 2019 |
| "Crusader Kings III" | 2020 |
| "Spider-Man" | 2018 |

Запрос 3:

Запрос: SELECT name, games\_released FROM Developers ORDER BY games\_released DESC;

Значение: сводка данных о разработчиках отсортированная в порядке убывания количества выпущенных игр

Ожидаемый результат: название студии разработчиков и количество выпущенных игр

Результат запроса, преобразованный в таблицу средствами MS Word:

|  |  |
| --- | --- |
| "name" | "games\_released" |
| "Paradox Development Studio" | 100 |
| "Ubisoft Montreal" | 50 |
| "Epic Games" | 40 |
| "Treyarch" | 30 |
| "BioWare" | 30 |
| "Valve Corporation" | 30 |
| "Insomniac Games" | 30 |
| "Infinity Ward" | 25 |
| "CD Projekt Red" | 20 |
| "Arrowhead Game Studios" | 10 |

Запрос 4:

Запрос: SELECT title, ROUND(peak\_players/1000) thousands\_of\_players\_in\_peak FROM Games

Значение: Выборка игр с выраженным в тысячах количеством игроков в пике

Ожидаемый результат: название игры, пиковое количество одновременных игроков в тысячах.

Результат запроса, преобразованный в таблицу средствами MS Word:

|  |  |
| --- | --- |
| "title" | "thousands\_of\_players\_in\_peak" |
| "Fortnite" | 12000 |
| "Half-Life: Alyx" | 16 |
| "The Witcher 3" | 103 |
| "Assassin`s Creed Odyssey" | 62 |
| "Call of Duty: Modern Warfare" | 2000 |
| "Call of Duty: Black Ops" | 900 |
| "Dragon Age: Inquisition" | 4 |
| "Crusader Kings III" | 100 |
| "Helldivers" | 50 |
| "Spider-Man" | 500 |

Запрос 5:

Запрос: SELECT g.title, ROUND(AVG(r.score)) AS average\_score FROM Games g JOIN Ratings r ON g.game\_id = r.game\_id GROUP BY g.title;

Значение: Найдем среднюю оценку каждой игры.

Ожидаемый результат: название игры и её средняя оценка

Результат запроса, преобразованный в таблицу средствами MS Word:

|  |  |
| --- | --- |
| "title" | "average\_score" |
| "Crusader Kings III" | 9 |
| "Dragon Age: Inquisition" | 9 |
| "Half-Life: Alyx" | 8 |
| "Call of Duty: Black Ops" | 7 |
| "Assassin`s Creed Odyssey" | 8 |
| "Fortnite" | 9 |
| "Helldivers" | 8 |
| "Call of Duty: Modern Warfare" | 9 |
| "The Witcher 3" | 10 |
| "Spider-Man" | 10 |

Запрос 6:

Запрос: SELECT DISTINCT age\_rating FROM Games;

Значение: Выберем все возрастные рейтинги без повторений.

Ожидаемый результат: столбец возрастных рейтингов

Результат запроса:

"age\_rating"

"16+"

"18+"

"13+"

"17+"

Запрос 7:

Запрос: SELECT (137\*548) AS result;

Значение: Псевдо-выборка для вычисления произведения.

Ожидаемый результат: число, равное 137\*548

Результат запроса:

"result"

75076

Запрос 8:

Запрос: VACUUM FULL ANALYZE;

Значение: Очистить таблицы от удаленных значений.

Ожидаемый результат: Таблицы будут занимать меньше места на диске.Результат запроса:

VACUUM

Query returned successfully in 2 secs 544 msec.

Листинг использованных инструкций SQL

-- Запрос 1: Полная выборка всех игр

SELECT \* FROM Games;

-- Запрос 2: Условная выборка игр, выпущенных после 2016 года

SELECT title, release\_year FROM Games WHERE release\_year > 2016;

-- Запрос 3: Упорядоченная выборка разработчиков по количеству выпущенных игр

SELECT name, games\_released FROM Developers ORDER BY games\_released DESC;

-- Запрос 4: Выборка игр с выраженным в тысячах количеством игроков в пике

SELECT title, ROUND(peak\_players/1000) thousands\_of\_players\_in\_peak FROM Games

-- Запрос 5: Запрос, который вычисляет среднюю оценку каждой игры и округляет до целого

SELECT g.title, ROUND(AVG(r.score)) AS average\_score FROM Games g JOIN Ratings r ON g.game\_id = r.game\_id GROUP BY g.title;

-- Запрос 6: Выборка уникальных возрастных рейтингов

SELECT DISTINCT age\_rating FROM Games;

-- Запрос 7: Псевдо-выборка для оценки выражения

SELECT (137\*548) AS result;

-- Запрос 8: Очистка таблиц на диске и статистики от удаленных записей.

VACUUM FULL;

Заключение

В данной работе изучалась работа с базой данных, язык запросов SQL, его возможности и команды.

На первом этапе работы подготовлены тестовые данные для вставки в таблицу. Также дополнен скрипт init\_db из предыдущей работы, который стирает все таблицы, заново создает их таблицы и заполняет тестовыми данными. Все дальнейшие запросы вводились вручную с помощью SQL Query Tool в клиенте pgAdmin 4.

На втором этапе работы изучались: полная выборка, условная выборка, упорядочивание данных, использование операторов, использование агрегатных функций, выборка уникальных записей, псевдо-выборка и очистка таблицы от старых удаленных записей

Все запросы успешны исполнены.

Приложение

Файл create.sql, реализующий создание таблицы.

CREATE TABLE Developers (

    developer\_id SERIAL PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    employee\_count INT,

    games\_released INT

);

CREATE TABLE Publishers (

    publisher\_id SERIAL PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    games\_published INT,

    average\_annual\_revenue DECIMAL(15, 2)

);

CREATE TABLE Genres (

    genre\_id SERIAL PRIMARY KEY,

    genre\_name VARCHAR(100) NOT NULL

);

CREATE TABLE Games (

    game\_id SERIAL PRIMARY KEY,

    title VARCHAR(100) NOT NULL,

    release\_year INT,

    developer\_id INT REFERENCES Developers(developer\_id),

    publisher\_id INT REFERENCES Publishers(publisher\_id),

    age\_rating VARCHAR(10),

    peak\_players INT

);

CREATE TABLE GameGenres (

    game\_id INT,

    genre\_id INT,

    PRIMARY KEY (game\_id, genre\_id),

    FOREIGN KEY (game\_id) REFERENCES Games(game\_id),

    FOREIGN KEY (genre\_id) REFERENCES Genres(genre\_id)

);

CREATE TABLE Users (

    user\_id SERIAL PRIMARY KEY,

    username VARCHAR(100) NOT NULL,

    email VARCHAR(100) UNIQUE NOT NULL,

    age INT,

    gender VARCHAR(10)

);

CREATE TABLE Ratings (

    rating\_id SERIAL PRIMARY KEY,

    game\_id INT REFERENCES Games(game\_id),

    user\_id INT REFERENCES Users(user\_id),

    score INT,

    review TEXT,

    rating\_date DATE

);

CREATE TABLE DeveloperPublisherRelationships (

    relationship\_id SERIAL PRIMARY KEY,

    developer\_id INT NOT NULL,

    publisher\_id INT NOT NULL,

    relationship\_type VARCHAR(50),

    start\_date DATE,

    end\_date DATE,

    FOREIGN KEY (developer\_id) REFERENCES Developers(developer\_id),

    FOREIGN KEY (publisher\_id) REFERENCES Publishers(publisher\_id)

);