

Реализация представлений

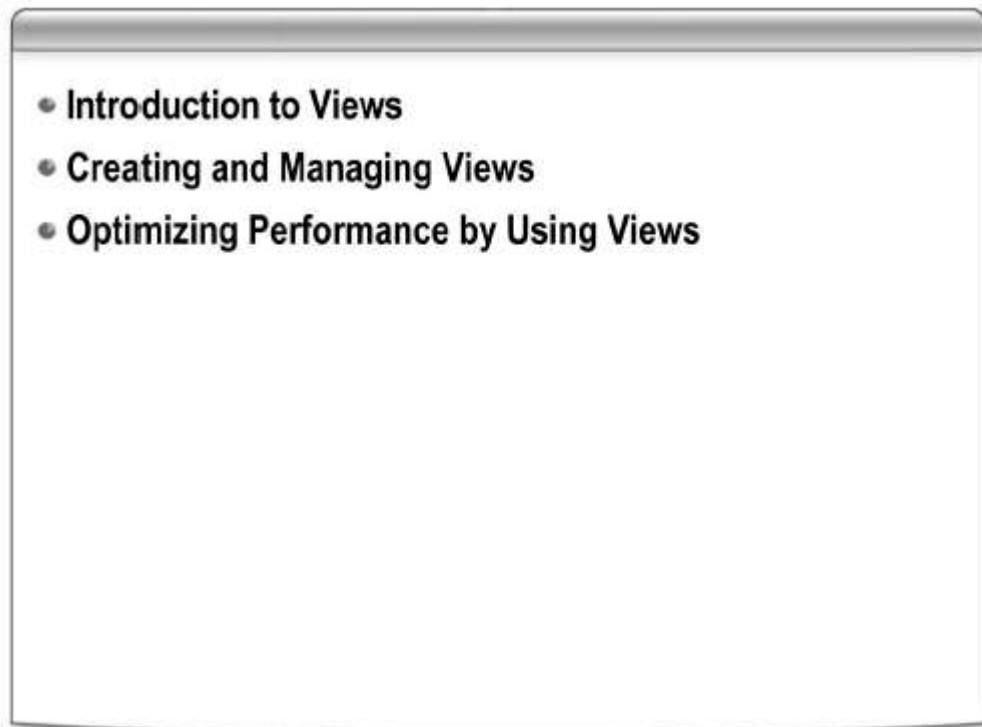
Содержание:

[Урок 1: Введение в представления](#)

[Урок 2: Создание представлений и управление ими](#)

[Урок 3: Оптимизация производительности с помощью представлений](#)

Тема: Реализация представлений



Цели темы

После завершения этой темы, студенты смогут:

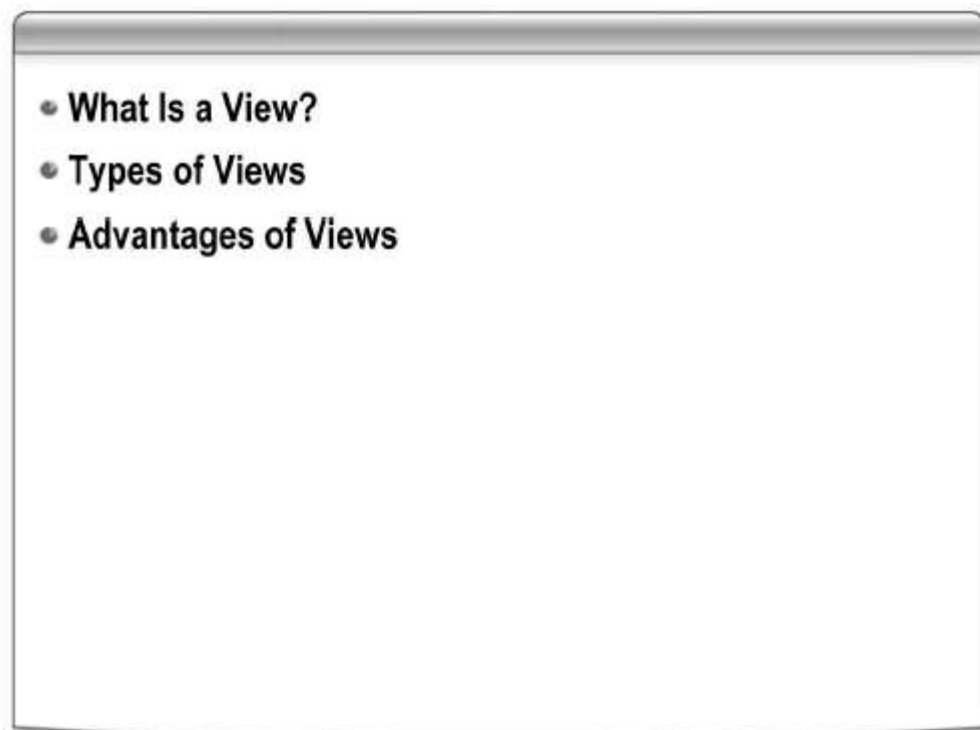
- Описывать типы и преимущества представлений.
- Создавать представления и управлять ими.
- Оптимизировать производительность с помощью представлений.

Введение

Большинство реализаций баз данных включают представления как удобный способ получения данных через предопределенный запрос. Изучив представления, Вы сможете облегчить доступ к данным для пользователей и сможете улучшить производительность Вашей базы данных.

Эта тема знакомит с представлениями и описывает преимущества, которые они предоставляют. Здесь описывается, как создать представления при использовании Microsoft® SQL Server™ Management Studio и Transact-SQL, доступные опции и как найти информацию о представлениях. Затем в теме рассматриваются ограничения на изменение данных через представления и то, как представления могут улучшить производительность базы данных.

Урок 1: Введение в представления



Цели урока

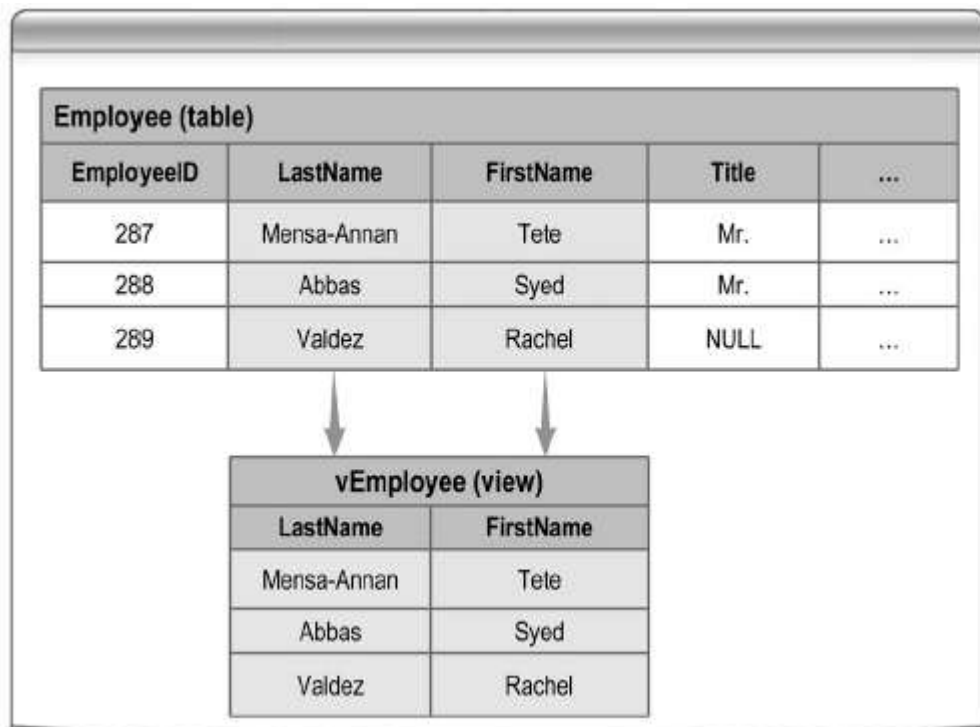
После завершения этого урока, студенты смогут:

- Объяснять, как используются представления.
- Описывать типы представлений.
- Описывать преимущества представлений.

Введение

В этом уроке определяется, что такое представление, описывается использование представлений, и различные их типы, доступные в Microsoft SQL Server 2005. Затем в этом уроке описываются преимущества использования представлений.

Что такое представление?



Определение

Представление - виртуальная таблица, содержание которой определяется запросом. Как реальная таблица, представление состоит из множества именованных столбцов и строк данных. Пока представление не индексировано, оно не существует как хранимый набор значений в базе данных. Строки и столбцы данных формируются из таблиц, на которые есть ссылки в запросе, определяющем представление и, производятся динамически при ссылке на представление. Таблицы, запрашиваемые в представлении, называют базовыми таблицами.

Общие примеры представлений:

- подмножество строк или столбцов базовой таблицы
- объединение двух или нескольких базовых таблиц
- соединение двух или нескольких базовых таблиц
- статистические итоги базовой таблицы
- подмножество другого представления или некоторая комбинация представлений и базовых таблиц

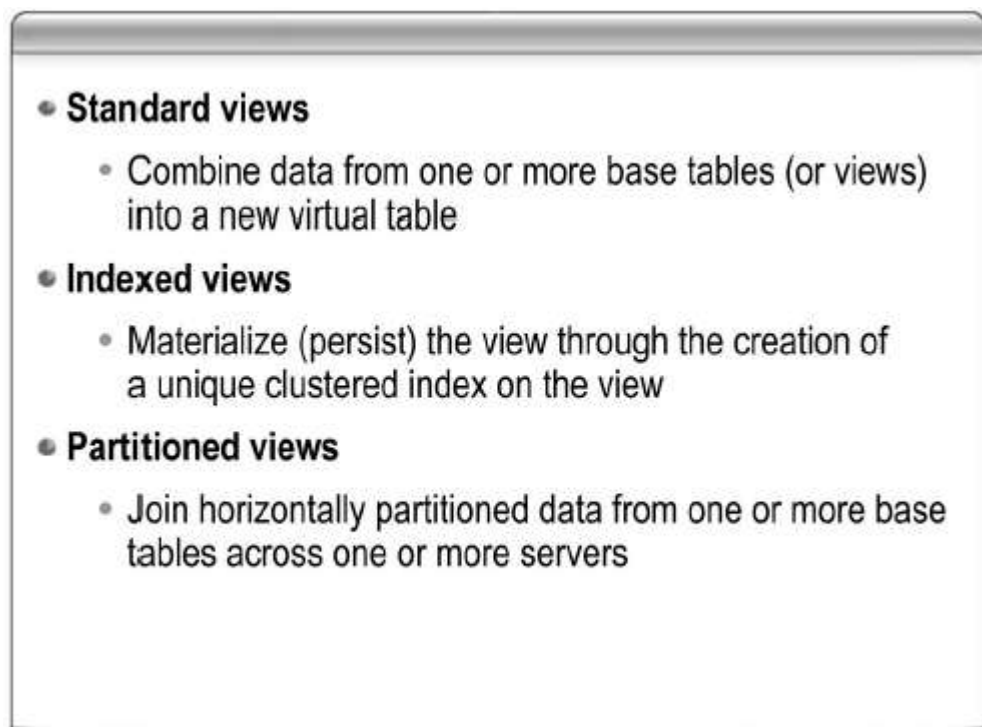
Использование представлений

Представление обычно используется, чтобы:

- Обеспечить возможность пользователям сосредотачиваться на определенных данных, которые их интересуют, и на определенных задачах, за которые они ответственны. Ненужные или уязвимые данные можно в представлении опустить.
- Упростить пользователям работу с данными. Вы можете определить как представления часто используемые в запросах операции соединения, проекции, объединения, выборки так, чтобы пользователи не должны будут определять все условия и квалификаторы каждый раз при выполнении дополнительных операций на этих данных.
- Улучшить безопасность, позволяя пользователям доступ к данным только через представление, без предоставления прав на непосредственный доступ к основным базовым таблицам представления.
- Обеспечить обратную совместимость, определяя представление для эмулирования таблицы, которая существует, но ее схема изменилась.
- Определить набор данных, который пользователь может экспортировать и импортировать в SQL Server.
- Обеспечить объединенное представление разделенных данных, то есть, похожих данных, хранимых во множестве таблиц.

Дополнительная информация. Дополнительную информацию о представлениях, см. в разделе “Understanding Views” SQL Server Books Online.

Типы представлений



Введение

В SQL Server 2005 существует три типа представлений:

- Стандартные представления
- Индексированные представления
- Секционированные представления

Стандартные представления

Стандартные представления объединяют данные от одной или более базовых таблиц в новой виртуальной таблице. Сохраняется только определение стандартного представления, а не его строки. Всякий раз при обработке ссылки на представление Двигатель Базы данных создает данные для него динамически. Стандартные представления – наиболее часто используемый тип представления.

Дополнительная информация. Дополнительную информацию о стандартных представлениях, см. в разделе “Scenarios for Using Views” в SQL Server Books Online.

Индексированные представления

Индексированное представление - представление, которое было материализовано, т.е. оно было вычислено и сохранено физически. Когда Вы индексируете представление, создается уникальный кластерный индекс на представление. Индексированные представления значительно улучшают работу некоторых типов запросов. Индексированные представления работают лучше всего на запросах, которые агрегируют много строк, но они плохо подходят для базовых таблиц, которые часто обновляются.

Дополнительная информация. Дополнительную информацию об индексированных представлениях, см. в разделе “Designing Indexed Views” в SQL Server Books Online.

Секционированные представления

Секционированное представление объединяет горизонтально разделенные данные от одной или более базовых таблиц, расположенных на одном или нескольких серверах. Это делает возможным похожие данные из множества базовых таблиц представить в виде одной таблицы. Представление, которое объединяет таблицы на одном экземпляре SQL Server, называют *локальным секционированным представлением*. Когда представление объединяет данные таблиц через несколько серверов, его называют *распределенным секционированным представлением*.

Замечание. Локальные секционированные представления включены в SQL Server 2005 только для обратной совместимости и находятся в упряднения. Для локального разделения данных предпочтительнее использовать метод секционирования таблиц. Дополнительную информацию см. в разделе “Partitioned Tables and Indexes” в SQL Server Books Online.

Дополнительная информация. Дополнительную информацию о секционированных представлениях, см. в разделе “Creating Partitioned Views” в SQL Server Books Online.

Преимущества представлений

- 
- Focus the data for a user
 - Mask database complexity
 - Simplify management of user permissions
 - Improve performance
 - Organize data for export to other applications

Преимущества использования представления

Представления предоставляют следующие преимущества:

Фокусировка данных для пользователя. Представления создают контролируемую среду окружения, которая позволяет доступ к определенным данным, в то время как другие данные скрыты. Данные, которые являются ненужными или уязвимыми, могут быть исключены из представления. Пользователи могут манипулировать отображением данных в представлении, как в обычной таблице. Кроме того, с надлежащими правами и некоторыми ограничениями, пользователи могут изменять данные представления.

Защита от сложности базы данных. Представления защищают от пользователя сложность проекта базы данных. Это предоставляет разработчикам возможность изменить проект без влияния на взаимодействие пользователей с базой данных. Кроме того, пользователи могут видеть более дружелюбную версию данных, потому что Вы можете создать представления с именами, которые легче понять, чем сокращенные имена, которые часто используются в базах данных.

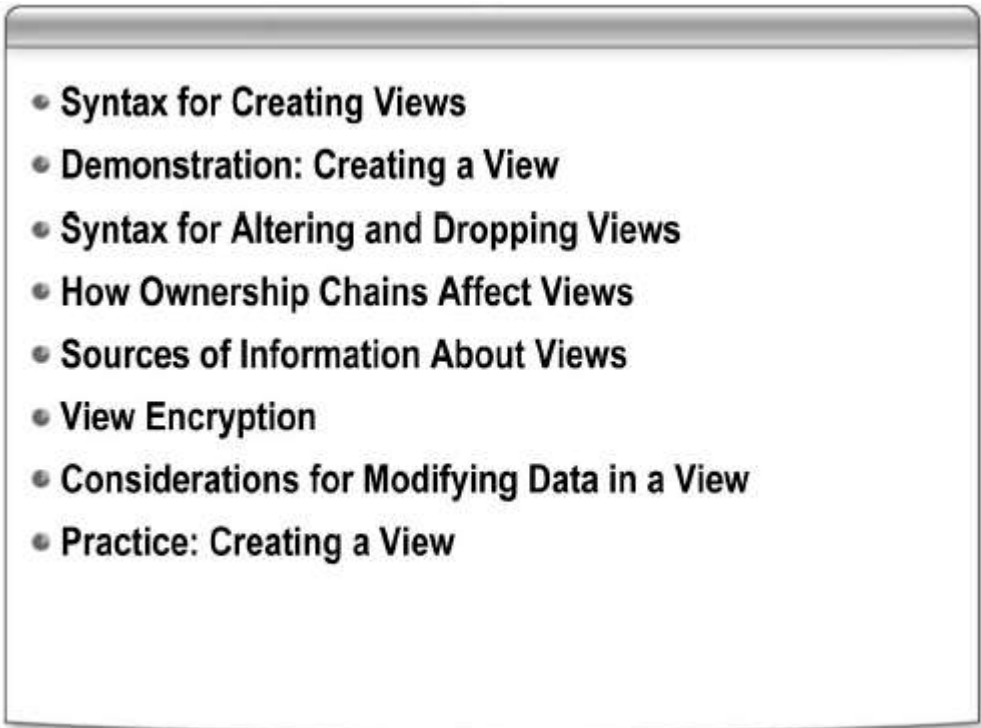
Сложные запросы, включая распределенные запросы к гетерогенным данным, могут также быть скрыты через представления. Пользователь выполняет запрос к представлению вместо того, чтобы писать запрос или выполнять скрипт.

Упрощение управления правами пользователей. Вместо предоставления прав пользователям выполнять запросы к определенным столбцам в базовых таблицах владельцы базы данных могут предоставить право для пользователей запрашивать данные только через представления. Это также защищает изменения в проекте основных базовых таблиц. Пользователи могут выполнять запросы к представлению без прерывания.

Улучшение производительности. Представления позволяют хранить результаты сложных запросов. Другие запросы могут использовать эти полученные в итоге результаты. Представления также позволяют Вам секционировать данные. Вы можете поместить индивидуальные секции на отдельных компьютерах и легко комбинировать их для пользователя.

Организация данных для экспорта в другие приложения. Вы можете создать представление со сложным запросом, который соединяет две или более таблиц, а затем экспортирует данные в другое приложение для дальнейшего анализа.

Урок 2: Создание представлений и управление ими

- 
- **Syntax for Creating Views**
 - **Demonstration: Creating a View**
 - **Syntax for Altering and Dropping Views**
 - **How Ownership Chains Affect Views**
 - **Sources of Information About Views**
 - **View Encryption**
 - **Considerations for Modifying Data in a View**
 - **Practice: Creating a View**

Цели урока

После завершения этого урока, студенты смогут:

- Описывать синтаксис создания представление.
- Объяснять, как создается представление.
- Описывать синтаксис для изменения или удаления представления.
- Описывать, как цепочки владения воздействуют на представления.
- Описывать источники информации о представлениях.
- Объяснять цели шифрования представления.
- Описывать, как обновляются данные через представление.

Введение

В этом уроке описывается, как создать, изменить и удалить представления. Здесь также описывается, как избежать разрыва цепочек владения, как скрыть определения представления и как получить информацию о представлениях внутри Вашей базы данных.

Синтаксис создания представления

- Use CREATE VIEW Transact-SQL statement:

```
CREATE VIEW [ schema_name. ] view_name [ ( column [ ,...n ] ) ]  
[ WITH [ ENCRYPTION ] [ SCHEMABINDING ] [ VIEW_METADATA ] ]  
AS select_statement [ ; ]  
[ WITH CHECK OPTION ]
```

- Restrictions:

- Cannot nest more than 32 levels deep
- Cannot contain more than 1,024 columns
- Cannot use COMPUTE, COMPUTE BY, or INTO
- Cannot use ORDER BY without TOP

Введение

Создать представление можно, выполнив оператор Transact-SQL CREATE VIEW, или используя графический интерфейс SQL Server Management Studio. Как часть определения представления, необходимо задать его содержание, используя оператор SELECT.

Совет Вы должны разработать последовательную систему именования объектов, чтобы отличать представления от таблиц. Например, Вы можете добавить символ 'v' или слово 'view' как префикс к имени каждого представления, которое Вы создаете. Этот подход позволяет легко отличать таблицы от представлений.

Создание представления

Чтобы создать представление, используйте View Designer в SQL Server Management Studio или оператор Transact-SQL CREATE VIEW. Открыть View Designer в Object Explorer, раскройте базу данных, с которой Вы хотите работать, правым щелчком раскройте узел **Views**, затем щелкните **New View**. Затем Вы можете проектировать свое представление при использовании графического интерфейса, с помощью которого Вы можете выбирать таблицы и столбцы, чтобы включить в представление, определять связи между столбцами, ограничивать возвращаемые строки и формировать различные опции, например, такие как

псевдонимы столбцов и порядок сортировки, которые используются, чтобы сгенерировать представление.

Оператор CREATE VIEW имеет следующий синтаксис.

```
CREATE VIEW [ schema_name . ] view_name [ (column [ ,...n ] ) ]  
[ WITH [ ENCRYPTION ] [, SCHEMABINDING ] [, VIEW_METADATA ] ]  
AS select_statement [ ; ]  
[ WITH CHECK OPTION ]
```

Следующий код демонстрирует создание представления **HumanResources.vEmployee**, которое состоит из столбцов нескольких таблиц в базе данных **AdventureWorks**.

```
CREATE VIEW [HumanResources].[vEmployee]  
AS  
SELECT  
    e.[EmployeeID],c.[Title],c.[FirstName],c.[MiddleName],c.[LastName]  
    ,c.[Suffix],e.[Title] AS [JobTitle],c.[Phone],c.[EmailAddress]  
    ,c.[EmailPromotion],a.[AddressLine1],a.[AddressLine2],a.[City]  
    ,sp.[Name] AS [StateProvinceName],a.[PostalCode]  
    ,cr.[Name] AS [CountryRegionName],c.[AdditionalContactInfo]  
FROM [HumanResources].[Employee] e  
    INNER JOIN [Person].[Contact] c  
    ON c.[ContactID] = e.[ContactID]  
    INNER JOIN [HumanResources].[EmployeeAddress] ea  
    ON e.[EmployeeID] = ea.[EmployeeID]  
    INNER JOIN [Person].[Address] a  
    ON ea.[AddressID] = a.[AddressID]  
    INNER JOIN [Person].[StateProvince] sp  
    ON sp.[StateProvinceID] = a.[StateProvinceID]  
    INNER JOIN [Person].[CountryRegion] cr  
    ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
```

Дополнительная информация. Дополнительную информацию о создании представлений, см. в разделе “Query and View Designer (Visual Database Tools)” и “CREATE VIEW (Transact-SQL)” в SQL Server Books Online.

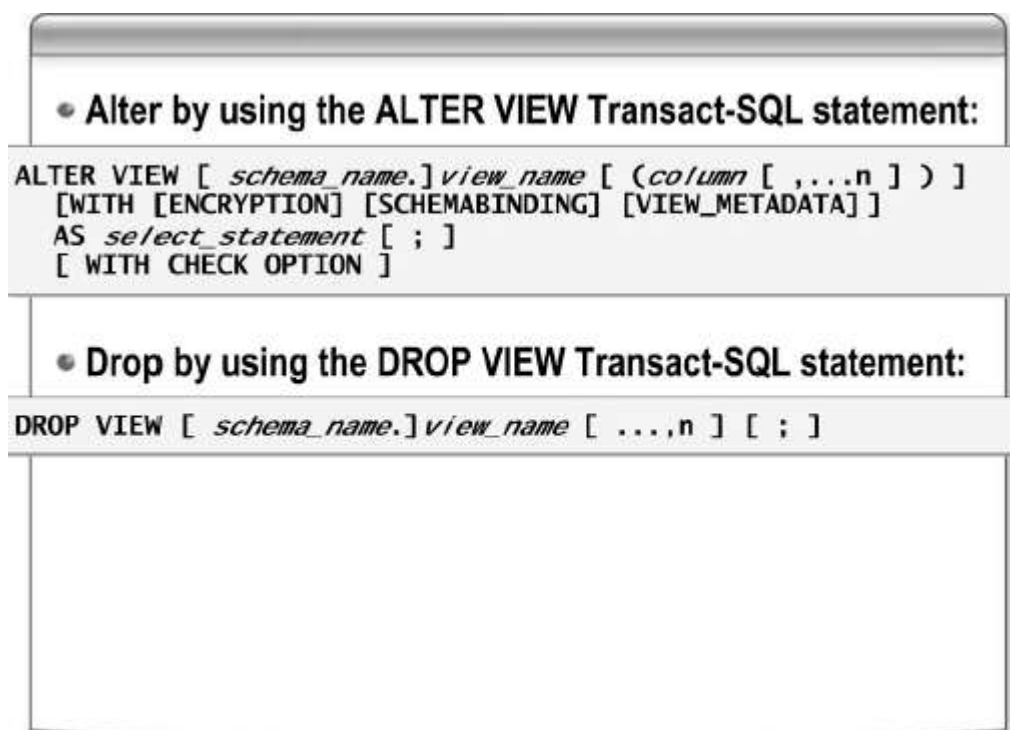
Требования для создания представлений

Помните о следующих фактах, когда создаете представления:

- Вы должны быть членом роли **sysadmin**, или **db_owner**, или **db_ddladmin**, или Вам должно быть предоставлено право **CREATE VIEW** в базе данных и право **ALTER SCHEMA** на схеме, в которой следует создать представление. Также, у Вас должно быть право **SELECT** на все таблицы или представления, на которые ссылается создаваемое представление.
- Вы можете создать представления только в текущей базе данных.
- Имя представления должно следовать правилам для идентификаторов и должно отличаться от любого другого имени представления или таблицы в базе данных.
- Вы можете строить представления на других представлениях. Вложение не может превысить 32 уровня, но может быть также ограничено сложностью представлений и доступной памятью.
- Представление может содержать максимум 1 024 столбцов.
- Вы должны определить имена столбцов, если:
 - какой-нибудь столбец представления получен из арифметического выражения, встроенной функции, или является константой.
 - какие-нибудь столбцы в соединяемых таблицах имеют одни и те же имена.
- Нельзя создать временные представления и представления на временных таблицах.
- Нельзя связать объекты **Rule** или **Default** с представлением.
- Нельзя связать триггеры **AFTER** с представлениями, только триггеры **INSTEAD OF**.
- Нельзя включать разделы **COMPUTE** или **COMPUTE BY** или ключевое слово **INTO** в запросе, который определяет Ваше представление.
- Нельзя включать раздел **ORDER BY** в запрос, который определяет представление, если нет раздела **TOP** в списке выбора оператора **SELECT**.
- Нельзя включать раздел **OPTION**, определяющий подсказку в запросе, который определяет представление.
- Нельзя включать раздел **TABLESAMPLE** в запрос, который определяет представление.

Дополнительная информация. Дополнительную информацию о требованиях к созданию представления, см. в разделе “Designing and Implementing Views” в SQL Server Books Online.

Синтаксис изменения и удаления представлений



Введение

Если Вам нужно изменить представление, Вы можете это сделать, используя SQL Server Management Studio или выполняя оператор Transact-SQL ALTER VIEW. Если представление Вам больше не нужно, Вы можете удалить его определение из базы данных, используя SQL Server Management Studio или выполняя оператор Transact-SQL DROP VIEW.

Изменение представлений

Вы можете изменить определение своего представления, открывая инструмент View Designer в Object Explorer или используя оператор Transact-SQL ALTER VIEW. Вы можете изменить состав таблиц и столбцов, включенных в представление, столбцы связи, ограничения строк, которые возвращает представление, различные опции, такие как псевдонимы столбцов и порядок сортировки, используемые для генерации представления.

Оператор ALTER VIEW изменяет определение представления, включая индексированные представления, не затрагивая зависимые хранимые процедуры и триггеры. Это позволяет Вам сохранять права для представления. Этот оператор имеет те же ограничения, как и оператор CREATE VIEW.

Оператор CREATE VIEW имеет следующий синтаксис.

```

ALTER VIEW [ schema_name . ] view_name [ ( column [ ,...n ] ) ]
[ WITH <view_attribute> [ ,...n ] ]
AS select_statement [ ; ]
[ WITH CHECK OPTION ]

<view_attribute> ::=
{
    [ ENCRYPTION ]
    [ SCHEMABINDING ]
    [ VIEW_METADATA ]
}

```

Следующий код изменяет представление **HumanResources.vEmployee** в базе данных **AdventureWorks**, он удаляет из представления всю информацию о почтовом адресе.

```

ALTER VIEW [HumanResources].[vEmployee]
AS
SELECT
    e.[EmployeeID]
    ,c.[Title]
    ,c.[FirstName]
    ,c.[MiddleName]
    ,c.[LastName]
    ,c.[Suffix]
    ,e.[Title] AS [JobTitle]
    ,c.[Phone]
    ,c.[EmailAddress]
FROM [HumanResources].[Employee] e
    INNER JOIN [Person].[Contact] c
    ON c.[ContactID] = e.[ContactID]

```

Дополнительная информация. Дополнительную информацию об изменении представлений, см. в разделе “Query and View Designer Tools (Visual Database Tools)” и “ALTER VIEW (Transact-SQL)” в SQL Server Books Online.

Удаление представлений

При удалении представления удаляются его определение и все права на него. Кроме того, если пользователи запрашивает какие-нибудь представления, которые ссылаются на удаленное представление, они получают сообщение об ошибке. Однако, удаление таблицы, на которую ссылается представление, не удаляет представление автоматически. Вы должны удалить представление явно. Вы можете удалить представление, удаляя его в Object Explorer или, используя оператор DROP VIEW.

Оператор DROP VIEW имеет следующий синтаксис.

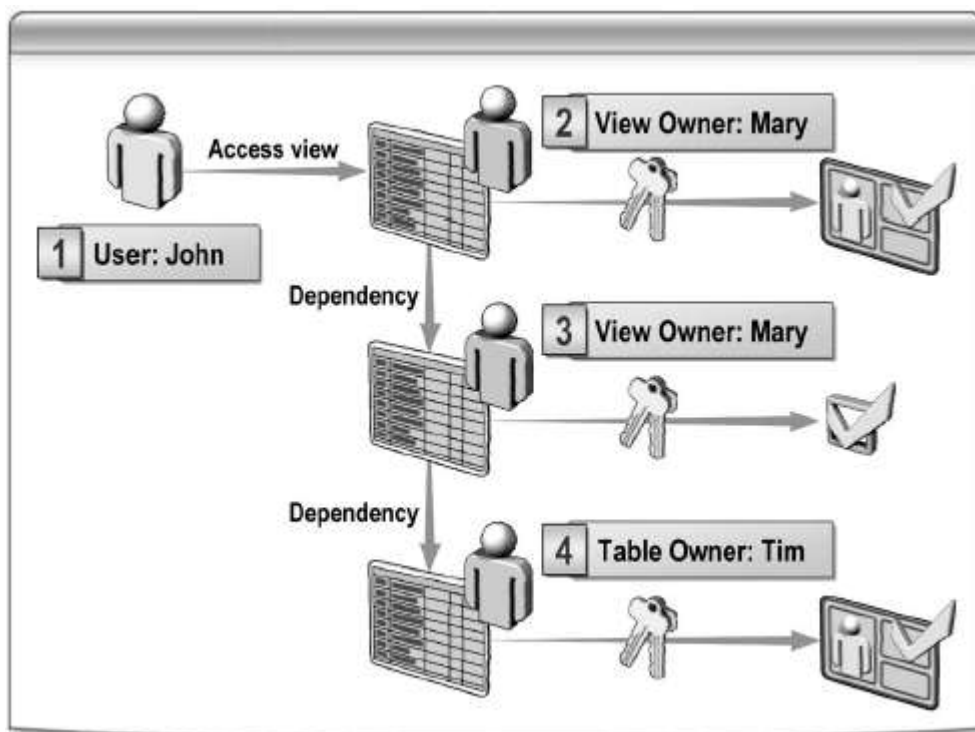
```
DROP VIEW [ schema_name . ] view_name [ ...,n ] [ ; ]
```

Следующий код демонстрирует удаление представления **HumanResources.vEmployee** из базы данных **AdventureWorks**.

```
DROP VIEW [HumanResources].[vEmployee]
```

Дополнительная информация. Дополнительную информацию о синтаксисе оператора DROP VIEW, см. в разделе, что “ DROP VIEW (Transact-SQL)” SQL Server Books Online.

Как цепочка владения влияет на представление



Введение

Представления зависят от таблиц и от других представлений. Когда SQL Server вычисляет содержимое представления, он проходит иерархию зависимостей таблиц и представлений, чтобы получить соответствующее содержимое. Когда множество объектов базы данных последовательно вызывают друг друга, эта последовательность называется *цепочкой*. SQL Server оценивает права доступа к объектам в цепочке иначе, чем если бы эти объекты были вызваны отдельно.

Замечание. Вы можете так сконфигурировать SQL Server, чтобы разрешить формирование цепочки владения между указанными базами данных или между всеми базами данных внутри одного экземпляра SQL Server. По умолчанию возможность формирования цепочек владения между различными базами данных отключена, и не следует включать ее без особой необходимости.

Как SQL Server проверяет права в цепочке владения

Когда пользователь получает доступ к объекту базы данных (такому как представление) и он вызывает другой объект (такой как таблица), SQL Server не проверяет права пользователя на второй объект, если владелец обоих объектов – один и тот же. Это означает, что когда пользователь имеет право доступа к представлению, SQL Server не оценивает права

пользователя на каждую вложенную таблицу или представление до тех пор, пока они принадлежат владельцу первого вызванного представления. Когда SQL Server вызывает объект с другим владельцем, только тогда оцениваются права пользователя на этот объект.

Формирование цепочки владения позволяет Вам управлять доступом к множеству объектов, таких как таблицы, устанавливая права на один объект – представление. Цепочки владения также позволяют повысить производительность сценариев за счет того, что не проверяются права доступа к объектам.

Чтобы избежать разрыва цепочек владения, необходимо удостовериться в том, что все представления внутренние таблицы и функции принадлежат одному владельцу.

Дополнительная информация. Дополнительную информацию о цепочках владения, см. в разделе “Ownerships Chains” в SQL Server Books Online.

Источники информации о представлениях

• SQL Server Management Studio	
Source	Information
Object Explorer	List of views in database
	Access to columns, triggers, indexes, and statistics defined on views
View Properties dialog box	Properties of individual views
• Transact-SQL	
Source	Information
sys.views	List of views in database
sp_helptext	Definition of non-encrypted views
sys.sql_dependencies	Objects (including views) that depend on other objects

Введение

Перед тем, как создать, изменить, или удалить представление, Вам может потребоваться информация о существующих представлениях. В SQL Server 2005 Вы можете использовать следующие источники, чтобы получить информацию о представлениях:

- SQL Server Management Studio
- Представление каталога **sys.views**
- Системная хранимая процедура **sp_helptext**
- Представление каталога **sys.sql_dependencies**

Получение информации о представлениях при использовании SQL Server Management Studio

Чтобы получить информацию о представлении при использовании SQL Server Management Studio, откройте Object Explorer и раскройте базу данных, с которой Вы хотите работать. Раскройте узел Views, чтобы видеть список доступных представлений. Раскройте определенное представление для доступа к столбцам, триггерам, индексам, и статистикам, определенным для представления. Или правым щелчком на представлении отобразите

контекстное меню, которое позволяет Вам генерировать скрипты, для создания, изменения, и удаления представления, а также позволяет открыть диалоговое окно свойств представления Property.

Дополнительная информация. Дополнительную информацию о работе с представлениями в SQL Server Management Studio, см. в разделе “Working with Views (Visual Database Tools)” в SQL Server Books Online.

Получение списка представлений при использовании Transact-SQL

Вы можете запросить представление системного каталога **sys.views**, чтобы получить общую информацию о доступных представлениях. Результирующий набор будет содержать по одной строке для каждого доступного представления. Следующий пример возвращает список всех доступных представлений в базе данных **AdventureWorks**.

```
USE AdventureWorks
GO
SELECT * FROM sys.views
```

Дополнительная информация. Дополнительную информацию о представлении системного каталога **sys.views**, см. в разделе “sys.views (Transact-SQL)” в SQL Server Books Online.

Получение определения представления при использовании Transact-SQL

Чтобы получить текст незашифрованного представления, используйте системную хранимую процедуру **sp_helptext**, передавая имя представления как аргумент. Хранимая процедура **sp_helptext** возвратит ошибку, если указанное представление будет зашифровано. Следующий пример возвращает текст представления **HumanResources.vEmployee** базы данных **AdventureWorks**.

```
USE AdventureWorks
GO
EXEC sp_helptext 'HumanResources.vEmployee'
```

Дополнительная информация. Дополнительную информацию о хранимой процедуре **sp_helptext**, см. в разделе “sp_helptext (Transact-SQL)” в SQL Server Books Online.

Определение зависимостей представления при использовании Transact-SQL

Прежде, чем удалить любую таблицу или представление, Вы должны проверить, нет ли какого-нибудь зависимого от них представления при использовании представления системного каталога **sys.sql_dependencies**. Представление **sys.sql_dependencies** содержит по одной строке для каждого объекта базы данных (например, представления), который ссылается на указанную таблицу или представление.

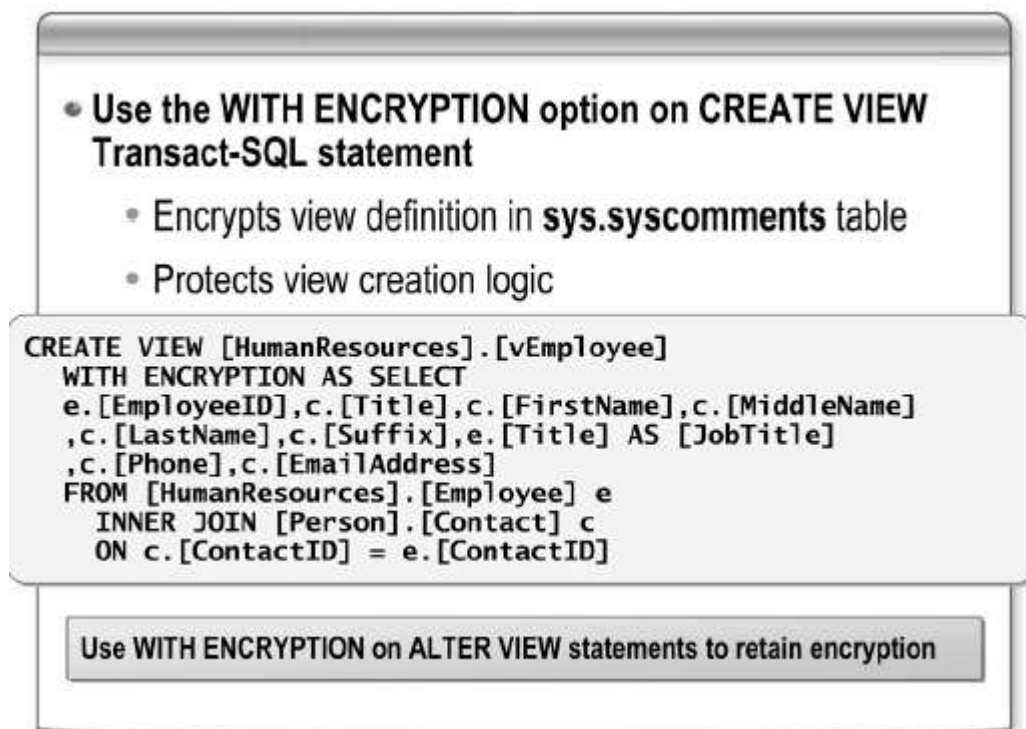
Следующий пример демонстрирует, как получить список всех объектов базы данных, которые имеют прямые зависимости от таблицы **HumanResources.Employee** базы данных **AdventureWorks**.

```
USE AdventureWorks

GO

SELECT DISTINCT OBJECT_NAME(object_id) AS Name
FROM sys.sql_dependencies
WHERE referenced_major_id =
      OBJECT_ID(N'AdventureWorks.HumanResources.Employee')
```

Шифрование представления



Цель шифрования представления

Текст, используемый в операторах **CREATE VIEW** или **ALTER VIEW** для определения представления, сохраняется в системной таблице **sys.syscomments**. Чтобы защитить логику определения представления, необходимо определить опцию **WITH ENCRYPTION**. Эта опция зашифрует текст, сохраненный в **sys.syscomments** так, что никто не сможет его прочитать.

Совет. Прежде, чем Вы создадите зашифрованное представление, Вы должны всегда сохранить копию операторов **CREATE VIEW** или **ALTER VIEW** в безопасном месте; иначе Вы не сможете получить доступ к определению представления, если оно Вам понадобится.

Пример шифрования представления

Следующий код изменяет представление **HumanResources.vEmployee** в базе данных **AdventureWorks** с опцией шифрования.

```
ALTER VIEW [HumanResources].[vEmployee]
WITH ENCRYPTION
AS
SELECT
```

```
e.[EmployeeID],c.[Title],c.[FirstName],c.[MiddleName]  
,c.[LastName],c.[Suffix],e.[Title] AS [JobTitle]  
,c.[Phone],c.[EmailAddress]  
FROM [HumanResources].[Employee] e  
INNER JOIN [Person].[Contact] c  
ON c.[ContactID] = e.[ContactID]
```

Совет. Если Вы создаете зашифрованное представление, то когда будете его изменять, Вы также должны определять опцию WITH ENCRYPTION; иначе шифрование будет отключено.

Основные сведения об изменении данных в представлении

- **Views do not maintain a separate copy of data (indexed views are an exception)**
- **Updates to views modify base tables**
- **Restrictions:**
 - Cannot affect more than one base table
 - Cannot modify columns derived from aggregate functions or calculations
 - Cannot modify columns affected by GROUP BY, HAVING, or DISTINCT clauses
- **Updates to views are restricted by using the WITH CHECK OPTION**

Введение

Представления не содержат отдельной копии данных. Вместо этого они показывают результирующий набор запроса на одной или нескольких базовых таблицах. Поэтому, всякий раз, когда Вы изменяете данные в представлении, Вы фактически изменяете базовую таблицу. С некоторыми ограничениями, Вы вполне можете вставить, обновить, или удалить данные таблиц через представление.

Ограничения на изменение данных в представлении

При изменении данных в представлении, действуют следующие ограничения:

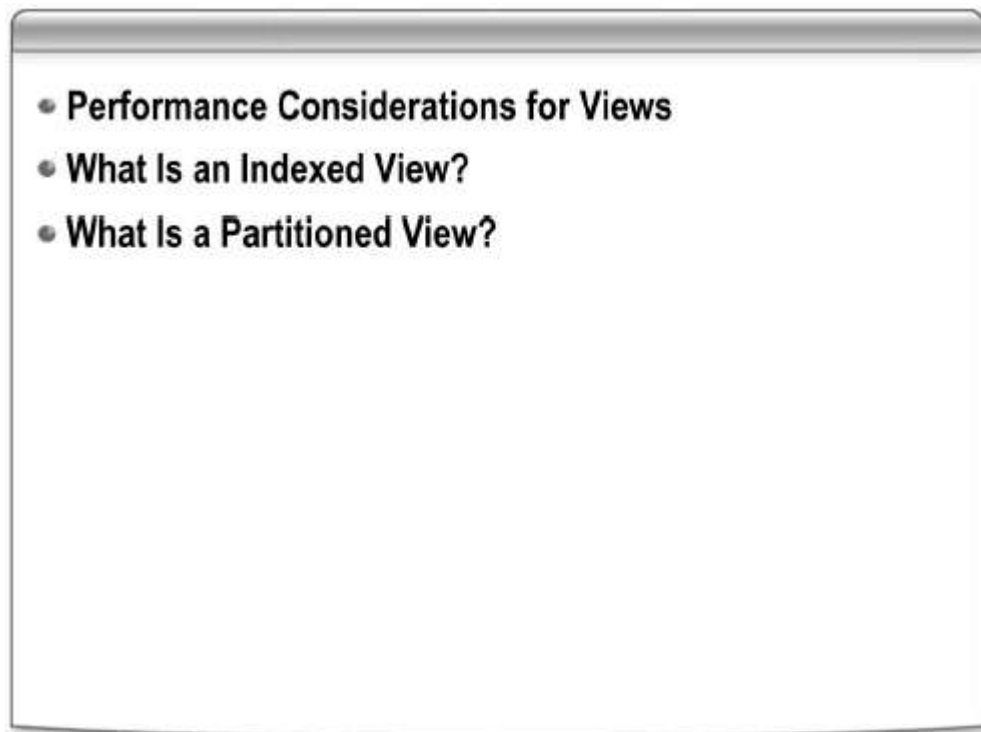
- Любые модификации, включая операторы INSERT, UPDATE, и DELETE, должны ссылаться на столбцы только одной базовой таблицы.
- Любые изменяемые столбцы должны непосредственно ссылаться на данные столбцов в таблице. Они не могут быть получены никаким другим способом, таким как с помощью агрегатной функции или с помощью вычислений по другим столбцам.
- Любые изменяемые столбцы не может быть использованы в разделах GROUP BY, HAVING или DISTINCT.

Использование CHECK OPTION для ограничения обновлений

Если в определении представления используется CHECK OPTION, то все операторы модификации данных, выполненные на представлении, должны придерживаться критериев выборки оператора SELECT, определяющего представление. Если используется опция CHECK OPTION, строки не могут быть изменены в том случае, если они исчезнут из представления. Любая модификация, которая к этому приводит, отменяется, и возникает соответствующая ошибка.

Дополнительная информация. Дополнительную информацию об изменении данных в представлениях, см. в разделе “Modifying Data Through a View” в SQL Server Books Online.

Урок 3: Оптимизация производительности при использовании представлений



Цели урока

После завершения этого урока, студенты смогут:

- Описывать основные сведения о производительности представлений.
- Объяснять цель индексируемого представления и когда оно используется.
- Описывать секционированные представления и их преимущества.

Введение

Этот урок описывает основные сведения об использовании представлений и как представления позволяют оптимизировать производительность, сохраняя результаты сложных запросов и разделяя данные.

Основные сведения о производительности представлений

- **Views introduce performance overhead because views are resolved dynamically**
- **Nested views introduce risk of performance problems**
 - Review definition of unencrypted nested views
 - Use SQL Server Profiler to review performance
- **Indexed views and partitioned views can improve performance**

Основные сведения о производительности представлений

При использовании стандартных представлений, возникают присущие им издержки производительности, так как каждый раз, когда выполняется стандартное представление, SQL Сервер должен выполнить оператор SELECT на одной или нескольких таблицах, чтобы получить данные для представления. Эти издержки увеличиваются, если представления вложенные. Вы должны быть осторожными, так как непреднамеренное использование длинных цепочек вложенных представлений окажет существенное влияние на работу запроса.

Вы можете легко определить, являются ли представления вложенными, просмотрев определение представления, чтобы понять, на каких таблицах или представлениях оно формируется. Однако, когда представление зашифровано, невозможно просмотреть его определение. Чтобы оценить производительность представления и определить действия, выполняемые зашифрованным вложенным представлением, Вы можете использовать SQL Server Profiler.

Дополнительная информация. Дополнительную информацию об улучшении производительности представления, см. в разделе “Using SQL Server Profiler” в SQL Server Books Online.

Улучшение производительности представлений

Чтобы улучшить производительность представлений, обычно используются два подхода,:

- Индексированные представления
- Секционированные представления

Что такое индексированное представление?

- **A view with a unique clustered index**
 - Materializes view, improving performance
 - Allows query optimizer to use view in query resolution
- **Use when:**
 - Performance gains outweigh maintenance overhead
 - Underlying data is modified infrequently
 - Queries perform a significant number of joins and aggregations

```
CREATE UNIQUE CLUSTERED INDEX  
[IX_vStateProvinceCountryRegion] ON  
[Person].[vStateProvinceCountryRegion]  
( [StateProvinceID] ASC, [CountryRegionCode] ASC)
```

Определение

Индексированное представление – представление, у которого есть уникальный кластерный индекс. Индексированное представление хранит результирующий набор представления на страницах индекса листового уровня. SQL Server может обратиться к индексу, чтобы быстро получить данные представления.

Выигрыш в производительности индексированных представлений

Поскольку модификации данных делаются в базовых таблицах, эти модификации отражаются на данных, сохраненных в индексированном представлении. Требование того, что кластерный индекс представления должен быть уникальным, улучшает эффективность, с которой SQL Server может найти строки индекса, затронутые любой модификацией данных. Из-за улучшения времени поиска, Вы можете использовать индексированные представления, чтобы улучшить производительность запроса.

Дополнительная информация. Дополнительную информацию о реализации индексированного представления см. в разделе “Designing Indexed Views” в SQL Server Books Online.

Другое преимущество создания индекса на представлении заключается в том, что оптимизатор запросов начинает использовать индекс представления в запросах, в которых не указывается непосредственно имя этого представления в разделе FROM. Существующие запросы могут повысить эффективность поиска данных от индексируемого представления без необходимости изменения их кода.

Дополнительная информация. Дополнительную информацию о том, как оптимизатор запросов использует представления, см. в разделе “Resolving Indexes on Views” в SQL Server Books Online.

Соглашения по использованию индексируемых представлений

Неудобство создания индексируемых представлений состоит в издержках на обслуживание индекса. Следует использовать индексируемые представления в следующих случаях:

- выигрыш в производительности запросов перевешивает издержки на обслуживание.
- данные представления обновляются нечасто.
- запросы выполняют значительное количество соединений и агрегаций, которые либо обрабатывают большое количество строк, либо выполняются часто многими пользователями.

Требования для индексируемых представлений

Существует ряд требований и ограничений, связанных с созданием индексируемых представлений. Вот главные требования:

- первый индекс, который Вы создаете на представлении, должен быть уникальным кластеризованным индексом
- представление должно быть определено с опцией SCHEMABINDING. Схема закрепления связывает представление со схемой основных базовых таблиц.
- представление может ссылаться на базовые таблицы, но оно не может ссылаться на другие представления.
- базовые таблицы, на которые ссылается представление, должны быть в той же самой базе данных и иметь того же самого владельца, что и представление.
- на таблицы и определенные пользователем функции, которые используются в представлении, нужно ссылаться двухчастными именами (имя схемы.имя объекта). Одно-, трех-, и четырехчастные имена недопустимы.

- функции, на которые ссылаются выражения в представлении, должны быть детерминированными.

Дополнительная информация Дополнительную информацию о создании внесенных в указатель представлений, см. в разделе “Создание Внесенных в указатель Представлений” SQL Server Books Online.

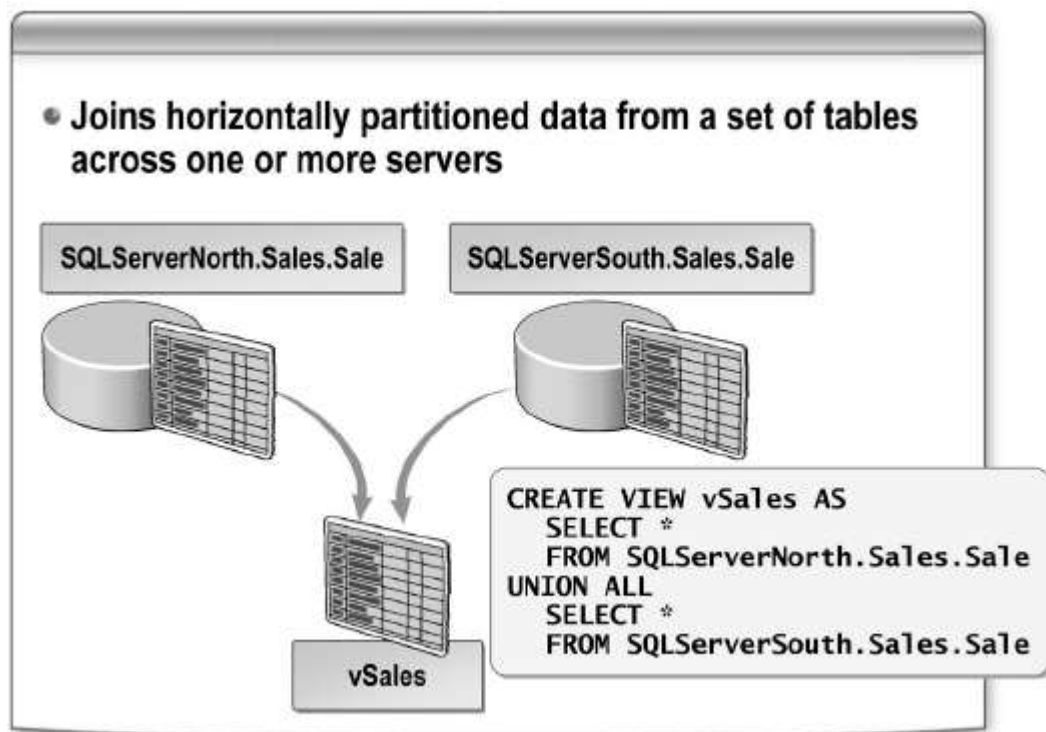
Пример индексированного представления

Индексы на представлениях можно создать при использовании команды CREATE INDEX. Следующий пример кода создает уникальный кластеризованный индекс с именем **IX_vStateProvinceCountryRegion** на представлении **Person.vStateProvinceCountryRegion** базы данных **AdventureWorks**.

```
USE [AdventureWorks]
GO
CREATE UNIQUE CLUSTERED INDEX [IX_vStateProvinceCountryRegion] ON
[Person].[vStateProvinceCountryRegion]
(
    [StateProvinceID] ASC, [CountryRegionCode] ASC
)
```

Дополнительная информация. Дополнительную информацию о команде CREATE INDEX см. в разделе " CREATE INDEX (Transact-SQL)" в SQL Server Books Online.

Что такое секционированное представление?



Определение

Секционированное представление соединяет горизонтально разделенные данные строк из набора таблиц-участников с одного или более серверов, возвращая данные как будто одной таблицы. Секционированное представление использует фразу **UNION ALL**, чтобы объединить результаты команд **SELECT** по всем таблицам-участникам в один результирующий набор.

Типы секционированных представлений

SQL Server 2005 различает локальные и распределенные секционированные представления. В локальном секционированном представлении все участвующие таблицы и представления располагаются на одном и том же экземпляре SQL Server. Локальные секционированные представления включены в SQL Server 2005 только для обратной совместимости. Предпочтительнее использовать секционированные таблицы для разделения данных одного и того же сервера.

Дополнительная информация Дополнительную информацию о секционированных таблицах, см. в разделе “Partitioned Tables and Indexes” в SQL Server Books Online.

В распределенном секционированном представлении, по крайней мере, одна из участвующих таблиц хранится на другом (удаленном) сервере. Кроме того, SQL Server 2005 различает секционированные представления, которые являются модифицируемыми и представлениями, которые являются копиями только для чтения основных таблиц.

Дополнительная информация. Дополнительную информацию о секционированных представлениях см. в разделе “Using Partitioned Views” в SQL Server Books Online.

Преимущества секционированных представлений

Если таблицы в секционированном представлении находятся на различных серверах, или на многопроцессорном компьютере, каждая таблица, включенная в запрос, может быть просмотрена параллельно, таким образом улучшая производительность запроса. Кроме того, такие задачи обслуживания, как перестройка индексов или восстановление таблиц, могут быть выполнены быстрее.

Замечание. Нельзя создать индекс на секционированном представлении. Определение индекса для представления требует использовать названия с двумя частями; Секционированное же представление требует использование трех- или четырехчастных имен, т.е. *servername.databasename.schema.objectname*.

Федеративные и разделенные серверы

Распределенные секционированные представления используются, чтобы реализовать *федерацию* серверов баз данных. Федерация – группа серверов, которыми управляют независимо, но они кооперируются для разделения обработки нагрузки на систему. Формирование федерации серверов базы данных путем разделения данных является механизмом, который позволяет масштабировать набор серверов для поддержки требования обработки больших, многосвязных (multitiered) вебсайтов.

Дополнительная информация. Дополнительную информацию о федеративных серверах базы данных см. в разделе “Federated Database Servers” в SQL Server Books Online.