**Лабораторная работа №6.**

**Создание и использование представлений.**

*Цель**работы*:

приобретение практических навыков по созданию представлений средствами языка SQL, использованию представлений в запросах.

*Краткие теоретические сведения.*

1. **Инструкции DDL и представления**

Представления — это объекты базы данных, которые создаются на основе одной или

нескольких таблиц (или других представлений). При этом физически сохраняются только имя представления и запросы для отбора строк в представление. По этой причине представления также называются виртуальными таблицами.

1.1. **Создание представления**

Представление создается посредством инструкции CREATE VIEW, синтаксис которой выглядит следующим образом:

CREATE VIEW view\_name [(column\_list)]

[WITH {ENCRYPTION SCHEMABINDING | VIEW\_METADATA}]

AS select\_statement

[WITH CHECK OPTION]

Параметр view\_name задает имя определяемого представления, а в параметре column\_list указывается список имен, которые будут использоваться в качестве имен столбцов представления. Если этот необязательный параметр опущен, то используются имена столбцов таблиц, по которым создается представление. Параметр select\_statement задает инструкцию SELECT, которая извлекает строки и столбцы из таблиц (или других представлений). Параметр WITH ENCRYPTION задает шифрование инструкции SELECT, повышая таким образом уровень безопасности системы баз данных.

Предложение SCHEMABINDING привязывает представление к схеме таблицы, по которой оно создается. Когда это предложение указывается, имена объектов баз данных в инструкции SELECT должны состоять из двух частей, т. е. в виде owner.db\_object, где owner — владелец, а db\_object может быть таблицей, представлением или определяемой пользователем функцией.

Любая попытка модифицировать структуру представлений или таблиц, на которые ссылается созданное таким образом представление, будет неудачной. Чтобы такие таблицы или представления можно было модифицировать (инструкцией ALTER) или удалять (инструкцией DROP), нужно удалить это представление или убрать из него предложение SCHEMABINDING.

Когда при создании представления указывается параметр VIEW\_METADATA, все его столбцы можно обновлять (за исключением столбцов с типом данных TIMESTAMP), если представление имеет триггеры INSERT или UPDATE INSTEAD OF.

Представления можно использовать для следующих целей.

1. Для ограничения использования определенных столбцов и/или строк таблиц. Таким образом, представления можно использовать для управления доступом к определенной части одной или нескольких таблиц.

2. Для скрытия подробностей сложных запросов. Если для приложения базы данных требуются запросы со сложными операциями соединения, создание соответствующих представлений может упростить такие запросы.

3. Для ограничения вставляемых или обновляемых значений некоторым диапазоном.

В примере 7.1 показано создание представления.

Пример 7.1. Создание представления

USE sample;

GO

CREATE VIEW v\_clerk

AS SELECT emp\_no, project\_no, enter\_date

FROM works\_on

WHERE job = 'Clerk';

Запрос в примере 7.1 выбирает из таблицы works\_on строки, удовлетворяющие условию job='Clerk'. Представление v\_clerk определяется строками и столбцами, возвращаемыми этим запросом. В табл. 7.1 отображена таблица works\_on, в которой строки, выбранные в представлении v\_clerk, выделены жирным шрифтом.

Таблица 7.1. Базовая таблица works\_on

|  |  |  |  |
| --- | --- | --- | --- |
| **emp\_no** | **project\_no** | **job** | **enter\_date** |
| 10102 | p1 | Analyst | 2006.10.1 00:00:00 |
| 10102 | p3 | Manager | 2007.1.1 00:00:00 |
| 25348 | p2 | Clerk | 2007.2.15 00:00:00 |
| 18316 | p2 | NULL | 2007.6.1 00:00:00 |
| 29346 | p2 | NULL | 2006.12.15 00:00:00 |
| 2581 | p3 | Analyst | 2007.10.15 00:00:00 |
| 9031 | p1 | Manager | 2007.4.15 00:00:00 |
| 28559 | p1 | NULL | 2007.7.1 00:00:00 |
| 28559 | p2 | Clerk | 2007.2.1 00:00:00 |
| 9031 | p3 | Clerk | 2006.11.15 00:00:00 |
| 29346 | p1 | Clerk | 2007.1.4 00:00:00 |

Запрос в примере 7.1 задает выборку строк, т. е. он создает горизонтальное подмножество базовой таблицы works\_on. Возможно также создание представления с ограничениями на включаемые в него столбцы и строки. Создание такого представления показано в примере 7.2.

Пример 7.2. Ограничение столбцов, выбираемых в представление

USE sample;

GO

CREATE VIEW v\_without\_budget

AS SELECT project\_no, project\_name

FROM project;

Запрос в примере 7.2 выбирает для включения в представление v\_without\_budget все столбцы таблицы project, за исключением столбца budget.

В общем формате инструкции CREATE VIEW не обязательно указывать имена столбцов представления. Однако, с другой стороны, в приведенных далее двух случаях обязательно требуется явно указывать имена столбцов:

• если столбец представления создается из выражения или агрегатной функции;

• если два или больше столбцов представления имеют одинаковое имя в базовой таблице.

В примере 7.3 показано создание представления, для которого явно указываются имена столбцов.

Пример 7.3. Явное задание имен столбцов для представления

USE sample;

GO

CREATE VIEW v\_count(project\_no, count\_project)

AS SELECT project\_no, COUNT(\*)

FROM works\_on

GROUP BY project\_no;

В примере 7.3 имена столбцов представления v\_count должны быть указаны явно по той причине, что инструкция SELECT содержит агрегатную функцию COUNT(\*), которая требует, чтобы все столбцы представления были именованы.

Не требуется явно указывать список столбцов в инструкции CREATE VIEW, если применить заглавия столбцов, как это показано в примере 7.4.

Пример 7.4. Создание представления с заглавиями столбцов

USE sample;

GO

CREATE VIEW v\_count1

AS SELECT project\_no, COUNT(\*) count\_project

FROM works\_on

GROUP BY project\_no;

Представление можно создать из другого представления, как показано в примере 7.5.

Пример 7.5. Создание представления по другому представлению

USE sample;

GO

CREATE VIEW v\_project\_p2

AS SELECT emp\_no

FROM v\_clerk

WHERE project\_no ='p2';

Представление v\_project\_p2 в примере 7.5 создается из представления v\_clerk (см. пример 7.1). Все запросы, использующие представление v\_project\_p2, преобразовываются в эквивалентные запросы к базовой таблице works\_on.

Представления можно также создавать посредством среды Management Studio. Для этого выберите в обозревателе объектов базу данных, в которой требуется создать представление, щелкните в ней правой кнопкой мыши узел Views (Представления) и в открывшемся контекстном меню выберите пункт New View (Новое представление). Откроется редактор представлений, в котором можно выполнять следующие действия:

• выбрать базовые таблицы и строки в этих таблицах для создания представления;

• присвоить представлению имя и определить условия в предложении WHERE соот-ветствующего запроса.

**1.2. Изменение и удаление представлений**

Для изменения определения представления в языке Transact-SQL применяется инструкция ALTER VIEW. Синтаксис этой инструкции аналогичен синтаксису инструкции CREATE VIEW, применяющейся для создания представления.

Использование инструкции ALTER VIEW позволяет избежать переназначения существующих разрешений для представления. Кроме этого, изменение представления

посредством этой инструкции не влияет на объекты базы данных, зависящие от этого представления. Если же модифицировать представление, сначала удалив его (инструкция DROP VIEW), а затем создав новое представление с требуемыми свойствами (инструкция CREATE VIEW), то все объекты базы данных, которые ссылаются на это представление, не будут работать должным образом, по крайней мере, в период времени между удалением представления и его воссоздания.

Использование инструкции ALTER VIEW показано в примере 7.6.

Пример 7.6. Изменение представления посредством инструкции ALTER VIEW

USE sample;

GO

ALTER VIEW v\_without\_budget

AS SELECT project\_no, project\_name

FROM project

WHERE project\_no >= 'p3';

В примере 7.6 инструкция ALTER VIEW расширяет инструкцию SELECT в представлении v\_without\_budget (см. пример 7.2) новым условием в предложении WHERE.

Инструкцию ALTER VIEW можно также применять и для изменения индексированных представлений. Эта инструкция удаляет все индексы для таких представлений.

Инструкция DROP VIEW удаляет из системных таблиц определение указанного в ней представления. Применение этой инструкции показано в примере 7.7.

Пример 7.7. Удаление представления посредством инструкции DROP VIEW

USE sample;

GO

DROP VIEW v\_count;

При удалении представления инструкцией DROP VIEW все другие представления, основанные на удаленном, также удаляются, как показано в примере 7.7.

Пример 7.7. Неявное удаление представления

USE sample;

GO

DROP VIEW v\_clerk;

В примере 7.8 инструкция DROP VIEW явно удаляет представление v\_clerk, при этом неявно удаляя представление v\_project\_p2, основанное на представлении v\_clerk

(см. пример 7.5). Теперь попытка выполнить запрос по представлению v\_project\_ p2 возвратит сообщение об ошибке: "Invalid Object name (Недействительное имя: 'v\_clerk)".

**1.3. Редактирование информации о представлениях**

Наиболее важным представлением каталога применительно к представлениям является sys.objects. Как уже упоминалось, это представление каталога содержит информацию касательно всех объектов в текущей базе данных. Все строки этого представления со значением V в столбце type содержат информацию о представлениях.

А представление каталога sys.views содержит дополнительную информацию о существующих представлениях. Наиболее важным столбцом этого представления является столбец with\_check\_option, который информирует, указано или нет предложение WITH CHECK OPTION.

Запрос для определенного представления можно отобразить посредством системной процедуры sp\_helptext.

**2. Инструкции DML и представления**

Выборка информации из представлений и ее модификация осуществляются посредством тех же самых инструкций языка Transact-SQL, что и для выборки и модификации информации из базовых таблиц.

**2.1. Выборка информации из представления**

Для всех практических целей представление это то же самое, что и любая базовая таблица базы данных. Поэтому выборку информации из представления можно рассматривать, как преобразование инструкций запроса по представлению в эквивалентные операции по базовым таблицам, на основе которых создано это представление. Такой запрос на выборку данных из представления показан в примере 7.9.

Пример 7.9. Запрос на выборку данных из представления

USE sample;

GO

CREATE VIEW v\_d2

AS SELECT emp\_no, emp\_lname

FROM employee

WHERE dept\_no ='d2';

GO

SELECT emp\_lname

FROM v\_d2

WHERE emp\_lname LIKE 'J%';

Результат выполнения этого запроса:

|  |
| --- |
| **emp\_lname** |
| James |

Инструкция SELECT в примере 7.9 трансформируется в следующую эквивалентную форму с использованием таблицы из представления v\_d2:

SELECT emp\_lname

FROM employee

WHERE emp\_lname LIKE 'J%'

AND dept\_no ='d2';

В следующих трех разделах мы рассмотрим использование представлений с тремя другими инструкциями языка DML: INSERT, UPDATE и DELETE. Модифицирование данных посредством этих инструкций подобно выборке данных. Единственное от-личие состоит в том, что для представления, используемого для вставки, модифицирования и удаления данных из таблицы, на основе которой оно создано, существуют некоторые ограничения.

**2.2. Инструкция INSERT и представление**

Инструкцию INSERT можно применять с представлением, как если бы оно было обычной базовой таблицей. Вставляемые в представление строки в действительности вставляются в таблицу в основе представления.

В примере 7.10 создается представление v\_dept, которое содержит первые два столбца таблицы department. Последующая инструкция INSERT вставляет две строки в таблицу, связанную с представлением, используя значения d4 и Development. Столбцу location, который не вошел в представление v\_dept, присваивается значение NULL.

Пример 7.10. Вставка строк в представление

USE sample;

GO

CREATE VIEW v\_dept

AS SELECT dept\_no, dept\_name

FROM department;

GO

INSERT INTO v\_dept

VALUES('d4', 'Development');

При использовании представления обычно возможно вставить строку, которая не удовлетворяет условиям в предложении WHERE запроса представления. Чтобы ограничить вставку только строками, которые удовлетворяют условиям запроса, применяется предложение WITH CHECK OPTION. При использовании этого предложения компонент Database Engine проверяет каждую вставляемую строку на удовлетворение условий предложения WHERE. Если это предложение отсутствует, такая проверка не выполняется, вследствие чего каждая вставляемая в представление строка также вставляется в таблицу в его основе. Это может вызвать путаницу, когда строка вставляется в представление, но впоследствии не возвращается из этого представления инструкцией SELECT, т. к. для нее принудительно выполняются условия предложения WHERE. Предложение WITH CHECK OPTION также применяется и с инструкцией UPDATE.

В примерах 7.11—7.12 показана разница между применением и неприменением предложения WITH CHECK OPTION соответственно.

Пример 7.11. Запрос на вставку строк в представление с применением предложения WITH CHECK OPTION

USE sample;

GO

CREATE VIEW v\_2006\_check

AS SELECT emp\_no, project\_no, enter\_date FROM works\_on

WHERE enter\_date BETWEEN '01.01.2006' AND '12.31.2006'

WITH CHECK OPTION;

GO

INSERT INTO v\_2006\_check

VALUES (22334, 'p2', '1.15.2007');

В примере 7.11 система проверяет, соответствует ли вставляемое в столбец enter\_date значение True (истина) при вычислении условия в предложении WHERE инструкции SELECT. Если вставляемое значение не удовлетворяет этим условиям, строка не вставляется.

Пример 7.12. Запрос на вставку строк в представление без применения предложения WITH CHECK OPTION

USE sample;

GO

CREATE VIEW v\_2006\_nocheck

AS SELECT emp\_no, project\_no, enter\_date

FROM works\_on

WHERE enter\_date BETWEEN '01.01.2006' AND '12.31.2006';

GO

INSERT INTO v\_2006\_nocheck

VALUES (22334, 'p2', '1.15.2007');

SELECT \*

FROM v\_2006\_nocheck;

В результате выполнения этого запроса будут вставлены следующие строки:

|  |  |  |
| --- | --- | --- |
| **emp\_no** | **project\_no** | **enter\_date** |
| 10102 | p1 | 2006-10-01 |
| 29346 | p2 | 2006-12-15 |
| 9031 | p3 | 2006-11-15 |

Поскольку в примере 7.12 предложение WITH CHECK OPTION не применяется, инструкция будет выполнена, и строка вставляется в основную таблицу works\_on. Обратите внимание на тот факт, что вставленная строка не будет возвращена инструкцией SELECT, поскольку ее нельзя извлечь посредством представления v\_2006\_nocheck.

Вставку строк в таблицу, на которой основано представление, нельзя выполнить, если это представление содержит одну из следующих возможностей:

• предложение FROM в определении представления содержит более чем одну таблицу, и список столбцов содержит столбцы более чем из одной таблицы;

• столбец в представлении создается из агрегатной функции;

• инструкция SELECT в представлении содержит предложение GROUP BY или параметр DISTINCT;

• столбец в представлении создается из константы или выражения.

В примере 7.13 показано представление, которое нельзя использовать для вставки строк в таблицу, на которой основано это представление.

Пример 7.13. Представление, неприменимое для вставки строк в таблицу в своей основе

USE sample;

GO

CREATE VIEW v\_sum(sum\_of\_budget)

AS SELECT SUM(budget)

FROM project;

GO

SELECT \* FROM v\_sum;

Запрос в примере 7.13 создает представление v\_sum, которое содержит агрегатную функцию SUM() в инструкции SELECT. Поскольку представление в этом примере возвращает результат объединения нескольких строк (а не одну строку таблицы project), то нет смысла пытаться вставить одну строку в базовую таблицу, используя это представление.

**2.3. Инструкция UPDATE и представление**

Инструкцию UPDATE можно применять с представлением, как будто бы это была ба-зовая таблица. При модифицировании строк представления также модифицируется содержимое таблицы в его основе.

Запрос в примере 7.14 создает представление, посредством которого затем модифицируется таблица works\_on.

Пример 7.14. Представление, модифицирующее базовую таблицу в своей основе

USE sample;

GO

CREATE VIEW v\_p1

AS SELECT emp\_no, job

FROM works\_on

WHERE project\_no = 'p1';

GO

UPDATE v\_p1

SET job = NULL

WHERE job = 'Manager';

Операцию обновления представления v\_p1 в примере 7.14 можно рассматривать эквивалентной выполнению следующей инструкции UPDATE:

UPDATE works\_on

SET job = NULL

WHERE job = 'Manager'

AND project\_no = 'p1'

Логическое значение предложения WITH CHECK OPTION для инструкции UPDATE имеет такое же значение, как и для инструкции INSERT. Использование предложения WITH CHECK OPTION в инструкции UPDATE показано в примере 7.15.

Пример 7.15. Использование предложения WITH CHECK OPTION в инструкции UPDATE

USE sample;

GO

CREATE VIEW v\_100000

AS SELECT project\_no, budget

FROM project

WHERE budget > 100000

WITH CHECK OPTION;

GO

UPDATE v\_100000

SET budget = 93000

WHERE project\_no = 'p3';

В примере 7.15 компонент Database Engine проверяет, будет ли измененное значе-ние столбца budget давать значение True в условии предложения WHERE инструкции SELECT. Попытка изменения значения завершается неудачей, поскольку условие не удовлетворяется, т. е. вставляемое значение 93000 не больше, чем значение 100000.

Модификацию столбцов таблицы, на которой основано представление, нельзя выполнить, если это представление содержит одну из следующих возможностей:

• предложение FROM в определении представления включает более чем одну таблицу, и список столбцов содержит столбцы из более чем одной таблицы;

• столбец представления создается из агрегатной функции;

• инструкция SELECT в представлении содержит предложение GROUP BY или параметр DISTINCT;

• столбец в представлении создается из константы или выражения.

В примере 7.16 показано представление, которое нельзя использовать для изменения значений в таблице, на которой основано представление.

Пример7.16. Представление, неприменимое для изменения строк в таблице, на которой основано представление

USE sample;

GO

CREATE VIEW v\_uk\_pound (project\_number, budget\_in\_pounds)

AS SELECT project\_no, budget\*0.65

FROM project

WHERE budget > 100000;

GO

SELECT \* FROM v\_uk\_pound;

Результат выполнения этого запроса:

|  |  |
| --- | --- |
| **project\_number** | **budget\_in\_pounds** |
| p1 | 78000 |
| p3 | 121225 |

В примере 7.16 представление v\_uk\_pound нельзя использовать с инструкцией UPDATE (или с инструкцией INSERT), поскольку значения столбца budget\_in\_pound являются результатом вычисления арифметического выражения, а не первоначальными значениями столбца таблицы, на которой основано это представление.

**2.4. Инструкция DELETE и представление**

С помощью представления можно удалить строки из таблицы, на которой оно основано, как это показано в примере 7.17.

Пример 7.17. Удаление строк из таблицы посредством ее представления

USE sample;

GO

CREATE VIEW v\_project\_p1

AS SELECT emp\_no, job

FROM works\_on

WHERE project\_no = 'p1';

GO

DELETE FROM v\_project\_p1 WHERE job = 'Clerk';

Запрос в примере 7.17 создает представление, посредством которого затем удаляются строки из таблицы works\_on. Удаление строк из таблицы, на которой основано представление, невозможно, если:

• предложение FROM в определении представления содержит более чем одну таблицу, и список столбцов содержит столбцы более чем из одной таблицы;

• столбец в представлении создается из агрегатной функции;

• инструкция SELECT представления содержит предложение GROUP BY или параметр DISTINCT.

В отличие от инструкций INSERT и UPDATE, инструкция DELETE допускает значения, получаемые из констант или выражений, в столбце представления, используемого для удаления строк из таблицы, на которой оно основано.

В примере 7.18 показано представление, посредством которого можно удалять строки, но не вставлять строки или изменять значения столбцов.

Пример 7.17. Представление, позволяющее удалять, но не вставлять троки или изменять значения столбцов

USE sample;

GO

CREATE VIEW v\_budget (budget\_reduction)

AS SELECT budget\*0.9

FROM project;

GO

DELETE FROM v\_budget;

Инструкция DELETE в примере 7.18 удаляет все строки таблицы project, на которой основано представление v\_budget.

**3. Индексированные представления**

Определение представления всегда содержит запрос, играющий роль фильтра. Если представление не имеет индексов, то компонент Database Engine динамически создает результирующий набор из всех запросов, которые обращаются к представлению. (Выражение "динамически" здесь означает, что модифицированное содержимое таблицы будет всегда отображаться в соответствующем представлении.) Кроме этого, если представление содержит вычисления по одному или больше столбцов таблицы, то эти вычисления выполняются при каждом обращении к представлению.

Если инструкция SELECT представления обрабатывает большое количество строк из одной или более таблиц, динамическое создание результирующего набора запроса может понизить уровень производительности запроса. Если подобное представление часто используется в запросах, уровень производительности можно значительно повысить, создав кластеризованный индекс для этого представления. Создание кластеризованного индекса означает, что система материализует динамические данные в страницах листьев структуры индекса.

Компонент Database Engine позволяет создавать индексы для представлений. Такие представления называются индексированными или материализованными представлениями. Результирующий набор, возвращаемый представлением с кластеризованным индексом, сохраняется в базе данных таким же образом, как и таблица с кластеризованным индексом. Это означает, что узлы листьев B+-дерева кластеризованного индекса содержат страницы данных.

**3.1. Создание индексированного представления**

Индексированное представление создается в два этапа.

1. Создается представление посредством инструкции CREATE VIEW с предложением

SCHEMABINDING.

2. Создается кластеризованный индекс для этого представления.

В примере 7.19 показан первый шаг создания индексированного представления — создание представления. В этом примере предполагается, что таблица works\_on имеет очень большой размер.

Пример 7.19. Создание представления для последующего индексирования

USE sample;

GO

CREATE VIEW v\_enter\_month

WITH SCHEMABINDING

AS SELECT emp\_no, DATEPART(MONTH, enter\_date) AS enter\_month FROM dbo.works\_on;

Таблица works\_on базы данных sample содержит столбец enter\_date, который представляет дату начала работы сотрудника над соответствующим проектом. Всех сотрудников, которые начали работать над проектами в указанный месяц, можно выбрать с помощью представления, представленного в примере 7.19. Для выборки этого результирующего набора Database Engine не может использовать индекс таблицы, поскольку индекс для столбца enter\_date будет определять значения этого столбца по полной дате, а не только по месяцу. В таком случае можно воспользо-ваться индексированным представлением, создание которого показано в приме-ре 7.20.

Пример 7.20. Создание индекса для представления

USE sample;

GO

CREATE UNIQUE CLUSTERED INDEX

c\_workson\_deptno ON v\_enter\_month (enter\_month, emp\_no);

Чтобы создать представление индексированным, необходимо создать однозначный (уникальный) кластеризованный индекс для столбца (столбцов) этого представления. (Как уже упоминалось ранее, кластеризованный индекс является единственным типом индекса, который содержит значения данных в своих страницах листьев.) После создания такого индекса система баз данных выделяет память для этого представления, после чего можно создавать любое число некластеризованных индексов, поскольку теперь это представление рассматривается как (базовая) таблица.

Индексированное представление можно создать только в том случае, если оно является детерминированным, т. е. представление всегда возвращает один и тот же результирующий набор. Для этого следующим параметрам инструкции SET нужно присвоить значение ON:

QUOTED\_IDENTIFIER;

CONCAT\_NULL\_YIELDS\_NULL; ANSI\_NULLS;

ANSI\_PADDING; ANSI\_WARNINGS.

Кроме этого, параметру NUMERIC\_ROUNDABORT нужно присвоит значение OFF.

Чтобы создать индексированное представление, представление должно отвечать следующим требованиям:

• все используемые в представлении функции (как системные, так и определяе-мые пользователем) должны быть детерминированными, т. е. для одних и тех же аргументов они всегда должны возвращать один и тот же результат;

• представление должно ссылаться только на базовые таблицы;

• представление и ссылки на базовую таблицу (таблицы) должны иметь одного владельца и принадлежать к одной и той же базе данных;

• представление должно быть создано с опцией SCHEMABINDING. Эта опция связыва-ет представление со схемой, содержащей базовые таблицы, лежащие в основе представления;

• определенные пользователем функции, на которые ссылается представление, должны быть созданы с предложением SCHEMABINDING;

• инструкция SELECT в представлении не должна содержать следующие предложе-ния, параметры и прочие элементы: DISTINCT, UNION, TOP, ORDER BY, MIN, MAX, COUNT,

• OUTER, SUM (для выражений, допускающих значения NULL), подзапросы или произ-водные таблицы.

Удовлетворение всех этих требований можно проверить посредством функции свойств objectproperty с параметром свойств IsIndexable, как показано в примере 7.21. Если функция возвращает значение 1, то представление удовлетворяет всем требованиям для создания для него индекса.

Пример 7.21. Проверка удовлетворения требований для создания индекса представления

USE sample;

SELECT objectproperty(object\_id('v\_enter\_month'), 'IsIndexable');

Модифицирование структуры индексированного представления

Чтобы удалить однозначный кластеризованный индекс в индексированном представлении, необходимо также удалить все его некластеризованные индексы. После удаления кластеризованного индекса представления система рассматривает его как обычное представление.

Если вы хотите изменить обычное представление на индексированное, то для него вам нужно создать кластеризованный индекс. Чтобы сделать это, вы сначала должны указать предложение SCHEMABINDING. Представление можно удалить, а потом воссоздать, указав предложение SCHEMABINDING в инструкции CREATE SCHEMA, или же можно создать другое представление, которое имеет такой же текст, как и существующее представление, но имеет другое имя.

**Редактирование информации, связанной с индексированными представлениями**

Проверить, активирован ли какой-либо параметр инструкции, можно с помощью функции свойств sessionproperty. Если функция возвращает значение 1, то указанный параметр установлен (т. е. имеет значение ON). В примере 7.22 показано использование этой функции для проверки значения параметра

QUOTED\_IDENTIFIER.

Пример 7.22. Проверка значения параметра QUOTED\_IDENTIFIER

SELECT sessionproperty ('QUOTED\_IDENTIFIER');

Наиболее простым способом является использование динамически административного представления sys.dm\_exec\_session, поскольку оно позволяет получить значения всех параметров инструкции SET, используя только один запрос. (Опять же, если значение столбца равно 1, то соответствующий параметр активирован.) В примере 7.23 демонстрируется использование этой функции для получения значений первых четырех параметров инструкции SET, перечисленных в

разд. "Создание индексированного представления".

Пример 7.23. Определение значения параметров инструкции SET посредством динамического административного представления sys.dm\_exec\_session

USE sample;

SELECT quoted\_identifier, concat\_rmll\_yields\_null, ansi\_nulls, ansi\_padding

FROM sys.dm\_exec\_sessions

WHERE session\_id = @@spid;

Узнать, материализовано ли представление, т. е. использует ли оно дисковое пространство или нет, можно с помощью системной процедуры sp\_spaceused. Результат выполнения запроса в примере 7.24 показывает, что представление v\_enter\_month использует область памяти как для данных, так и для определенного индекса.

Пример 7.24. Запрос для проверки использования хранилища представлением

USE sample;

EXEC sp\_spaceused 'v\_enter\_month';

Этот запрос возвращает следующий результирующий набор:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **rows** | **Reserved** | **data** | **index\_size** | **unused** |
| v\_enter\_month | 11 | 16KB | 8KB | 8KB | 0KB |

**Преимущества индексированных представлений**

Кроме возможного повышения уровня производительности для сложных представлений, к которым часто обращаются запросы, применение индексированных представлений имеет два других преимущества:

• индекс представления может быть использован даже в том случае, если в представлении явно не указана ссылка на предложение FROM;

• все изменения данных отражаются в соответствующих индексированных представлениях.

Возможно, самой важной особенностью индексированных представлений является то, что у запроса в представлении нет явного указания на использование индекса

в этом представлении. Иными словами, если запрос содержит ссылку на столбцы

в базовой таблице (или таблицах), которые также существуют в индексированных представлениях, и оптимизатор запросов определит, что самым лучшим способом выполнения запроса будет использование индексированного представления, то он выбирает индексы представления таким образом, как и индексы таблиц, когда запрос не ссылается на них явно.

При создании индексированного представления его результирующий набор сохраняется на диске (одновременно с созданием индекса). Таким образом, все данные, которые изменяются в базовых таблицах, также изменяются в соответствующем результирующем наборе индексированного представления.

Кроме всех преимуществ, которые можно получить благодаря использованию индексированных представлений, имеется также и потенциальный недостаток: индексы индексированных представлений обычно более сложны в обслуживании, чем индексы базовых таблиц. Причиной этому является то, что структура однозначного

кластеризованного индекса индексированного представления более сложна, чем структура соответствующего индекса базовой таблицы.

Можно значительно повысить уровень производительности следующих далее запросов, если проиндексировать представления, к которым они обращаются:

• запросы, которые обрабатывают большое количество строк и содержат операции соединения или агрегатные функции;

• операции соединения и агрегатные функции, которые часто выполняются в од-ном или нескольких запросах.

Если запрос ссылается на обычное представление, и системе баз данных требуется обработать большое количество строк, используя операцию соединения, то оптимизатор обычно выбирает менее оптимальный метод соединения. Но если для этого представления определить кластеризованный индекс, то уровень производительности запроса можно значительно повысить, т. к. оптимизатор запросов может использовать наиболее подходящий метод. (То же самое относится и к агрегатным функциям.)

Даже если запрос, который обращается к обычному представлению, и не обрабатывает большое количество строк, все равно, в случае частого использования такого запроса, применение индексированного представления может быть очень полезным. То же самое относится и к группе запросов, которые соединяют одни и те же таблицы или используют один и тот же тип агрегатных функций.

***Задание на лабораторную работу***

**Вариант 1**

1. Создать представление, содержащее код издательства, название издательства, год издания и количество книг каждого издательства.

С помощью созданного представления, определить издательство, книги которого не

поступали в прошлом году.

1. Создать представление, содержащее код читателя, номер читательского билета, фамилию читателя, книги, находящиеся у читателя и срок их сдачи.

С помощью созданного представления  составить список читателей, у которых просрочена

сдача книг.

1. Создать представление, содержащее полную информацию о выданных книгах и взявших их читателях.

С помощью созданного представления определить читателя, имеющего на руках максимальное количество книг и сроки сдачи этих книг.

1. Создать представление, содержащее журнал выдачи книг, в котором содержится полная информация о книгах и читателях.

С помощью созданного представления определить наиболее популярную книгу и читателя прочитавшего максимальное количество книг.

**Вариант 2**

1. Создать представление, содержащее фамилию владельца, номер карточки, тип, срок действия. С помощью созданного представления отыскать фамилии владельцев и номера карточек, срок действия которых истек.
2. Создать представление, показывающее сколько карточек каждого типа выдано.

С помощью созданного представления определить какого типа карточек выдано меньше всего, какого больше всего.

1. Создать представление, показывающее номера карточек , на которые в течении 30 дней не было операций «зачисление». С помощью созданного представления отыскать владельцев этих карточек.
2. Создать представление, содержащее журнал операций, в котором содержится полная информация о карточках, их владельцах и операциях. С помощью созданного представления получить полную информацию (Фамилия, Имя, номер и тип карточки, количество операций) о владельце карточки, который совершил максимальное количество операций за месяц .

**Вариант 3**

1. Создать представление, содержащее полную информацию о штатном расписании

(Сотрудник, отдел, должность, оклад, дата приема, дата увольнения). С помощью созданного представления составить таблицу уволенных в прошлом месяце сотрудников.

1. Создать представление, содержащее информацию о должностях в каждом отделе.

С помощью созданного представления определить с самым высоким средним окладом.

1. Создать представление, содержащее информацию о принятых за последний месяц сотрудниках. С помощью созданного представления определить отдел, имеющий максимальное количество новых сотрудников.
2. Создать представление, содержащее информацию о вакантных должностях. С помощью созданного представления определить отдел с максимальным количеством вакансий.

**Вариант 4**

1. Создать представление, содержащее полную информацию о группах всех факультетов.

С помощью созданного представления определить факультет, имеющий максимальное количество групп.

1. Создать представление, содержащее полную информацию о студентах (Фамилия, имя, отчество, группа, факультет, номер и дата приказа о зачислении).

С помощью созданного представления определить количество студентов, зачисленных каждым приказом.

1. Создать представление, содержащее информацию о студентах факультета ИТР. С помощью созданного представления определить количество приказов о зачислении студентов факультета ИТР.
2. Создать представление, содержащее информацию о старостах всех факультетов. С помощью созданного представления определить старост факультета ИТР, не имеющих телефонов.

**Вариант 5**

1. Создать представление, содержащее полную информацию о журнале выездов. С помощью созданного представления определить фамилию водителя, имеющего наибольший километраж поездок.
2. Создать представление, содержащее информацию о количестве выездов по датам.

С помощью созданного представления определить даты, когда число выездов было выше среднего.

1. Создать представление, содержащее информацию о среднем километраже выездов каждого водителя за месяц. С помощью созданного представления определить фамилии водителей, имеющих наименьший километраж и наибольший средний километраж.
2. Создать представление, содержащее информацию об использовании автомобилей за прошедший месяц . С помощью созданного представления определить автомобиль, который имеет максимальное число выездов.

**Вариант 6**

1. Создать представление, содержащее полную информацию о заселенных номерах

С помощью созданного представления определить фамилию клиента, заселившегося раньше всех.

1. Создать представление, содержащее полную информацию обо всех номерах отеля.

С помощью созданного представления определить количество самых дорогих номеров.

1. Создать представление, содержащее информацию о среднем времени проживания жильцов в каждом номере. С помощью созданного представления определить номера с наименьшей средней продолжительностью проживания в них.
2. Создать представление, содержащее информацию о клиентах, проживавших в отеле в прошлом месяце. С помощью созданного представления определить выручку гостиницы за прошлый месяц.

**Вариант 7**

1. Создать представление, содержащее полную информацию о графике приема для каждого врача на завтра. С помощью созданного представления определить фамилии пациентов, записавшихся на прием к нескольким врачам.
2. Создать представление, содержащее полную информацию обо всех врачах поликлиники.

С помощью созданного представления создать список врачей-терапевтов.

1. Создать представление, содержащее информацию о среднем времени приема пациентов каждым врачом на прошедшей неделе. С помощью созданного представления определить врачей с наименьшей средней продолжительностью приема пациента.
2. Создать представление, содержащее информацию о пациентах, посетивших поликлинику на прошлой неделе, и о врачах их принимавших. С помощью созданного представления определить каких врачей посещали пациенты, которые неоднократно посещали поликлинику на прошлой неделе.

**Вариант 8**

1. Создать представление, содержащее полную информацию о каждом заказе на завтра. С помощью созданного представления определить фамилии клиентов с максимальной суммой заказов на завтра.
2. Создать представление, содержащее полную информацию обо всех товарах, заказанных на завтра. С помощью созданного представления определить суммарную выручку магазина, которую предполагается получить завтра.
3. Создать представление, содержащее информацию о средней стоимости заказов, сделанных каждым клиентом в прошлом месяце. С помощью созданного представления определить фамилию клиента с максимальной средней стоимостью заказа.
4. Создать представление, содержащее информацию о продажах товаров в прошлом месяце.

С помощью созданного представления определить группу товаров, приносящую максимальную выручку магазину.

**Вариант 9**

1. Создать представление, содержащее полную информацию о продажах за прошлый месяц.

С помощью созданного представления определить рейс, на который продано максимальное количество билетов.

1. Создать представление, содержащее полную информацию обо всех рейсах, на которые были проданы билеты на следующий месяц. С помощью созданного представления определить рейс, приносящий максимальную выручку.
2. Создать представление, содержащее информацию о среднем количестве билетов, продаваемых на рейс. С помощью созданного представления определить рейс, средняя загрузка которого минимальна.
3. Создать представление, содержащее информацию о рейсах, совершаемых в аэропорт «Минск» и из него. С помощью созданного представления определить аэропорт, полеты в который из Минска наиболее.

**Вариант 10**

1. Создать представление, содержащее полную информацию об успеваемости учеников.

С помощью созданного представления определить предмет, по которому выставлено минимальное количество оценок.

1. Создать представление, содержащее полную информацию о предметах и ведущих эти предметы преподавателях. С помощью созданного представления определить предмет, который ведут максимальное число преподавателей.
2. Создать представление, содержащее информацию о среднем балле успеваемости у каждого преподавателя. С помощью созданного представления определить фамилию преподавателя, у которого самая низкая успеваемость.
3. Создать представление, содержащее информацию об учениках, получавших оценки в прошлом месяце. С помощью созданного представления определить лучшего ученика за прошлый месяц по каждому предмету.

**Вариант 11**

1. Создать представление, содержащее полную информацию о продажах билетов на спектакли за прошлую неделю. С помощью созданного представления определить дату, время и название спектакля, на который было продано максимальное количество билетов.
2. Создать представление, содержащее полную информацию о спектаклях на следующий месяц (название спектакля, дата, время). С помощью созданного представления определить название спектакля, который запланирован чаще всех.
3. Создать представление, содержащее информацию о спектаклях и средней выручки от продажи билетов на них за прошлый месяц. С помощью созданного представления определить спектакль, дающий максимальную выручку.
4. Создать представление, содержащее информацию о рядах в зрительном зале и среднем количестве проданных мест в этих рядах за прошлый месяц. С помощью созданного представления определить наиболее популярный ряд в театре.

**Вариант 12**

1. Создать представление, содержащее полную информацию о размещении студентов в общежитии. С помощью созданного представления создать полную информацию о дисциплинах, читаемых заданной кафедрой.
2. Создать представление, содержащее информацию о комнатах, количестве мест в них и количестве размещенных в них студентов. С помощью созданного представления сформировать список комнат, в которых есть свободные места.
3. Создать представление, содержащее информацию о количестве студентов каждой группы, размещенных в общежитии. С помощью созданного представления определить группу, количество студентов которой в общежитии максимально.
4. Создать представление, содержащее информацию о комнатах, размещенных в них студентах и времени их размещения. С помощью созданного представления определить комнаты, в которой раньше всех были размещены студенты.

**Вариант 13**

1. Создать представление, содержащее полную информацию о модулях, дисциплинах и кафедрах их читающих. С помощью созданного представления определить комнат, в которых проживают студенты заданной группы.
2. Создать представление, содержащее информацию о кафедрах и количестве читаемых ими дисциплин. С помощью созданного представления сформировать список кафедр, которые читают максимальное количество дисциплин.
3. Создать представление, содержащее информацию о кафедрах и количестве читаемых ими учебных часов. С помощью созданного представления определить кафедру, число читаемых часов которой минимально.
4. Создать представление о дисциплинах, по которым проводится экзамен в соответствующем семестре. С помощью созданного представления определить семестр, число экзаменов в котором минимально.

**Вариант 14**

1. Создать представление, содержащее полную информацию о информацию обо всех объектах недвижимости. С помощью созданного представления определить город, в котором имеется самая дешевая недвижимость заданного типа.
2. Создать представление, содержащее информацию о городах и количестве объектов недвижимости в них. С помощью созданного представления сформировать список городов, в которых, которых количество объектов недвижимости минимально.
3. Создать представление, содержащее информацию о городах и количестве объектов недвижимости каждого типа в них. С помощью созданного представления определить города, в которых средняя цена на объекты какого-либо типа минимальна.
4. Создать представление, об объектах недвижимости заданного города. С помощью созданного представления район с самыми дорогими объектами недвижимости.

**Вариант 15**

1. Создать представление, содержащее полное расписание работы парикмахерской на завтра. С помощью созданного представления определить сотрудников, у которых завтра должно быть максимальное количество клиентов.
2. Создать представление, содержащее информацию о сотрудниках и ожидаемой суммарной выручке по каждому из них. С помощью созданного представления сформировать список сотрудников, у которых завтра ожидается максимальная выручка.
3. Создать представление, содержащее информацию об услугах и их количестве, запланированном на завтра. С помощью созданного представления определить перечень услуг, которые оказываются чаще всех.
4. Создать представление о клиентах и оказанных им услугах за прошлый месяц. С помощью созданного представления определить клиентов, которым были оказаны самые дорогие услуги.

***Контрольные вопросы.***

|  |  |
| --- | --- |
| 1. | Дайте определение представлению. |
| 2. | Перечислите достоинства и недостатки, свойственные представлениям. |

|  |  |
| --- | --- |
|  | Каким требованиям должно удовлетворять представление, чтобы оно могло быть обновляемым? |