Именованные запросы

MySQL

Псевдонимы при соединении таблиц

Вывести пары фамилий студентов, носящих одно имя.

```
SELECT FIRST.SURNAME, SECOND.SURNAME
FROM STUDENT FIRST, STUDENT SECOND
WHERE FIRST.NAME = SECOND.NAME;
```

```
SELECT FIRST.SURNAME, SECOND.SURNAME
FROM STUDENT FIRST, STUDENT SECOND
WHERE FIRST.NAME = SECOND.NAME
AND FIRST.SURNAME < SECOND.SURNAME;
```

В чем разница?

Именованные запросы (представления)

- Позволяют:
  - Ограничивать число столбцов, из которых пользователь выбирает или в которые вводит данные
  - Ограничивать число строк, из которых пользователь выбирает или в которые вводит данные
  - Выводить дополнительные столбцы, преобразованные из других столбцов базовой таблицы
  - Выводить группы строк таблицы

CREATE VIEW
(СОЗДАТЬ ПРЕДСТАВЛЕНИЕ)

```
CREATE VIEW TULA_STUD AS
SELECT *
FROM STUDENT
WHERE CITY = "Тула";
```

Извлечь данные

```
SELECT *
FROM TULA_STUD;
```

Представления таблиц

- В простейшем представлении таблиц выбираются *все строки и столбцы* базовой таблицы.

```
CREATE OR REPLACE
VIEW NEW_STUD_TAB AS
SELECT *
FROM STUDENT;
```

Заменить, если уже существует

Представления столбцов

- В простейшем виде представление столбцов выбирает все строки и столбцы, подобно представлению таблиц; кроме того, в качестве имен столбцов применяются псевдонимы. Например:

```
CREATE OR  REPLACE
VIEW NEW_STUDENT
(NEW_STUDENT_ID, NEW_SURNAME, NEW_NAME, NEW_STIPEND,
NEW_KURS, NEW_CITY, NEW_BIRTHDAY, NEW_UNIV_ID)
AS SELECT STUDENT_ID, SURNAME, NAME, STIPEND,
KURS, CITY, BIRTHDAY, UNIV_ID
FROM STUDENT;
```

Представления , маскирующие столбцы

- Данный вид представлений ограничивает число столбцов базовой таблицы, к которым возможен доступ . Например, представление

```
CREATE VIEW STUD AS
SELECT STUDENT_ID, SURNAME, CITY
FROM STUDENT;
```

- дает доступ пользователю к полям STUDENT_ID, SURNAME, CITY базовой таблицы STUDENT

Представления , маскирующие строки

```
CREATE VIEW TULA_STUD AS
SELECT *
FROM STUDENT
WHERE CITY = “Тула”;
```

показывает пользователю только те строки таблицы STUDENT, для которых значение поля CITY равно «Тула».

Агрегированные представления

- Предположим, необходимо ежедневно следить за количеством студентов , сдающих экзамены, количеством сданных экзаменов , количеством сданных предметов , средним баллом по каждому предмету:

```
CREATE VIEW TOTALDAY AS
SELECT EXAM_DATE, COUNT(DISTINCT SUBJ_ID) AS SUBJ_CNT,
COUNT(STUDENT_ID) AS STUD_CNT,
COUNT(MARK) AS MARK_CNT,
AVG(MARK) AS MARK_AVG, SUM(MARK) AS MARK_SUM
FROM EXAM_MARKS
GROUP BY EXAM_DATE;
```

Представления , основанные на нескольких таблицах

- представление, объединяющее несколько таблиц, может использоваться как промежуточный макет при формировании сложных отчетов, скрывающий детали объединения большого количества исходных таблиц.
- предварительно объединенные поисковые и базовые таблицы обеспечивают наилучшие условия для транзакций, позволяют использовать компактные схемы кодов , устраняя необходимость написания для каждого отчета длинных объединяющих процедур.
- позволяет использовать при формировании отчетов более надежный модульный подход.
- предварительно объединенные и проверенные представления уменьшают вероятность ошибок, связанных с неполным выполнением условий объединения

Представления, основанные на нескольких таблицах

- Можно, например, создать представление, которое показывает имена и названия сданных предметов для каждого студента:

```
CREATE VIEW STUD_SUBJ AS
SELECT A.STUDENT_ID, C.SUBJ_ID, A.SURNAME, C.SUBJ_NAME
FROM STUDENT A, EXAM_MARKS B, SUBJECT C
WHERE A.STUDENT_ID = B.STUDENT_ID
AND B.SUBJ_ID = C.SUBJ_ID;
```

- Теперь все предметы студента или всех студентов для каждого предмета можно выбрать с помощью простого запроса. Например, чтобы увидеть все предметы, сданные студентом Ивановым, подается запрос:

```
SELECT SUBJ_NAME
FROM STUD_SUBJ
WHERE SURNAME = "Иванов";
```

Представления и подзапросы

- Предположим, предусматривается премия для тех студентов, которые имеют самый высокий балл на любую заданную дату. Получить такую информацию можно с помощью представления:

```
CREATE VIEW ELITE_STUD
AS SELECT B.EXAM_DATE, A.STUDENT_ID, A.SURNAME
FROM STUDENT A, EXAM_MARKS B
WHERE A.STUDENT_ID = B.STUDENT_ID
AND B.MARK =
( SELECT MAX(MARK)
FROM EXAM_MARKS C
WHERE C.EXAM_DATE = B.EXAM_DATE);
```

Представления и подзапросы

- Если, с другой стороны, премия будет назначаться только студенту, который имел самый высокий балл и не меньше 10-ти раз , то необходимо использовать другое представление, основанное на первом:

```
CREATE VIEW BONUS
AS SELECT DISTINCT STUDENT_ID, SURNAME
FROM ELITE_STUD A
WHERE 10 <=
( SELECT COUNT(*)
FROM ELITE_STUD B
WHERE A.STUDENT_ID = B.STUDENT_ID);
```

- Извлечение из этой таблицы записей о студентах, которые будут получать премию, выполняется простым запросом:

```
SELECT * FROM BONUS;
```