# Опишите разницу между динамическими и статическими SQL выражениями.

**Mô tả sự khác biệt giữa các câu lệnh SQL động và tĩnh.**

**Статические SQL** выражения определяются на этапе компиляции программы. В этом случае, запрос уже заранее известен, и его можно проверить на наличие ошибок и оптимизировать. В статическом SQL выражении, все параметры и элементы запроса, такие как имена таблиц, столбцов, условия, задаются явно в самом запросе. Такие запросы являются более быстрыми и безопасными, так как компилятор может проверить запрос на наличие ошибок и оптимизировать его выполнение. Примером статического SQL может быть запрос типа:

Các biểu thức SQL tĩnh được xác định trong quá trình biên dịch chương trình. Trong trường hợp này, yêu cầu đã được biết trước và nó có thể được kiểm tra lỗi và tối ưu hóa. Trong biểu thức SQL tĩnh, tất cả các tham số và thành phần truy vấn, chẳng hạn như tên bảng, tên cột và điều kiện, đều được chỉ định rõ ràng trong chính truy vấn đó. Các truy vấn như vậy nhanh hơn và an toàn hơn vì trình biên dịch có thể kiểm tra lỗi truy vấn và tối ưu hóa việc thực thi nó. Một ví dụ về SQL tĩnh sẽ là một truy vấn như:

SELECT \* FROM customers WHERE id = 100;

**Динамические SQL** выражения, напротив, определяются и выполняются во время выполнения программы. В этом случае, запрос создается динамически, используя параметры, полученные во время выполнения программы. Это означает, что запросы могут меняться в зависимости от входных данных. Динамические SQL запросы могут быть менее безопасными, так как могут быть подвержены SQL-инъекциям. Примером динамического SQL запроса может быть:

Mặt khác, các câu lệnh SQL động được xác định và thực thi trong quá trình thực thi chương trình. Trong trường hợp này, yêu cầu được tạo động bằng cách sử dụng các tham số thu được trong quá trình thực hiện chương trình. Điều này có nghĩa là các truy vấn có thể thay đổi tùy thuộc vào dữ liệu đầu vào. Các truy vấn SQL động có thể kém an toàn hơn vì chúng có thể dễ bị tiêm SQL. Một ví dụ về truy vấn SQL động có thể là:

EXECUTE IMMEDIATE 'SELECT \* FROM customers WHERE id = ' || cust\_id;

В данном примере, запрос формируется динамически в зависимости от параметра cust\_id.

Trong ví dụ này, yêu cầu được tạo động tùy thuộc vào tham số cust\_id.

Основное отличие между статическими и динамическими SQL запросами заключается в том, что статические запросы выполняются быстрее, более безопасны и оптимизированы на этапе компиляции, в то время как динамические запросы предлагают большую гибкость в написании запросов и могут меняться в зависимости от входных данных.

Sự khác biệt chính giữa truy vấn SQL tĩnh và động là truy vấn tĩnh nhanh hơn, an toàn hơn và được tối ưu hóa tại thời điểm biên dịch, trong khi truy vấn động mang lại sự linh hoạt hơn trong việc viết truy vấn và có thể thay đổi dựa trên dữ liệu đầu vào.

# Опишите хотя бы один из методов анализа исходного кода, который позволяет обнаружить уязвимые места для внедрения SQL инъекции

Mô tả ít nhất một phương pháp phân tích mã nguồn cho phép bạn phát hiện các lỗ hổng để chèn SQL

Один из методов анализа исходного кода для обнаружения уязвимостей инъекций SQL называется анализом статического кода. Анализ статического кода включает анализ исходного кода приложения без его выполнения. Этот подход может помочь обнаружить уязвимости инъекции SQL, выявляя шаблоны кода и конструкции, которые могут позволить злоумышленнику вводить злонамеренный код SQL в приложение.

Một phương pháp phân tích mã nguồn để phát hiện các lỗ hổng SQL SQL được gọi là phân tích mã tĩnh. Phân tích mã tĩnh liên quan đến việc phân tích mã nguồn của ứng dụng mà không cần thực thi nó. Cách tiếp cận này có thể giúp phát hiện các lỗ hổng SQL SQL bằng cách xác định các mẫu và cấu trúc mã có thể cho phép kẻ tấn công tiêm mã SQL độc hại vào ứng dụng.

Некоторые инструменты, которые используют анализ статического кода для обнаружения уязвимостей инъекции SQL, включают:

Một số công cụ sử dụng phân tích mã tĩnh để phát hiện lỗ hổng SQL SQL bao gồm:

Findbugs - анализатор статического кода с открытым исходным кодом, который может идентифицировать потенциальную уязвимость в инъекциях SQL, анализируя исходный код Java.

Findbugs là một công cụ phân tích mã tĩnh mã nguồn mở có thể xác định các lỗ hổng SQL tiềm ẩn bằng cách phân tích mã nguồn Java.

CheckMarx - коммерческий инструмент, который использует анализ статического кода для обнаружения уязвимостей, включая инъекцию SQL, на различных языках программирования.

CheckMarx là một công cụ thương mại sử dụng phân tích mã tĩnh để phát hiện các lỗ hổng, bao gồm cả việc chèn SQL, bằng nhiều ngôn ngữ lập trình khác nhau.

Sonarqube - платформа с открытым исходным кодом для непрерывного качества кода, которая включает в себя статический анализатор кода для обнаружения уязвимостей инъекций SQL.

Sonarqube là một nền tảng nguồn mở đảm bảo chất lượng mã liên tục, bao gồm một bộ phân tích mã tĩnh để phát hiện các lỗ hổng chèn SQL.

Анализ статического кода может быть полезен при обнаружении уязвимостей инъекций SQL, но важно отметить, что у него есть ограничения. Анализ статического кода может обнаружить только уязвимости, которые могут быть идентифицированы с помощью анализа кода, и он не сможет обнаружить все возможные точки впрыска. Поэтому важно использовать комбинацию различных методов, включая анализ статического кода, чтобы обеспечить безопасное приложение.

Phân tích mã tĩnh có thể hữu ích trong việc xác định các lỗ hổng chèn SQL, nhưng điều quan trọng cần lưu ý là nó có những hạn chế. Phân tích mã tĩnh chỉ có thể phát hiện các lỗ hổng có thể được xác định thông qua phân tích mã và sẽ không thể phát hiện tất cả các điểm tiêm có thể có. Do đó, điều quan trọng là sử dụng kết hợp các kỹ thuật khác nhau, bao gồm phân tích mã tĩnh, để đảm bảo ứng dụng an toàn.

# Приведите хотя бы один пример как ' или '1'='1 могут быть использованы для тестирование на возможность внедрения SQL инъекции

# Приведите хотя бы один пример как UNION может быть использован для внедрения SQL инъекции

# Требования к базам данных (адекватность предметной области, удобство использования, производительность, защищенность данных)

* Адекватность предметной области выражается в том, что база данных должна позволять выполнять все необходимые операции, которые объективно нужны в реальной жизни в контексте той работы, для которой предназначена база данных.
* Удобство использования (в контексте проектирования баз данных) не связано с удобством для конечного пользователя (который может даже не знать, что в мире существуют какие-то базы данных). Этот термин относится к использованию базы данных в следующих ситуациях:
* при взаимодействии с приложениями (в основном, здесь упор делается на простоту и лёгкость написания безошибочных запросов);
* в процессе её дальнейшего развития (и тогда говорят о таких показателях качества, как поддерживаемость и сопровождаемость).

Таким образом, речь идёт об «удобстве для программиста». Почему это важно? Чем лучше база данных отвечает данному требованию, тем проще и быстрее программисты могут организовать взаимодействие с ней, допуская минимум ошибок и с меньшими затратами решая вопросы быстродействия, безопасности и т.д.

* Производительность - база данных должна обеспечивать быстрое выполнение операций, особенно для больших объемов данных. Это может быть достигнуто оптимизацией структуры базы данных, правильным использованием индексов и запросами, и использованием эффективного оборудования.
* Защищенность данных - база данных должна быть защищена от несанкционированного доступа, обеспечить конфиденциальность, целостность и доступность данных. Для этого могут быть использованы различные методы, такие как аутентификация и авторизация, шифрование данных и резервное копирование. Защищённость данных стоит воспринимать не только в контексте безопасности (ограничения доступа), но и в том смысле, что с данными не должно происходить никаких случайных непредвиденных изменений. Для достижения этой цели порой приходится продумывать огромное количество ограничений, реализуемых через специфические объекты базы данных и даже через отдельный интерфейс в виде набора хранимых процедур

**Как узнать количество столбцов таблицы с помощью UNION?**

В PostgreSQL, оператор `UNION` используется для объединения результатов двух или более запросов.

написать два запроса: один для получения структуры таблицы (например, с помощью `information\_schema.columns`), а другой для получения данных, и объединить их через `UNION`.

SELECT column\_name

FROM information\_schema.columns

WHERE table\_name = 'имя\_таблицы'

UNION

SELECT \*

FROM имя\_таблицы;

…………………………………………………..

SELECT COUNT(\*)

FROM (

SELECT column\_name

FROM information\_schema.columns

WHERE table\_name = 'имя\_таблицы'

) AS subquery;

узнать количество столбцов в таблице PostgreSQL — это использовать системное представление `information\_schema.columns`.

**Без обращения к information\_schema**

Если … узнать количество столбцов таблицы без обращения к `information\_schema`, можно использовать PL/pgSQL для динамического построения запроса и выполнения его с использованием `UNION`.

1. Создать функцию, которая динамически определяет количество столбцов в таблице.

2. Использовать динамическое SQL для выполнения запроса и подсчета количества столбцов.

CREATE OR REPLACE FUNCTION get\_column\_count(table\_name text)

RETURNS integer AS $$

DECLARE

col\_count integer;

sql\_query text;

BEGIN

-- Формируем динамический запрос для подсчета количества столбцов

sql\_query := 'SELECT COUNT(\*) FROM (SELECT \* FROM ' || table\_name || ' LIMIT 0) AS subquery;';

-- Выполняем динамический SQL и сохраняем результат в переменную col\_count

EXECUTE sql\_query INTO col\_count;

-- Возвращаем количество столбцов

RETURN col\_count;

END;

$$ LANGUAGE plpgsql;

после создания этой функции вы можете вызвать её, чтобы узнать количество столбцов в таблице.

SELECT get\_column\_count('имя\_таблицы');

**Именно с помощью инъекции.**

Для определения количества столбцов в таблице с помощью SQL-инъекции, можно использовать технику поэтапного угадывания. Основная идея заключается в последовательном увеличении количества столбцов в запросе до тех пор, пока запрос не выполнится успешно.

1. Начальная попытка:

Предположим, что у вас есть базовый уязвимый запрос, например:

SELECT \* FROM таблица WHERE условие = 'значение';

2. Добавление UNION SELECT:

Вы добавляете `UNION SELECT`, начиная с одного столбца:

' UNION SELECT 1; --

Если запрос не удался, увеличиваете количество столбцов.

3. Пошаговое увеличение количества столбцов\*\*:

Продолжаете добавлять столбцы до тех пор, пока запрос не будет успешным:

' UNION SELECT 1, 2; --

' UNION SELECT 1, 2, 3; --

...

' UNION SELECT 1, 2, 3, 4, 5; --

```

4. \*\*Успешное выполнение\*\*:

Когда запрос выполнится успешно, количество столбцов будет соответствовать числу добавленных столбцов.

1. **Синтаксис по разграничению доступа к объектам с помощью команды grant. К каким объектам БД может быть ограничен доступ? Виды привилегий**

**Cú pháp hạn chế quyền truy cập vào các đối tượng bằng lệnh cấp. Những đối tượng cơ sở dữ liệu nào có thể bị hạn chế quyền truy cập? Các loại đặc quyền**

Команда GRANT используется в системе управления базами данных (СУБД) для назначения определенных привилегий пользователям или пользователям-группам. Синтаксис команды GRANT может быть следующим:

**Lệnh GRAN**T được sử dụng trong hệ thống quản lý cơ sở dữ liệu (DBMS) để gán các đặc quyền cụ thể cho người dùng hoặc nhóm người dùng. Cú pháp lệnh GRANT có thể như sau:

*GRANT* ***permission\_list*** *ON* ***object\_name*** *TO* ***role\_specification*** *[WITH GRANT OPTION];*

*где:*

* *‘permission\_list’ - список разрешений, которые необходимо предоставить (например, SELECT, INSERT, UPDATE, DELETE);*
* *‘object\_name’ - имя объекта базы данных (например, таблица, представление, процедура);*
* *‘role\_specification’ can be:*

*Text

Description automatically generated*

* *‘WITH GRANT OPTION’ - опция, которая позволяет пользователю, получившему разрешение на доступ к объекту, предоставлять доступ к этому объекту другим пользователям.*

*Một tùy chọn cho phép người dùng có quyền truy cập vào một đối tượng có thể cấp quyền truy cập vào đối tượng đó cho những người dùng khác.*

Пример команды, предоставляющей разрешение на чтение таблицы "users" пользователю "user1":

*Một lệnh ví dụ cấp quyền đọc bảng "users" cho người dùng "user1":*

*GRANT SELECT ON users TO user1;*

В данном примере группе пользователей Role1 назначено право на выполнения хранимой процедуры spTest:

*GRANT EXECUTE ON dbo.spTest TO Role1;*

В данном примере всем пользователям базы данных назначено право на выполнение операции UPDATE для таблицы Product:

*GRANT UPDATE ON Product TO PUBLIC;*

В реляционных базах данных доступ может быть ограничен к следующим объектам:

Trong cơ sở dữ liệu quan hệ, quyền truy cập có thể bị giới hạn ở các đối tượng sau:

* Таблицы: доступ к чтению, записи, удалению или изменению данных в таблицах может быть ограничен.
* Bảng: Quyền truy cập vào việc đọc, viết, xóa hoặc thay đổi dữ liệu trong bảng có thể bị hạn chế.
* Представления: доступ к чтению или изменению представлений может быть ограничен.
* Lượt xem: Quyền truy cập vào chế độ xem đọc hoặc chỉnh sửa có thể bị hạn chế.
* Хранимые процедуры: доступ к выполнению хранимых процедур может быть ограничен.
* Thủ tục lưu trữ: Quyền truy cập để thực hiện các thủ tục được lưu trữ có thể bị hạn chế.
* Индексы: доступ к индексам может быть ограничен.
* Chỉ mục: Việc truy cập vào chỉ mục có thể bị hạn chế.
* Секреты: доступ к конфиденциальным данным о пользователях или подсистемах может быть ограничен.
* Bí mật: Quyền truy cập vào dữ liệu nhạy cảm về người dùng hoặc hệ thống con có thể bị hạn chế.
* Роли: управление правами доступа в зависимости от роли пользователя.
* Vai trò: Quản lý quyền truy cập dựa trên vai trò của người dùng.

Привилегии включают следующие виды:

Đặc quyền bao gồm các loại sau:

* SELECT: разрешает выборку данных из объектов БД.
* SELECT: cho phép lựa chọn dữ liệu từ các đối tượng cơ sở dữ liệu.
* INSERT: разрешает добавление новых записей в таблицу.
* INSERT: Cho phép thêm bản ghi mới vào bảng.
* UPDATE: разрешает обновление записей в таблице.
* UPDATE: Cho phép cập nhật các bản ghi trong bảng.
* DELETE: разрешает удаление записей из таблицы.
* DELETE: Cho phép xóa bản ghi khỏi bảng.
* EXECUTE: разрешает выполнение хранимых процедур.
* EXECUTE: Cho phép thực hiện các thủ tục được lưu trữ.
* REFERENCES: разрешает ссылаться на столбцы таблицы.