Оглавление

Представления (VIEW)

Представление VIEW, также иногда называемые виртуальными таблицами, представляют собой именованные и сохраненные (в базе данных) SQL запросы. VIEW (представление) часто обладают всеми свойствами обычных таблиц: в представлении (при выполнении определенных условий) можно модифицировать данные, представление может быть объектом администрирования, на работу с представлениями могут выдаваться специальные разрешения.

Однако следует помнить, что представления только похожи на базовые таблицы и, на самом деле, имеют некоторые ограничения. Например, построение индекса на представлении невозможно, на представлении невозможно определить ограничения.

Представления позволяют:

- ограничивать число столбцов
- ограничивать число строк
- выводить дополнительные столбцы
- выводить группы строк таблицы
- упрощать вид сложных запросов
- многократное использование в разных запросах.

Одно представление может быть построено на базе другого представления.

Запрос, при помощи которого формируется VIEW, выполняется каждый раз, когда VIEW становится объектом команды SQL.

Материализованные представления

Материализованное представление — физический объект базы данных, содержащий результат выполнения запроса.

Материализованные представления позволяют многократно ускорить выполнение запросов, обращающихся к большому количеству (сотням тысяч или миллионам) записей, позволяя за секунды (и даже доли секунд) выполнять запросы к терабайтам данных. Это достигается за счет прозрачного использования заранее вычисленных итоговых данных и результатов соединений таблиц. Предварительно вычисленные итоговые данные обычно имеют очень небольшой объем по сравнению с исходными данными.

Целостность данных в материализованных представлениях поддерживается за счёт периодических синхронизаций или с использованием триггеров.

Создание VIEW

Представление создается командой CREATE VIEW

```
CREATE VIEW view_name [ (column [ ,...n ] ) ] AS
SELECT column1, column2, ...
FROM table_name
[WHERE condition] [WITH CHECK OPTIONS];
```

Пример. Создать представление «Студенты из Киева»

```
create view KIEV_STUD as
       select * from STUDENTS where CITY = 'Киев';
```

Оператор WITH CHECK OPTION

Оператор **WITH CHECK OPTION** указывает на то, что при попытке изменить данные во VIEW таким образом, что будет нарушено условие WHERE, операция будет отклонена (возникнет ошибка).

Пример: представление «Преподаватели, преподающие в киевских университетах»

```
create view KIEV_LECT as
select ID as lec_id
      ,SURNAME as lec_surname
      ,NAME as lec_name
      ,CITY as lec_city
      ,UNIV_ID as lec_univ
from LECTURERS
where UNIV_ID in (select ID from UNIVERSITIES where CITY = 'Киев')
WITH CHECK OPTION;
```

Представление может быть построено на нескольких базовых таблицах и использовать лишь некоторые из столбцов в каждой из таблиц.

```
create view V_LEC as
select
       l.id
       ,concat(l.surname, ' ', l.name) as lec_full_name
       ,u.name as lec_university
from LECTURERS l
inner join UNIVERSITIES u on l.univ_id = u.id;
```

Использование VIEW

VIEW может использоваться в SQL запросах как обычная таблица

```
select * from KIEV_LECT
select * from KIEV_LECT where lec_city = 'Киев';
select * from V_LEC;
```

Агрегирующие представления

Агрегирующие представления - это представления, которые содержит предложение GROUP BY, или основывается на других агрегирующих представлениях. Агрегирующие представления могут стать превосходным способом обрабатывать полученную информацию непрерывно.

Пример:

```
create view TOTAL_DAYS
as
       select
                 exam_date
               , count(distinct subj_id) as    subj_count
               , count(distinct student_id) as student_count
               , count(mark) as                mark_count
               , avg(mark) as                  mark_avg
               , sum(mark) as                  mark_rating
       from EXAM_MARKS
       group by
                       exam_date;
```

Использование:

```
select * from TOTAL_DAYS;
```

Модификация данных «содержащихся» в представлении

Данные могут быть «добавлены» в представление, могут быть «модифицированы или удалены» из представления. При этом следует понимать, что модифицируется не сама виртуальная таблица, а запрос перенаправляется к базовой таблице.

Однако, из-за того, что представления отображают данные базовой таблицы в преобразованном или усеченном виде на модификацию данных наложены определенные ограничения.

Пример добавления записи в представление

```
select * from STUDENTS;
select * from KIEV_STUD;

insert into KIEV_STUD
(
          id
        , surname
        , name
        , gender
        , stipend
        , course
        , city
        , birthday
        , univ_id
)
values
(
          46
        , 'Якимчук'
        , 'Валерий'
        , 'm'
        , 450
        , 2
        , 'Чернигов'
        , '19970427'
        , 3
);

select * from KIEV_STUD;
select * from STUDENTS;
```

Следующее добавление не будет выполнено потому, что мы пытаемся добавить преподавателя, который не работает в киевском университете. А представление мы создавали с опцией WITH CHECK VIEW

```
insert into KIEV_LECT
values (26, 'Шатило', 'ВП', 'Чернигов', 3);

insert into KIEV_LECT
values (26, 'Шатило', 'ВП', 'Чернигов', 4);
```

Пример модификации записи в представлении

```
update KIEV_LECT set
       lec_name = 'ВН'
where lec_id = 26;

select * from KIEV_LECT;
```

При попытке изменить университет на такой, который расположен не в Киеве, возникнет ошибка

```
update KIEV_LECT set
       lec_univ = 3
where lec_id = 26;
```

Updatable views (обновляемые представления)

Команды модификации данных в общем случае не может быть применены для изменения данных «содержащихся» в представлении. Это означает, что некоторые представления являются обновляемыми, и ссылки на них можно использовать для указания таблиц, которые должны обновляться в операторах изменения данных. То есть вы можете использовать их в таких выражениях, как UPDATE, DELETE или INSERT, чтобы обновить содержимое базовой таблицы.

Чтобы представление было обновляемым, между строками в представлении и строками в базовой таблице должна быть взаимно-однозначная связь. Также следует иметь ввиду, что некоторые конструкции языка SQL делают представление недоступным для обновления. Более точно: представление не может быть обновляемым, если оно содержит любую из следующих конструкций:

- Агрегированные функции (SUM (), MIN (), MAX (), COUNT () и т. Д.)
- DISTINCT
- GROUP BY
- HAVING
- UNION или UNION ALL
- Подзапрос в списке выбора
- Некоторые соединения
- Ссылку на не обновляемое представление в предложении FROM
- Подзапрос в предложении WHERE, который ссылается на таблицу в предложении FROM
- Ссылается только на литералы (в этом случае нет базовой таблицы для обновления)
- Несколько ссылок на любой столбец базовой таблицы (ошибка для INSERT, хорошо для UPDATE, DELETE)

Однако существуют представления, которые допускают модификацию (вставку, обновление и удаление) информации, содержащейся в базовой таблице, на которой построено представление.

Когда можно обновлять представление?

1. В представлении должен быть PRIMARY KEY таблицы, на основе которой было создано представление.
2. Для INSERT, представление должно содержать любые поля основной таблицы, которые имеют ограничение NOT NULL, если другое ограничение по умолчанию, не определено.

Multiple-table updatable views

Иногда представление, составленное из нескольких таблиц, может быть обновляемым. Чтобы это работало, представление должно использовать внутреннее соединение (не внешнее соединение или UNION). Кроме того, только одна таблица в определении представления может быть обновлена, поэтому предложение SET должно содержать имена только столбцов одной из таблиц в представлении. Представления, использующие UNION ALL, недопустимы, даже если они могут быть теоретически обновляемыми.

Что касается возможности вставки (обновляемости с помощью операторов INSERT), представление является обновляемым (иногда говорят, что представление является insertable), если оно также удовлетворяет следующим дополнительным требованиям для столбцов представления:

- Не должно быть повторяющихся имен столбцов представления.

- Представление должно содержать все столбцы в базовой таблице, которые не имеют значения по умолчанию.
- Столбцы представления должны быть простыми ссылками на столбцы. Они не должны быть выражениями или составными выражениями, такими как эти:
    - 3,14159
    - col1 + 3
    - UPPER (col2)
    - col3 / col4
    - (Подзапрос)

Пример. Создадим представление, которое будет содержать информацию о студентах, проживающих в Киеве и полное название университета:

```
create view V_KIEVSTUDUNIV as
select
          s.id
        , surname
        , s.name
        , gender
        , stipend
        , course
        , s.city
        , birthday
        , u.name as uname
from STUDENTS as s
       inner join UNIVERSITIES as u on s.univ_id = u.id
where s.city = 'Киев'
with check option;
```

Попытаемся студента Козьменко Игнат (id = 4), который учится на первом курсе ХАИ «перевести» на второй курс КПИ:

```
begin tran
update V_KIEVSTUDUNIV set
       course = 2
       ,uname = 'КПИ'
where id = 4;
rollback
```

В результате попытки изменить данные мы получим сообщение об ошибке (для MS SQL Server):

```
View or function 'V_KIEVSTUDUNIV' is not updatable because the modification affects multiple
base tables.
```

Теперь попытаемся просто «перевести» этого студента в КПИ. Для начала сделаем две контрольных выборки:

```
select * from V_KIEVSTUDUNIV;
select * from UNIVERSITIES;
```

id	surname	name	gender	stipend	course	city	birthday	uname
2	Павленко	Игорь	m	600.00	1	Киев	1993-06-21	ЛГУ
4	Козьменко	Игнат	m	500.00	1	Киев	1994-04-26	ХАИ
6	Березовский	Роман	m	750.00	2	Киев	1992-03-09	КПИ
9	Печальнова	Алина	f	500.00	2	Киев	1993-02-25	ХАИ
10	Денисенко	Марк	m	500.00	3	Киев	1993-06-07	КПИ
12	Чайка	Ольга	f	550.00	5	Киев	1990-12-30	ЛГУ
15	Тарара	Леонид	m	900.00	4	Киев	1991-05-04	ХАИ
19	Бородюк	Никита	m	650.00	2	Киев	1990-08-30	ХАИ
20	Миланивская	Ольга	f	750.00	5	Киев	1989-10-22	ЛГУ
30	Забила	Алексей	m	450.00	3	Киев	1991-06-21	ЗДИА
33	Шелест	Юрий	m	450.00	2	Киев	1992-11-08	КМА

Теперь пошагово выполняем следующие инструкции:

```
begin tran -- открываем транзакцию, что бы не испортить данные

	-- выполняем "перевод" студента
	update V_KIEVSTUDUNIV set
		uname = 'КПИ'
	where id = 4;

	-- смотрим результаты
	select * from V_KIEVSTUDUNIV;
	select * from UNIVERSITIES;

rollback -- откатываем транзакцию, восстанавливаем данные
```

В результате модификации данных, произошло «неожиданное» изменение данных, которые касаются других студентов.

id	surname	name	gender	stipend	course	city	birthday	uname
2	Павленко	Игорь	m	600.00	1	Киев	1993-06-21	ЛГУ
4	**Козьменко**	**Игнат**	**m**	**500.00**	**1**	**Киев**	**1994-04-26**	**КПИ**
6	Березовский	Роман	m	750.00	2	Киев	1992-03-09	КПИ
9	Печальнова	Алина	f	500.00	2	Киев	1993-02-25	КПИ
10	Денисенко	Марк	m	500.00	3	Киев	1993-06-07	КПИ
12	Чайка	Ольга	f	550.00	5	Киев	1990-12-30	ЛГУ
15	**Тарара**	**Леонид**	**m**	**900.00**	**4**	**Киев**	**1991-05-04**	**КПИ**
19	**Бородюк**	**Никита**	**m**	**650.00**	**2**	**Киев**	**1990-08-30**	**КПИ**
20	Миланивская	Ольга	f	750.00	5	Киев	1989-10-22	ЛГУ
30	Забила	Алексей	m	450.00	3	Киев	1991-06-21	ЗДИА
33	Шелест	Юрий	m	450.00	2	Киев	1992-11-08	КМА

Это произошло потому, что мы изменили данные в базовой таблице

ID	NAME	RATING	CITY
1	КПИ	1257	Киев
2	КНУ	608	Киев
3	ЛПУ	593	Львов
4	КМА	588	Киев
5	ЛГУ	556	Львов
6	КПИ	534	Харьков
7	ДПИ	529	Днепр
8	ДНТУ	501	Донецк
9	ХНАДУ	500	Харьков
10	ОНПУ	496	Одесса
11	КНУСА	483	Киев
12	ТНТУ	441	Тернополь
13	ЗДИА	427	Запорожье
14	БНАУ	399	Белая Церковь
15	ХСХА	45	Херсон

Преимущества представлений

- **Безопасность.** Каждому пользователю можно разрешить доступ к не большому числу представлений, содержащих только ту информацию, которую ему позволено знать. Таким образом, можно осуществить ограничение доступа пользователей к хранимой информации.
- **Простота запросов**. С помощью представления можно извлечь данные из нескольких таблиц и представить их как одну таблицу, превращая тем самым запрос ко многим таблицам в однотабличный запрос к представлению.
- **Структурная простота**. С помощью представлений для каждого пользователя можно создать собственную структуру базы данных, определив ее как множество доступных пользователю виртуальных таблиц.

- **Защита от изменений**. Представление может возвращать непротиворечивый и неизменный образ структуры базы данных, даже если исходные таблицы разделяются, реструктуризуются или переименовываются. Однако представление должно быть обновлено, когда переименовываются лежащие в его основе таблицы или столбцы.
- **Целостность данных**. Если доступ к данным или ввод данных осуществляется с помощью представления, СУБД может автоматически проверять, выполняются ли определенные условия целостности.

Недостатки представлений

- **Производительность.** Представление создает лишь видимость существования соответствующей таблицы. На самом деле при обращении к представлению выполняется запрос, на котором представление построено. Сложность запроса скрывается в представлении, так что пользователи не представляют, какой объем работы может вызвать даже кажущийся простым запрос.
- **Управляемость**. Представления, как и все прочие объекты баз данных, должны быть управляемы. Если разработчики и пользователи баз данных смогут бесконтрольно создавать представления, то работа администратора базы данных станет существенно сложнее. Это в особенности справедливо в том случае, когда создаются представления, в основе которых лежат другие представления, которые, в свою очередь, могут быть основаны на других представлениях. Чем больше уровней между базовыми таблицами и представлениями, тем сложнее решать проблемы с представлениями, которые могут возникнуть в такой системе.
- **Ограничения на обновление**. Когда пользователь пытается обновить строки представления, СУБД должна преобразовать запрос в запрос на обновление строк исходных таблиц. Это возможно для простых представлений; более сложные представления обновлять нельзя, они доступны только для выборки. Указанные недостатки означают, что не стоит без разбора применять представления, в особенности многоуровневые, и использовать их вместо исходных таблиц. В каждом конкретном случае необходимо учитывать перечисленные выше преимущества и недостатки представлений.

Изменение представлений

Изменение представления выполняется командой

```
ALTER VIEW view_name AS SELECT …
```

Для MySQL используется команда

```
CREATE OR REPLACE VIEW view_name AS SELECT …
```

Удаление представлений

Представления удаляются командой

```
DROP VIEW view_name;
```

Пример:

```
drop view V_LEC;
```

INFORMATION_SCHEMA

В реляционных базах данных информационная схема (information_schema) представляет собой стандартный ANSI-набор представлений, доступных только для чтения, которые предоставляют информацию обо всех таблицах, представлениях, столбцах и процедурах в базе данных. [1] Он может использоваться в качестве источника информации, которую некоторые базы данных делают доступной.

СУБД, которые поддерживают информационную схему:

- Microsoft SQL Server
- MySQL
- PostgreSQL
- SQLite
- InterSystems Caché
- H2 Database
- HSQLDB
- MariaDB

Получение «сырой» информации из системных таблиц:

```
select * from sys.objects;
select * from sys.columns;

select
       o.name
       , c.name
       , t.name
       , c.max_length
       , c.precision
       , c.scale
       , c.is_nullable
       , c.is_identity
from sys.columns as c
       inner join sys.objects as o on c.object_id = o.object_id
       inner join sys.types as t on c.system_type_id = t.system_type_id
where c.object_id in (select object_id from sys.tables)
```

Получение информации из INFORMATION_SCHEMA:

```
select * from INFORMATION_SCHEMA.COLUMNS;
```