Ощущение полной безопасности наиболее опасно.
Илья Нисонович Шевелев
Везде, где есть жизнь, есть и опасность.
Ральф Уолдо Эмерсон

Безопасность систем баз данных.

Тема лекции: Авторизация и отслеживание изменений

Работа с разрешениями с помощью инструкций языка Transact-SQL. Работа с разрешениями с помощью среды MS SQL Server Management Studio. Управление авторизацией и аутентификацией для автономных баз данных. Отслеживание изменений в таблицах базы данных. Защита данных с использованием представлений (VIEW). Преимущества и недостатки представлений.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ЛЕКЦИОННЫЕ МАТЕРИАЛЫ (ПРЕЗЕНТАЦИИ К ЛЕКЦИОННЫМ МАТЕРИАЛАМ)

Безопасность систем баз данных

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень	специалист *(бакалавриат, магистратура, специалитет)*
Форма обучения	очная *(очная, очно-заочная, заочная)*
Направление(-я) подготовки	10.03.01 «Информационная безопасность автоматизированных систем» *(код и наименование)*
Институт	Кибербезопасности и цифровых технологий *(полное и краткое наименование)*
Кафедра	Информационно-аналитические системы кибербезопасности (КБ-2) *(полное и краткое наименование кафедры, реализующей дисциплину (модуль))*
Лектор	К.т.н., доцент Шукенбаев Айрат Бисенгалеевич *(сокращенно – ученая степень, ученое звание; полностью – ФИО)*

Используются в данной редакции с учебного года 2023/2024
(учебный год цифрами)

Проверено и согласовано «____» _____20___г.

_____ А.А. Бакаев
(подпись директора Института/Филиала с расшифровкой)

Москва 2024 г.

♦ GRANT;
♦ DENY;
♦ REVOKE.

Модель безопасности компонента *Database Engine* - модель разделяет мир сервера базы данных на принципалов безопасности и защищаемые объекты. Каждый защищаемый объект имеет связанные с ним разрешения, которые могут быть предоставлены принципалу.

Принципалы, такие как отдельные лица, группы или приложения, могут обращаться к защищаемым объектам.

Database Engine. *Это сердце MS SQL Server. Технически он реализован в виде системного сервиса, позволяющего добавлять, извлекать и изменять хранящуюся информацию. Именно Database Engine обеспечивает реализацию высокопроизводительных систем с использованием механизмов транзакций (OLTP) и аналитики (OLAP).*

Database Engine, в свою очередь, тоже является мультикомпонентным и состоит из следующих модулей: Storage Engine, Security Subsystem, Programming Interfaces, Service Broker, SQL Server Agent, Replication и целый комплекс технологий, обеспечивающих возможность работы сервера с большим объемом запросов в единицу времени (High Availability).

Защищаемые объекты – это ресурсы, доступ к которым регулируется подсистемой авторизации.

Существует три основных класса защищаемых объектов: сервер, база данных и схема.

Инструкция GRANT

Инструкция GRANT предоставляет разрешения принципалам на защищаемые объекты. Эта инструкция имеет следующий синтаксис:

```
GRANT {ALL [PRIVILEGES]} | permission_list
   [ON [class::] securable] TO principal_list [WITH GRANT OPTION]    [AS principal ]
```

Предложение *ALL* означает, что указанному принципалу предоставляются все применимые к указанному защищаемому объекту разрешения.

В параметре *permission_list* указываются разрешаемые инструкции или объекты, а в параметре *class* - класс или имя защищаемого объекта, для которого предоставляются разрешения.

Предложение *on securable* указывает защищаемый объект, на который предоставляется разрешение.

В параметре *prinicpal_list* перечисляются все учетные записи, которым предоставляются разрешения. Параметр *principal* и составляющие списка *principal_list* могут быть учетной записью пользователя *Windows*, регистрационным именем или учетной записью пользователя, сопоставленной с сертификатом, регистрационным именем, сопоставленным с асимметричным ключом, пользователем базы данных, ролью базы данных или ролью приложения.

Разрешение	Применение	Описание
SELECT	Таблицы и их столбцы, синонимы, представления и их столбцы, возвращающие табличные значения функции	Предоставляет возможность выборки (чтения) строк. Это разрешение можно ограничить одним или несколькими столбцами, перечислив требуемые столбцы. (Если список столбцов отсутствует, то разрешение применимо ко всем столбцам таблицы)
INSERT	Таблицы и их столбцы, синонимы, представления и их столбцы	Предоставляет возможность вставлять столбцы
UPDATE	Таблицы и их столбцы, синонимы, представления и их столбцы	Предоставляет возможность изменять значения столбцов. Это разрешение можно ограничить одним или несколькими столбцами, перечислив требуемые столбцы. (Если список столбцов отсутствует, то разрешение применимо ко всем столбцам таблицы)
DELETE	Таблицы и их столбцы, синонимы, представления и их столбцы	Предоставляет возможность удалять столбцы
REFERENCES	Определяемые пользователем функции (SQL и среды CLR), таблицы и их столбцы, синонимы, представления и их столбцы	Предоставляет возможность обращаться к столбцам внешнего ключа в родительской таблице, когда пользователь не имеет разрешения SELECT для этой таблицы
EXECUTE	Хранимые процедуры (SQL и среды CLR), определяемые пользователем функции (SQL и среды CLR), синонимы	Предоставляет возможность выполнять указанную хранимую процедуру или определенную пользователем функцию
CONTROL	Хранимые процедуры (SQL и среды CLR), определяемые пользователем функции (SQL и среды CLR), синонимы	Предоставляет возможности, подобные возможностям владельца; получатель имеет практически все разрешения, определенные для защищаемого объекта. Принципал, которому было предоставлено разрешение CONTROL, также имеет возможность предоставлять разрешения на данный защищаемый объект. Разрешение CONTROL на определенной области видимости неявно включает разрешение CONTROL для всех защищаемых объектов в этой области видимости (см. пример 12.16)
ALTER	Хранимые процедуры (SQL и среды CLR), определяемые пользователем функции (SQL и среды CLR), таблицы, представления	Предоставляет возможность изменять свойства (за исключением владения) защищаемых объектов. Когда это право предоставляется применимо к области, оно также предоставляет права на выполнение инструкций ALTER, CREATE и DROP на любых защищаемых объектах в данной области
TAKE OWNERSHIP	Хранимые процедуры (SQL и среды CLR), определяемые пользователем функции (SQL и среды CLR), таблицы, представления, синонимы	Предоставляет возможность становиться владельцем защищаемого объекта, для которого оно применяется

Пример 2. В этом примере пользователям *Vasya* и [*ProfessorWeb\Alexandr*] дается право на выполнение инструкций языка *Transact-SQL CREATE TABLE и CREATE PROCEDURE*.

```
USE SampleDb;

GRANT CREATE TABLE, CREATE PROCEDURE
    TO Vasya, [ProfessorWeb\Alexandr];
```

Пример 3. Пользователю Vasya предоставляется возможность для создания определяемых пользователем функций в БД SampleDb:

```
USE SampleDb;

GRANT CREATE FUNCTION TO Vasya;
```

Пример 4. Показано использование разрешения *SELECT* в инструкции *GRANT*:

```
USE SampleDb;

GRANT SELECT ON Employee
    TO Vasya;
```

Пример 5. Показано использование разрешения *UPDATE* в инструкции *GRANT*:

```
USE SampleDb;

GRANT UPDATE ON Works_on (EmpId, EnterDate)
   TO Vasya;
```

Пример 6. Предоставление доступа для чтения метаданных.

```
USE SampleDb;

GRANT VIEW DEFINITION ON OBJECT::Employee TO Vasya;
GRANT VIEW DEFINITION ON SCHEMA::dbo TO Vasya;
```

Пример 7. Показано использование разрешения *CONTROL*

```
USE SampleDb;

GRANT CONTROL ON DATABASE::SampleDb TO Vasya;
```

Пример 8. Инструкция GRANT с предложением WITH GRANT OPTION

```
USE SampleDb;

GRANT SELECT ON Works_on TO Mary
    WITH GRANT OPTION;
```

Инструкция DENY - запрещает пользователю выполнять указанные действия на указанных объектах. Эта инструкция имеет следующий синтаксис:

```
DENY {ALL [PRIVILEGES] } | permission_list
    [ON [class::] securable] TO principal_list
    [CASCADE] [ AS principal ]
```

Пример 9. Запрещение пользователю peter двух предоставленных ранее разрешений

```
USE SampleDb;

DENY CREATE TABLE, CREATE PROCEDURE
    TO Vasya;
```

Пример 10. Запрещение разрешений отдельным пользователям

```
USE SampleDb;

GRANT SELECT ON Project TO PUBLIC;
DENY SELECT ON Project TO Vasya, Mary;
```

Инструкция REVOKE - удаляет предоставленное или запрещенное ранее разрешение. Эта инструкция имеет следующий синтаксис:

```
REVOKE [GRANT OPTION FOR]
    { [ALL [PRIVILEGES] ] | permission_list ]}
    [ON [class:: ] securable ]
    FROM principal_list [CASCADE] [ AS principal ]
```

Параметр *GRANT OPTION FOR* используется для отмены эффекта предложения *WITH GRANT OPTION* в соответствующей инструкции *GRANT*.

Инструкция *REVOKE* отменяет как "позитивные" разрешения, предоставленные инструкцией *GRANT*, так и "негативные" разрешения, предоставленные инструкцией *DENY*.

Пример 11. Использование инструкции REVOKE:

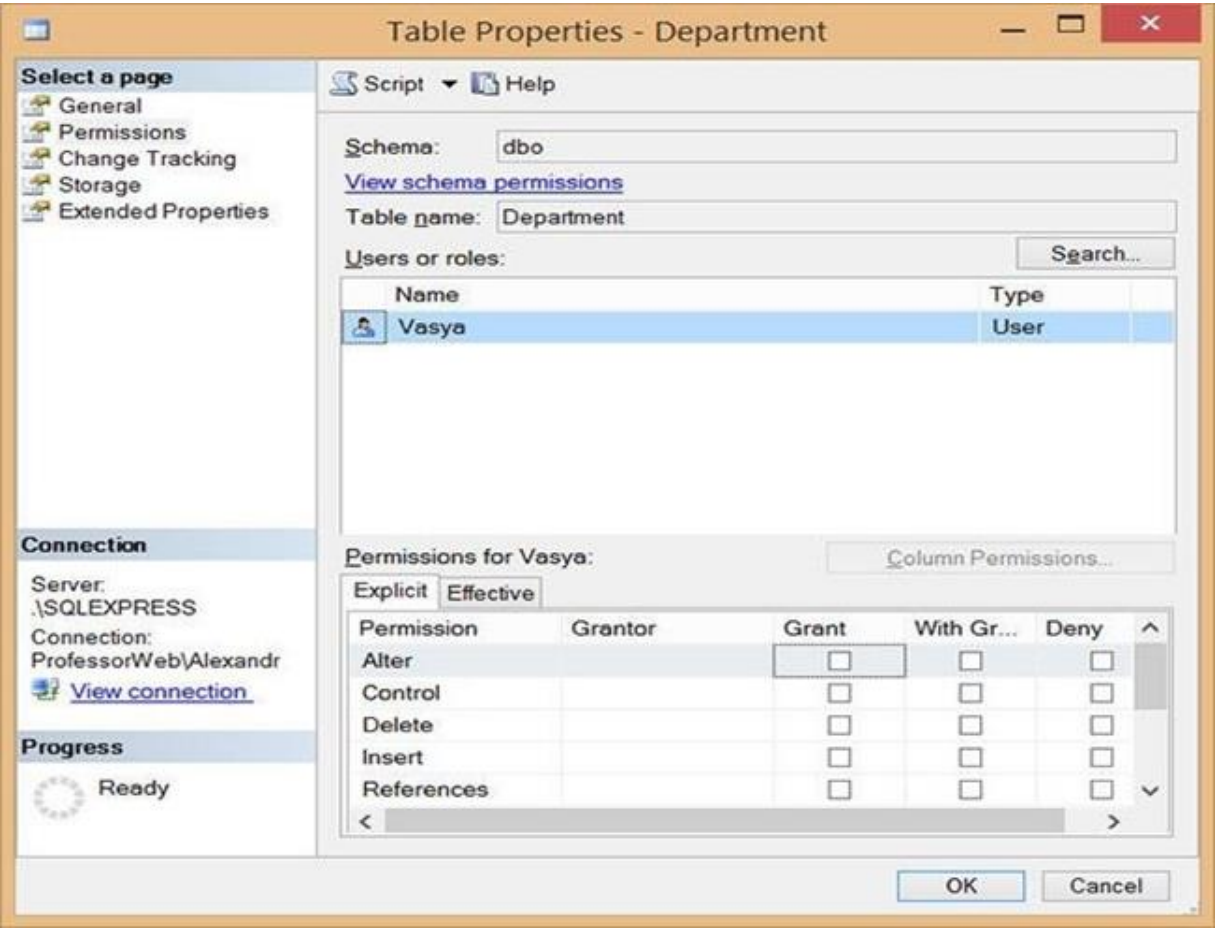
```
USE SampleDb;

REVOKE SELECT ON Project FROM public;
```

Работа с разрешениями с помощью среды MS SQL Server Management Studio

Разрешения на определенные действия установлены в значение *G* (от *GRANT*) в столбце *state* в представлении просмотра каталога *sys.database_permissions*. Негативная запись в таблице не дает возможность пользователям выполнять деятельность. Значение *D* (от *DENY*) в этом столбце *state* аннулирует разрешение, предоставленное пользователю явно или неявно посредством предоставления его роли, к котор ой он принадлежит. Таким образом, пользователь не может выполнять данное действие в любом случае. И последнее возможное значение *R* (от *REVOKE*) в столбце *state* означает, что пользователь не имеет никаких явных разрешений, но может выполнять действие, если роли, к которой он принадлежит, предоставлено соответствующее разрешение.

Рис. 1. Управление разрешениями для БД с помощью среды Management Studio



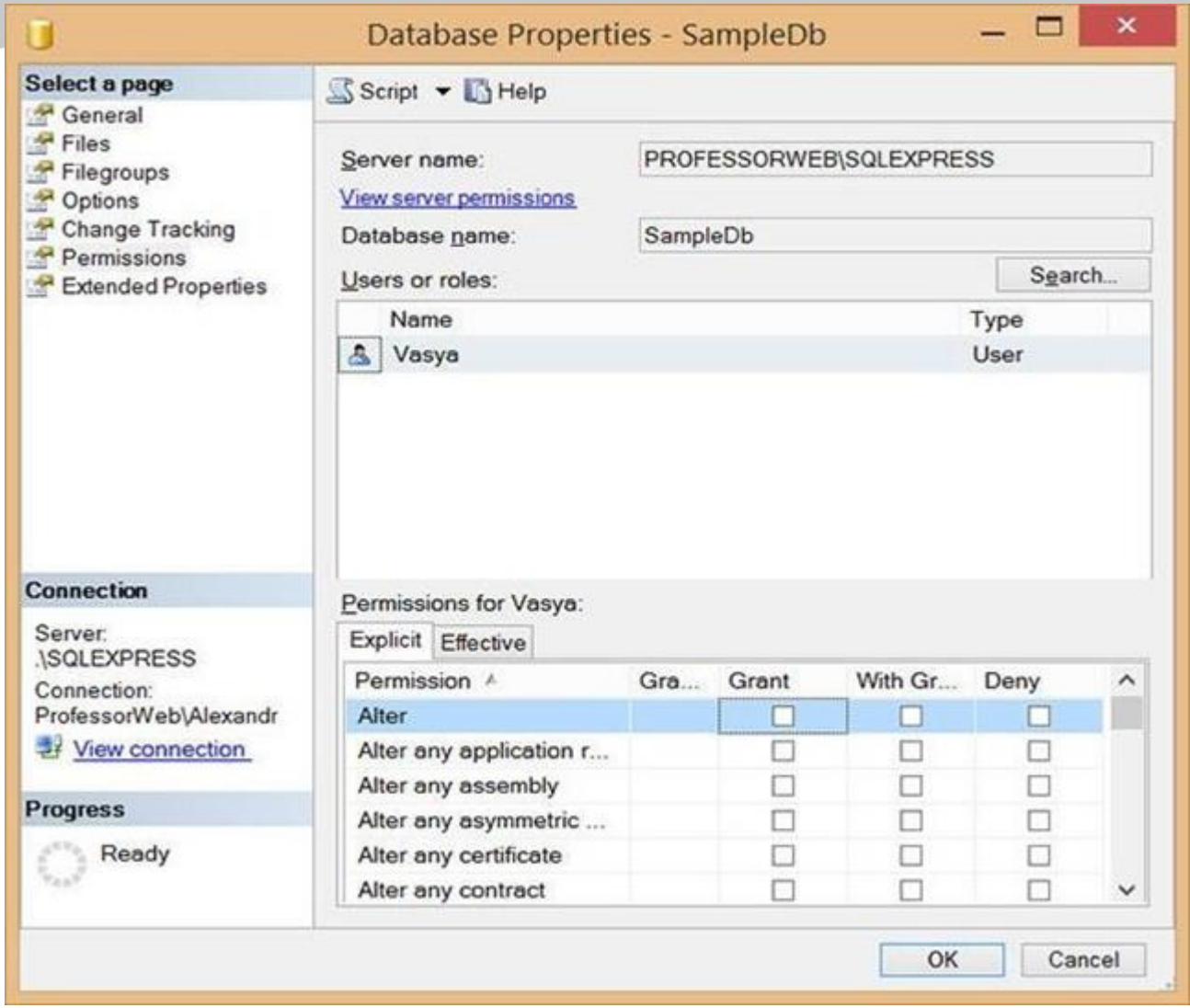


Рис. 2. Предоставление и запрещение разрешений для отдельной таблицы базы данных посредством среды Management Studio

Управление авторизацией и аутентификацией для автономных баз данных

Пример 12. Создание пользователя для автономной базы данных.

```
USE my_sample;
CREATE USER my_login
WITH PASSWORD = 'x1y2z3w4?';
```

Попытка создать автономного пользователя в обычной базе данных завершится ошибкой с выводом следующего сообщения об ошибке:

```
Msg 33233, Level 16, State 1, Line 1
You can only create a user with a password in a contained database.
(Сообщение 33233, уровень 16, состояние 1, строка 2
```

Пользователь, сопоставленный с регистрационным именем входа в *SQL Server*, преобразовывается в пользователя с паролем автономной базы данных посредством хранимой системной процедуры ***sp_migrate_user_to_contained***.

Пример 13. Применение системной процедуры ***sp_migrate_user_to_contained***

```
USE my_sample;
EXEC sp_migrate_user_to_contained
@username = 'mary_a',
@rename = N'keep_name',
@disablelogin = N'do_not_disable_login';
```

Чтобы узнать, какие части базы данных можно переместить в другой эк-земпляр сервера, можно воспользоваться динамическим административным представлением *sys.dm_db_uncontained_entities*.

Отслеживание изменений в таблицах базы данных

Понятие *отслеживание изменений* означает документирование всех операций вставок, обновлений и удалений, применяемых к таблицам базы данных. С помощью этих записей можно будет в дальнейшем определить, кто и когда обращался к данным. Отслеживание изменений можно реализовать двумя способами:

- ♦ используя триггеры;
- ♦ используя систему отслеживания измененных данных.

С помощью триггеров можно создать журнал аудита действий в таблицах базы данных.

Основной целью системы отслеживания измененных данных является создание журнала аудита, в котором фиксируется, кто и когда изменяет какие данные, но ее также можно использовать для поддержки параллельных обновлений (*concurrency update*).

Прежде чем для таблиц можно создавать экземпляры отслеживания, для базы данных, содержащей эти таблицы, требуется разрешить возможность отслеживания измененных данных. Это осуществляется с помощью системной хранимой процедуры *sys.sp_cdc_enable_db*, как это показано в примере 14. (Исполнять эту процедуру могут только члены фиксированной роли сервера *sysadmin*.)

Пример 14. Разрешение возможности отслеживания измененных данных

```
USE sample;
EXECUTE sys.sp_cdc_enable_db
```

Определить, разрешена ли для базы данных возможность отслеживания измененных данных, можно, исследовав значение столбца *is_cdc_enabled* представления каталога *sys.databases*. Если это значение равно 1, возможность отслеживания измененных данных для данной базы данных разрешена.

Возможность отслеживания низменных данных для таблицы разрешается посредством системной хранимой процедуры *sys.sp_cdc_enable_table*. Применение этой системной процедуры показано в примере 15.

Пример 15. Разрешение возможности отслеживания измененных данных для таблицы.

```
USE sample;
EXECUTE sys.sp_cdc_enable_table
@source_schema = N'dbo',
@source_name = N'works_on',
@role_name = N'cdc_admin';
```

Для демонстрации работы системы отслеживания измененных данных изменим содержание исходной таблицы *works_on* базы данных *sample* с помощью примера 16.

Пример 16. Изменение содержимого исходной таблицы

```
 USE sample;
INSERT INTO works_on VALUES (10102,
'p2', 'Ana-lyst', NULL);
INSERT INTO works_on VALUES (9031,
'p2', 'Ana-lyst', NULL);
INSERT INTO works_on VALUES (29346,
'p3', 'Clerk', NULL);
```

По умолчанию, для доступа к данным в связанной таблице изменений создается, по крайней мере, одна возвращающая табличное значение функция. Эту функцию можно использовать для запроса всех изменений, осуществляемых на протяжении заданного интервала времени. Имя функции составляется, конкатенируя префикс *cdc.fh_cdc_get_all_changes__* со значением параметра *@capture_instance*. Для данного примера значение этого параметра равно *dbo_works_on*, а запрос показан в примере 17.

Пример 17. Запрос для отображения измененных данных

```
USE sample;
SELECT *
FROM cdc.fn_cdc_get_all_changes_dbo_works_on
(sys .fn_cdc_get_min_lsn('dbo_works_on'),  sys .fn_cdc_get_max_lsn(), ' all ');
```

Часть результата, выдаваемого после запроса, приведенного в примере 17:

__$start_lsn	__$update mask	emp_no	project_no	job	enter_date
0x0000001C000001EF0003	0x0F	10102	p2	Analyst	NULL
0x00000010000000100003	0x0F	9031	p2	Analyst	NULL
0x00000010000000110003	0x0F	29346	p3	Clerk	NULL

Пример 18. Запрос для отслеживания изменений в течение периода времени

```
USE sample;
DECLARE @from_lsn binary(10), @to_lsn bina-ry(10); SELECT @from_lsn =
sys.fn_cdc_map_time_to_lsn('smallest greater than', GETDATE() - 1); SELECT @to_lsn =
sys.fn_cdc_map_time_to_lsn('largest less than or equal', GETDATE()); SELECT * FROM
cdc.fn_cdc_get_all_changes_dbo_works_on(@from_lsn,
@to_lsn, 'all');
```

Защита данных с использованием представлений (*VIEW*)

Назначение представлений

Представление – это фактически тот же запрос, который выполняется всякий раз при участии в какой-либо команде.

Представление – это предопределенный запрос, хранящийся в базе данных, который выглядит подобно обычной таблице и не требует для своего хранения дисковой памяти.

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

<определение_просмотра> ::=

```
{ CREATE| ALTER} VIEW имя_просмотра
   [(имя_столбца [,...n])]
   [WITH ENCRYPTION]
   AS SELECT_оператор
   [WITH CHECK OPTION]
```

Параметр *WITH ENCRYPTION* предписывает серверу шифровать SQL-код запроса, что гарантирует невозможность его несанкционированного просмотра и использования.

Параметр *WITH CHECK OPTION* предписывает серверу исполнять проверку изменений, производимых через представление, на соответствие критериям, определенным в операторе *SELECT*

Использование аргумента *WITH CHECK OPTION* гарантирует, что сделанные изменения будут отображены в представлении.

Пример 19. Создание представления: Показать в представлении клиентов из Москвы.

```
CREATE  VIEW view1 AS
SELECT КодКлиента, Фамилия, ГородКлиента
FROM Клиент
WHERE ГородКлиента='Москва'
    --Выборка данных из представления:
   SELECT * FROM view1
```

Выполним команду:

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

Пример 21. Создание представления с проверкой команд модификации

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

Представление удаляется командой:

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

Пример 22. Немодифицируемое представление с данными из разных таблиц

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

Пример 23. Немодифицируемое представление с группировкой и итоговыми функциями

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

Пример 24. Модифицируемое представление с вычислениями

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

Преимущества и недостатки представлений

Независимость от данных.
Актуальность.
Повышение защищенности данных
Снижение стоимости.
Дополнительные удобства.
Возможность настройки.
Обеспечение целостности данных

Недостатки использования представлений в среде SQL:

Ограниченные возможности обновления
Структурные ограничения
Снижение производительности
Защита данных с использованием представлений