Управление данными

Лекция 8

Операторы подзапроса и представления

Свечников С.В.

Содержание занятия

01. Операторы подзапроса
02. Представления
03. Изменение данных через представления

01. Операторы подзапроса

EXISTS

Результатом действия оператора EXISTS является логическое значение: True или False.

Он принимает подзапрос в качестве аргумента и выдает значение True при наличии в подзапросе выходных данных и False в противном случае.

Пример

```
SELECT fname FROM person
WHERE EXISTS
(SELECT gid FROM gruppa
WHERE name='ИТС-1-10');
```

EXISTS и логические операторы

Оператор EXISTS можно использовать с логическими операторами: AND, OR, NOT.

Чаще всего используется оператор NOT.

Пример

```
SELECT fname FROM person
WHERE NOT EXISTS
(SELECT gid FROM gruppa
WHERE name='КМ-1-10');
```

EXISTS и агрегаты

EXISTS можно использовать с агрегатными функциями.

Пример

```
SELECT fname FROM person
WHERE EXISTS
(SELECT count(*) FROM gruppa);
```

Но проще не использовать агрегаты.

Пример

```
SELECT fname FROM person
WHERE EXISTS
(SELECT * FROM gruppa);
```

ANY и SOME

Существует еще два оператора для работы с подзапросами: ANY и SOME. Они взаимозаменяемы.

Оператор ANY (SOME) принимает все значения, выводимые подзапросом и обращается в True, если хотя бы одно из них равно значению в текущей строке внешнего запроса.

Пример

```
SELECT lname, fname FROM person
WHERE group_id = ANY
(SELECT gid FROM gruppa);
```

ALL

При использовании оператора ALL, предикат внешнего запроса обращается в True, когда каждое из значений, выбранных подзапросом, удовлетворяет условию предиката.

Пример

```
SELECT lname, fname FROM person
WHERE group_id > ALL
(SELECT gid FROM gruppa WHERE gid<4);
```

02. Представления

Представление

Представление – это объект данных, который не содержит собственных данных – разновидность таблицы, содержимое которой заимствуется из других таблиц в результате выполнения запроса.

Таблицы, с которыми мы имели дело до сих пор называются постоянными базовыми таблицами. Эти таблицы содержат информацию постоянно хранящуюся в базе данных.

Представления – это виртуальные таблицы.

VIEW

Представления определяются с помощью оператора CREATE VIEW.

Пример

```
CREATE VIEW moscowperson
AS SELECT pid, lname, fname FROM person
WHERE city='Москва';

SELECT * FROM moscowperson;
```

Преимущество представления по сравнению с базовой таблицей состоит в том, что оно автоматически обновляется при любых изменениях в таблице.

Представления можно создавать на основе любой таблицы и даже другого представления.

Обновление представлений

Содержимое представления можно модифицировать с помощью операторов DML. Изменения не влияют на само представление, а передаются в основную таблицу.

Пример

```
UPDATE moscowperson
SET lname='Орлов'
WHERE pid=1;
```

Выполнится

Выполнение этой команды даст такой же результат, как если бы мы обратились к самой таблице person.

Пример

```
UPDATE moscowperson
SET phone='5043654'
WHERE pid=1;
```

Не выполнится

Именование столбцов

Иногда бывает нужно дать столбцам новые имена, отличные от базовой таблицы, например, когда:

- некоторые из столбцов в предложении SELECT основного запроса являются выражениями или производными величинами
- запрос в представлении обращается ко многим основным таблицам, имеющим два и более столбца с одинаковыми именами

Пример

```
CREATE VIEW groupperson
AS
SELECT pid, lname, fname, year*12 AS age, name
FROM person INNER JOIN gruppa
ON (group_id=gid);
```

Необязательный аргумент AS позволяет переименовывать столбцы представления.
Типы данных берутся из столбцов основного запроса.

Групповые представления

Групповые представления – это представления, содержащие предложение GROUP BY или основанные на других групповых представлениях.

Пример

```
CREATE VIEW agregat (acity, quantity, average, maximum, minimum, amount)
AS SELECT city, COUNT(pid), AVG(year), MAX(year), MIN(year), SUM(year)
FROM person
GROUP BY city;

SELECT * FROM agregat;
```

Теперь можно создавать запросы к представлению.

Пример

```
SELECT * FROM agregat
where quantity>1 OR average<40;
```

Представления и соединения

Представления необязательно создавать на основе единственной базовой таблицы. Они могут извлекать информацию из любого числа базовых таблиц и представлений.

Примеры

```
CREATE VIEW alltables
AS SELECT a.pid, a.lname, a.fname, b.name, c.title, c.lector
FROM person a, gruppa b, course c
WHERE a.group_id=b.gid
AND c.group_id=b.gid;

CREATE VIEW viewtable
AS SELECT a.lname, a.fname, a.name
FROM alltables a, person b
WHERE a.pid=b.pid
AND a.title='Управление данными';
```

Представления и подзапросы

Представления можно использовать с подзапросами.

Пример

```
CREATE VIEW subselect
AS SELECT * FROM person
WHERE group_id =
(SELECT gid FROM gruppa
WHERE name='ИТС-1-10');

SELECT * FROM subselect;
```

Удаление представлений

Представления удаляются с помощью команды DROP VIEW.

Примеры

```
DROP VIEW agregat;
DROP VIEW groupperson;
DROP VIEW moscowperson;
DROP VIEW alltables;
DROP VIEW viewtable;
DROP VIEW subselect;
```

03. Изменение данных через представления

DML и представления

Над представлениями можно использовать операторы обновления:

- INSERT
- UPDATE
- DELETE

При обновлении представлений изменения передаются в базовую таблицу, на которой основано представление.

Принцип обновляемости VIEW

Если к представлению можно применить оператор обновления, то она называется **обновляемым**, в противном случае представление является **только для чтения**.

Возможность обновления представления определяется основным запросом. Запрос должен удовлетворять требованиям:

- должен выполняться только над одной основной таблицей
- не должен содержать агрегатных функций
- не должен содержать атрибута DISTINCT
- не должен использовать GROUP BY или HAVING
- не должен содержать констант, строк или выражений

Обновляемое представление

Представленное в примере представление является **обновляемым**, т.к. удовлетворяет требованиям обновляемости.

Пример

```
CREATE VIEW its110
AS SELECT pid, lname, fname, year FROM person
WHERE group_id=1;

SELECT * FROM its110;

UPDATE its110
SET year=year*12;
```

Представление только для чтения

Следующий пример представления является представлением **только для чтения** из-за наличия агрегатной функции.

Пример

```
CREATE VIEW quantitycity (city, quantity)
AS SELECT city, COUNT(pid)
FROM person
GROUP BY city;

SELECT * FROM quantitycity;

UPDATE quantitycity
SET quantity=10
WHERE city='Москва';
```

Поглощение значений

При обновлении представлений может возникнуть ситуация, при которой значения будут «поглощены» основной таблицей.

Пример

```
CREATE VIEW undo30
AS SELECT pid, lname, year FROM person WHERE year<30;

SELECT * FROM undo30;
```

При добавлении строки с year>30, увидеть ее в представлении будет невозможно.

Пример

```
INSERT INTO undo30
VALUES(11, 'Сковородов', 31);
```

WITH CHECK OPTION

Поглощения можно предотвратить, включив в определение представления атрибут WITH CHECK OPTION.

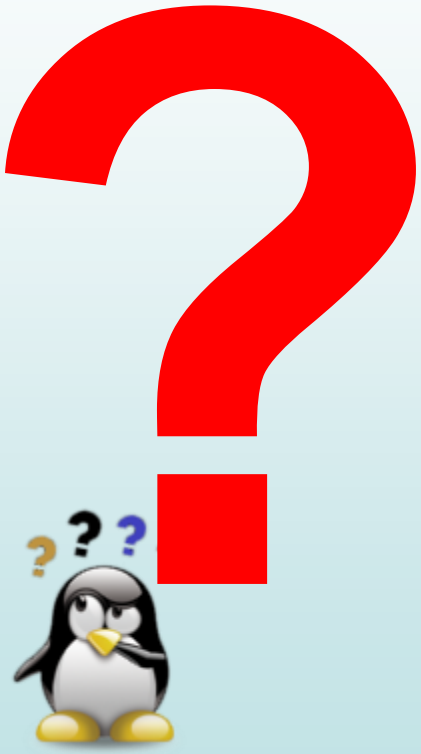
Пример

```
CREATE VIEW undo30
AS SELECT pid, lname, year FROM person WHERE year<30
WITH CHECK OPTION;

SELECT * FROM undo30;

INSERT INTO undo30
VALUES(12, 'Сковородов', 32);

INSERT INTO undo30
VALUES(13, 'Сковородов', 29);
```

Не добавится

Добавится

Вопросы



Домашнее задание

- Изучить главу 12, 14, 15 книги М.Грабера Введение в SQL
- Подготовиться к контрольной работе по лекции
- Выполнить лабораторную работу №4

Контроль

Для выполнения контрольного теста используем ссылку:

app.startexam.com/Center/Web/student

Короткая ссылка:

clck.ru/2Undm

Пароль на тест 6: **6385**

Пароль на тест 7: **4865**