Определение представления

Представления, или *просмотры* (`VIEW`), представляют собой временные, производные (иначе - виртуальные) таблицы и являются объектами базы данных, информация в которых не хранится постоянно, как в базовых таблицах, а формируется динамически при обращении к ним. Обычные таблицы относятся к базовым, т.е. содержащим данные и постоянно находящимся на устройстве хранения информации. *Представление* не может существовать само по себе, а определяется только в терминах одной или нескольких таблиц. Применение *представлений* позволяет разработчику базы данных обеспечить каждому пользователю или группе пользователей наиболее подходящие способы работы с данными, что решает проблему простоты их использования и безопасности. Содержимое *представлений* выбирается из других таблиц с помощью выполнения запроса, причем при изменении значений в таблицах данные в *представлении* автоматически меняются. *Представление* - это фактически тот же запрос, который выполняется всякий раз при участии в какой-либо команде. Результат выполнения этого запроса в каждый момент времени становится содержанием *представления*. У пользователя создается впечатление, что он работает с настоящей, реально существующей таблицей.

У СУБД есть две возможности *реализации представлений*. Если его определение простое, то система формирует каждую запись *представления* по мере необходимости, постепенно считывая исходные данные из базовых таблиц. В случае сложного определения СУБД приходится сначала выполнить такую операцию, как материализация *представления*, т.е. сохранить информацию, из которой состоит *представление*, во временной таблице. Затем система приступает к выполнению пользовательской команды и формированию ее результатов, после чего временная таблица удаляется.

Представление - это предопределенный запрос, хранящийся в базе данных, который выглядит подобно обычной таблице и не требует для своего хранения дисковой памяти. Для хранения *представления* используется только оперативная память. В отличие от других объектов базы данных *представление* не занимает дисковой памяти за исключением памяти, необходимой для хранения определения самого *представления*.

Создания и изменения *представлений* в стандарте языка и реализации в MS SQL Server совпадают и представлены следующей командой:

```
<определение_представления> ::=
    { CREATE| ALTER} VIEW имя_представления
    [(имя_столбца [,...n])]
    [WITH ENCRYPTION]
    AS SELECT_оператор
    [WITH CHECK OPTION]
```

Рассмотрим назначение основных параметров.

По умолчанию имена столбцов в *представлении* соответствуют именам столбцов в исходных таблицах. Явное указание имени столбца требуется для вычисляемых столбцов или при объединении нескольких таблиц, имеющих столбцы с одинаковыми именами. Имена столбцов перечисляются через запятую, в соответствии с порядком их следования в *представлении*.

Параметр `WITH ENCRYPTION` предписывает серверу шифровать SQL-код запроса, что гарантирует невозможность его несанкционированного просмотра и использования. Если при определении *представления* необходимо скрыть имена исходных таблиц и столбцов, а также алгоритм объединения данных, необходимо применить этот аргумент.

Параметр `WITH CHECK OPTION` предписывает серверу исполнять проверку изменений, производимых через *представление*, на соответствие критериям, определенным в операторе `SELECT` . Это означает, что не допускается выполнение изменений, которые приведут к исчезновению строки из *представления*. Такое случается, если для *представления* установлен горизонтальный фильтр и изменение данных приводит к несоответствию строки установленным фильтрам. Использование аргумента `WITH CHECK OPTION` гарантирует, что сделанные изменения будут отображены в *представлении*. Если пользователь пытается выполнить изменения, приводящие к исключению строки из *представления*, при заданном аргументе `WITH CHECK OPTION` сервер выдаст сообщение об ошибке и все изменения будут отклонены.

Пример 10.1. Показать в представлении клиентов из Москвы.

Создание представления:

```
CREATE  VIEW view1 AS
SELECT КодКлиента, Фамилия, ГородКлиента
FROM Клиент
WHERE ГородКлиента='Москва'
```

Пример 10.1. Представление клиентов из Москвы.

Выборка данных из представления:

```
SELECT * FROM view1
```

Обращение к *представлению* осуществляется с помощью оператора `SELECT` как к обычной таблице.

Представление можно использовать в команде так же, как и любую другую таблицу. К *представлению* можно строить запрос, модифицировать его (если оно отвечает определенным требованиям), соединять с другими таблицами. Содержание *представления* не фиксировано и обновляется каждый раз, когда на него ссылаются в команде. *Представления* значительно расширяют возможности управления данными. В частности, это прекрасный способ разрешить доступ к информации в таблице, скрыв часть данных.

Так, в примере 10.1 *представление* просто ограничивает доступ пользователя к данным таблицы `Клиент`, позволяя видеть только часть значений.

Выполним команду:

```
INSERT INTO view1 VALUES (12,'Петров', 'Самара')
```

Это допустимая команда в *представлении*, и строка будет добавлена с помощью *представления* `view1` в таблицу `Клиент`. Однако, когда информация будет добавлена, строка исчезнет из *представления*, поскольку название города отлично от Москвы. Иногда такой подход может стать проблемой, т.к. данные уже находятся в таблице, но пользователь их не видит и не в состоянии выполнить их удаление или модификацию. Для исключения подобных моментов служит `WITH CHECK OPTION` в определении *представления*. Фраза размещается в определении *представления*, и все команды модификации будут подвергаться проверке.

```
ALTER VIEW view1 AS
SELECT КодКлиента, Фамилия, ГородКлиента
FROM Клиент
WHERE ГородКлиента='Москва'
WITH CHECK OPTION
```

Пример 10.2. Создание представления с проверкой команд модификации.

Для такого *представления* вышеупомянутая вставка значений будет отклонена системой.

Таким образом, *представление* может изменяться командами модификации DML, но фактически модификация воздействует не на само *представление*, а на базовую таблицу.

Представление удаляется командой:

```
DROP VIEW имя_представления [,...n]
```

Обновление данных в представлениях

Не все *представления* в SQL могут быть модифицированы. *Модифицируемое представление* определяется следующими критериями:

- основывается только на одной базовой таблице;
- содержит первичный ключ этой таблицы;
- не содержит `DISTINCT` в своем определении;
- не использует `GROUP BY` или `HAVING` в своем определении;
- по возможности не применяет в своем определении подзапросы;
- не использует константы или выражения значений среди выбранных полей вывода;
- в *просмотр* должен быть включен каждый столбец таблицы, имеющий атрибут `NOT NULL` ;
- оператор `SELECT` *просмотра* не использует агрегирующие (итоговые) функции, соединения таблиц, хранимые процедуры и функции, определенные пользователем;
- основывается на одиночном запросе, поэтому объединение `UNION` не разрешено.

Если *просмотр* удовлетворяет этим условиям, к нему могут применяться операторы `INSERT` , `UPDATE` , `DELETE` . Различия между *модифицируемыми представлениями* и *представлениями*, предназначенными только для чтения, не случайны. Цели, для которых их используют, различны. С *модифицируемыми представлениями* в основном обходятся точно так же, как и с базовыми таблицами. Фактически, пользователи не могут даже осознать, является ли объект, который они запрашивают, базовой таблицей или *представлением*, т.е. прежде всего это средство защиты для сокрытия конфиденциальных или не относящихся к потребностям данного пользователя частей таблицы. *Представления* в режиме <только для чтения> позволяют получать и форматировать данные более рационально. Они создают целый арсенал сложных запросов, которые можно выполнить и повторить снова, сохраняя полученную информацию. Результаты этих запросов могут затем использоваться в других запросах, что позволит избежать сложных предикатов и снизить вероятность ошибочных действий.

```
CREATE VIEW view2 AS
SELECT Клиент.Фамилия, Клиент.Фирма,
       Сделка.Количество
FROM Клиент INNER JOIN Сделка
ON Клиент.КодКлиента=Сделка.КодКлиента

Пример 10.3. Немодифицируемое представление с данными из разных таблиц.

CREATE VIEW view3(Тип, Общ_остаток) AS
SELECT Тип, Sum(Остаток)
FROM Товар
GROUP BY Тип

Пример 10.4. Немодифицируемое представление с группировкой и итоговыми функциями.
```

Обычно в *представлениях* используются имена, полученные непосредственно из имен полей основной таблицы. Однако иногда необходимо дать столбцам новые имена, например, в случае итоговых функций или вычисляемых столбцов.

```
CREATE VIEW view4(Код, Название,
    Тип, Цена, Налог) AS
SELECT КодТовара, Название,
        Тип, Цена, Цена*0.05 FROM Товар

Пример 10.5. Модифицируемое представление с вычислениями.
```

Преимущества и недостатки представлений

Механизм *представления* - мощное средство СУБД, позволяющее скрыть реальную структуру БД от некоторых пользователей за счет определения *представлений*. Любая *реализация представления* должна гарантировать, что состояние представляемого отношения точно соответствует состоянию данных, на которых определено это *представление*. Обычно вычисление *представления* производится каждый раз при его использовании. Когда *представление* создается, информация о нем записывается в каталог БД под собственным именем. Любые изменения в данных адекватно отобразятся в *представлении* - в этом его отличие от очень похожего на него запроса к БД. В то же время запрос представляет собой как бы <мгновенную фотографию> данных и при изменении последних запрос к БД необходимо повторить. Наличие *представлений* в БД необходимо для обеспечения логической независимости данных. Если система обеспечивает физическую независимость данных, то изменения в физической структуре БД не влияют на работу пользовательских программ. Логическая независимость подразумевает тот факт, что при изменении логической структуры данных влияние на пользовательские программы также не оказывается, а значит, система должна уметь решать проблемы, связанные с ростом и реструктуризацией БД. Очевидно, что с увеличением количества данных, хранимых в БД, возникает необходимость ее расширения за счет добавления новых атрибутов или отношений - это называется ростом БД. Реструктуризация данных подразумевает сохранение той же самой информации, но изменяется ее расположение, например, за счет перегруппировки атрибутов в отношениях. Предположим, некоторое отношение в силу каких-либо причин необходимо разделить на два. Соединение полученных отношений в *представлении* воссоздает исходное отношение, а у пользователя складывается впечатление, что никакой реструктуризации не производилось. Помимо решения проблемы реструктуризации *представление* можно применять для просмотра одних и тех же данных разными пользователями и в различных вариантах. С помощью *представлений* пользователь имеет возможность ограничить объем данных для удобства работы. Наконец, механизм *представлений* позволяет скрыть служебные данные, не интересные пользователям.

В случае выполнения СУБД на отдельно стоящем персональном компьютере использование *представлений* обычно имеет целью лишь упрощение структуры запросов к базе данных. Однако в случае многопользовательской сетевой СУБД *представления* играют ключевую роль в определении структуры базы данных и организации защиты информации. Рассмотрим основные *преимущества применения представлений* в подобной среде.

Независимость от данных

С помощью *представлений* можно создать согласованную, неизменную картину структуры базы данных, которая будет оставаться стабильной даже в случае изменения формата исходных таблиц (например, добавления или удаления столбцов, изменения связей, разделения таблиц, их реструктуризации или переименования). Если в таблицу добавляются или из нее удаляются не используемые в *представлении* столбцы, то изменять определение этого *представления* не потребуется. Если структура исходной таблицы переупорядочивается или таблица разделяется, можно создать *представление*, позволяющее работать с виртуальной таблицей прежнего формата. В случае разделения исходной таблицы, прежний формат может быть виртуально воссоздан с помощью *представления*, построенного на основе соединения вновь созданных таблиц - конечно, если это окажется возможным. Последнее условие можно обеспечить с помощью помещения во все вновь созданные таблицы первичного ключа прежней таблицы.

Актуальность

Изменения данных в любой из таблиц базы данных, указанных в определяющем запросе, немедленно отображается на содержимом *представления*.

Повышение защищенности данных

Права доступа к данным могут быть предоставлены исключительно через ограниченный набор *представлений*, содержащих только то подмножество данных, которое необходимо пользователю. Подобный подход позволяет существенно ужесточить контроль за доступом отдельных категорий пользователей к информации в базе данных.

Снижение стоимости

Представления позволяют упростить структуру запросов за счет объединения данных из нескольких таблиц в единственную виртуальную таблицу. В результате многотабличные запросы сводятся к простым запросам к одному *представлению*.

Дополнительные удобства

Создание *представлений* может обеспечивать пользователей дополнительными удобствами - например, возможностью работы только с действительно нужной частью данных. В результате можно добиться максимального упрощения той модели данных, которая понадобится каждому конечному пользователю.

Возможность настройки

Представления являются удобным средством настройки индивидуального образа базы данных. В результате одни и те же таблицы могут быть предъявлены пользователям в совершенно разном виде.

Обеспечение целостности данных

Если в операторе `CREATE VIEW` будет указана фраза `WITH CHECK OPTION` , то СУБД станет осуществлять контроль за тем, чтобы в исходные таблицы базы данных не была введена ни одна из строк, не удовлетворяющих предложению `WHERE` в определяющем запросе. Этот механизм гарантирует целостность данных в *представлении*.

Практика ограничения доступа некоторых пользователей к данным посредством создания специализированных *представлений*, безусловно, имеет значительные преимущества перед предоставлением им прямого доступа к таблицам базы данных.

Однако использование *представлений* в среде SQL не лишено *недостатков*.

Ограниченные возможности обновления

В некоторых случаях *представления* не позволяют вносить изменения в содержащиеся в них данные.

Структурные ограничения

Структура *представления* устанавливается в момент его создания. Если определяющий запрос представлен в форме `SELECT * FROM_` , то символ `*` ссылается на все столбцы, существующие в исходной таблице на момент создания *представления*. Если впоследствии в исходную таблицу базы данных добавятся новые столбцы, то они не появятся в данном *представлении* до тех пор, пока это *представление* не будет удалено и вновь создано.

Снижение производительности

Использование *представлений* связано с определенным снижением производительности. В одних случаях влияние этого фактора совершенно незначительно, тогда как в других оно может послужить источником существенных проблем. Например, *представление*, определенное с помощью сложного многотабличного запроса, может потребовать значительных затрат времени на обработку, поскольку при его разрешении потребуется выполнять соединение таблиц всякий раз, когда понадобится доступ к данному *представлению*. Выполнение разрешения *представлений* связано с использованием дополнительных вычислительных ресурсов.