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.

```sql
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)
);
```