1.1.1. Команды определения данных

Команда изменения таблиц ALTER TABLE

Команда используется для модификации структуры существующей таблицы. Для модификации данных используется команда UPDATE. С помощью ALTER TABLE можно:

- добавить новый столбец к таблице
- удалить столбец из таблицы
- удалить ограничение целостности со столбца или с таблицы целиком
- изменить имя столбца, его тип, и позицию столбца в таблице

Перед тем как использовать ALTER TABLE

- сохраните данные таблицы
- удалите все ограничения целостности со столбца

Сохранение данных это процесс, состоящий из пяти шагов. Например, один из столбцов таблицы имеет тип CHAR(3), его надо поменять на CHAR(5). Для этого надо:

1. Создать временный столбец, определив его так же, как существующий.
2. Переписать данные из существующего столбца во временный
3. Модифицировать временный столбец
4. Переписать данные из временного столбца обратно
5. Уничтожить временный столбец

1. Добавляем новый столбец

```
ALTER TABLE employee ADD temp_no char(3);
```

2. Переписываем

```
UPDATE employee SET temp_no = office_no;
```

3. Модифицируем

```
ALTER TABLE ALTER temp_no TYPE char(4);
```

4. Переписываем обратно

```
UPDATE employee SET office_no = temp_no;
```

5. Удаляем

```
ALTER TABLE DROP temp_no;
```

Удаление столбца

ALTER TABLE <имя таблицы> DROP <имя столбца>[, <имя столбца>...];

Например:

ALTER TABLE employee

DROP emp_no,

DROP full_name;

Команда ALTER TABLE не сможет выполниться при следующих условиях:

- если данные в таблице после удаления нарушают ограничения PRIMARY KEY или UNIQUE
- если столбец является частью первичного ключа, ограничения UNIQUE, или столбец является внешним ключом
- столбец используется в ограничении CHECK
- столбец используется в представлении (view), в триггере, или используется при вычислении другого поля.

В случае, если что-либо мешает удалить столбец, надо сначала удалить это «что-то», а затем удалять столбец. Отсюда мораль – таблицу надо создавать один раз и на всю жизнь. Семь раз мерить, потом резать. Кажется, что легче уничтожить таблицу целиком и создать ее заново, чем что-либо в ней изменить. Уничтожать таблицу жалко, если в ней уже есть данные, и невозможно, если таблица связана с другими таблицами (сначала надо удалить связи). Вывод: к созданию таблицы надо подходить очень ответственно, так как любые ошибки исправляются тяжело.

Добавление столбца

ALTER TABLE <имя таблицы> ADD <определение столбца>

Например:

ALTER TABLE employee ADD emp_no INTEGER NOT NULL;

Добавление ограничения целостности на таблицу

ALTER TABLE <имя таблицы> ADD [CONSTRAINT <имя ограничения>] <определение ограничения>;

Можно добавить ограничения PRIMARY KEY, FOREIGN KEY, UNIQUE, или CHECK.

Например:

ALTER TABLE employee ADD CONSTRAINT dept_no UNIQUE (phone_ext);

Удаление ограничения целостности, наложенного на столбец

ALTER TABLE <имя таблицы>

DROP CONSTRAINT <имя ограничения>;

Например:

ALTER TABLE project

DROP CONSTRAINT team_constrt;

Если имя ограничения не было указано явно при создании таблицы, сервер генерирует имя автоматически. Посмотреть имена ограничений можно в системной таблице RDB$RELATION_CONSTRAINTS:

SELECT * FROM RDB$RELATION_CONSTRAINTS

Модификация столбца в таблице:

Можно изменить имя столбца, тип столбца, позицию столбца в таблице.

ALTER TABLE <имя таблицы> ALTER [COLUMN] <имя столбца> <определение изменения>

<определение изменения>:

- для изменения имени: TO <новое имя>
- для изменения типа данных: TYPE <новый тип>
- для изменения позиции: POSITION <целое число>

Примеры:

1. Изменение позиции

ALTER TABLE EMPLOYEE ALTER EMP_NO POSITION 2;

2. Изменение имени EMP_NO на EMP_NUM

ALTER TABLE EMPLOYEE ALTER EMP_NO TO EMP_NUM;

3. Изменение типа

ALTER TABLE EMPLOYEE ALTER EMP_NUM TYPE CHAR(20);

Преобразование из не-символьных в символьные типы возможны при следующих ограничениях.

- BLOB и массивы не преобразуются.
- Новое определение поля должно вмещать старые данные (CHAR(3) не вместит CHAR(4))

Для удаления объектов из БД используются команды:

– удаления пустой таблицы **DROP TABLE <имя таблицы>**;

– удаление представления **DROP VIEW <имя представления>;**
– удаление индекса **DROP INDEX <имя индекса>**.

Инструкцию DROP TABLE нельзя использовать для удаления таблицы, на которую ссылается ограничение FOREIGN KEY. Сначала следует удалить ссылающееся ограничение FOREIGN KEY или ссылающуюся таблицу. Если и ссылающаяся таблица, и таблица, содержащая первичный ключ, удаляются с помощью одной инструкции DROP TABLE, ссылающаяся таблица должна быть первой в списке.

Несколько таблиц можно удалить из любой базы данных. Если удаляемая таблица ссылается на первичный ключ другой таблицы, которая также удаляется, ссылающаяся таблица с внешним ключом должна стоять в списке перед таблицей, содержащей указанный первичный ключ.

При удалении таблицы относящиеся к ней правила и значения по умолчанию теряют привязку, а любые связанные с таблицей ограничения или триггеры автоматически удаляются. Если таблица будет создана заново, нужно будет заново привязать все правила и значения по умолчанию, заново создать триггеры и добавить необходимые ограничения.

Допустим создана копия тС с именем тС2.
3) DROB TABLE тС2.

1.1.2. Команды манипулирования данными

В состав группы команд SQL, которые предназначены непосредственно для манипулирования реляционными данными и таблицами, входят операторы (команды):
– вставки записей;
– удаления записей;
– модификации данных.

Синтаксис **оператора вставки** имеет вид
INSERT INTO <имя_таблицы>
[(<список полей>)]
{VALUES (<список значений>)}
[WHERE <спецификация выбора записей>]
где <список значений> ::=<значение> [{,<значение>...}],
<значение> ::= <спецификация значения> | NULL
Если <список полей> не указывается, то в поля новой записи значения вносятся в том порядке, в котором эти поля созданы. При этом те поля, значения которых пропущены, принимают значения по умолчанию или NULL.

Первая форма оператора INSERT – *без перечисления списка полей*

(применяется, когда вставляется одна строка, данные вставляются во все поля, значения полей перечислены в порядке определения полей в таблице).

Ввести строку в таблицу S (полные данные о поставщике)

INSERT INTO S VALUES ('s1', 'Smith', 20, 'London');

Возможные ошибки:

1)Поставщик S1 уже присутствует в таблице – нарушение ограничения первичного ключа!

2) Несоответствие типов значений типам соответствующих полей таблицы.

Ввести строку в таблицу P (полные данные о детали)

INSERT INTO P VALUES ('p1', 'nut', 20,'red', 12, 'London', 20.00);

Возможные ошибки – те же.

Ввести строку в таблицу SP (полные данные о поставке)

INSERT INTO SP
VALUES
('s1', 'p1', '2010-01-21', 300);

Возможные ошибки – те же и нарушение ограничения внешнего ключа: попытка вставить в таблицу код поставщика, которого нет в таблице S и (или) код детали, которого нет в таблице P.

Вторая форма оператора INSERT – ***с перечислением списка полей*** *(применяется, когда вставляется одна строка, но значения полей перечислены в порядке, отличном от порядка определения полей в таблице и (или) данные вставляются не во все поля строки).*

Ввести строку в таблицу SP (полные данные о поставке)

INSERT INTO SP (QTY, DATE_SP, SC, PC) VALUES (300, '2010-01-21', 's1', 'p1');

Возможные ошибки – те же.

Ввести неполную строку в таблицу S (вводятся только значения двух полей, остальные поля остаются пустыми).

INSERT INTO S (s, sname) VALUES ('s6', 'Wils');

Возможные ошибки – те же и нарушение ограничения NOT NULL в полях, не указанных в данном операторе, но присутствующих в таблице S.

Третья форма оператора INSERT – оператор множественной вставки строк

1) Создание временной таблицы temp1 (имеет ту же структуру, что и таблица S)

```
Create table temp1(
   sc          varchar(5) not null primary key,
   sname       varchar(20),
```

```
    status      integer,
    city        varchar(15));
```

2) Вставить в таблицу temp1 полные данные обо всех поставщиках из Лондона

```
INSERT INTO temp1
SELECT *
FROM   S
WHERE city='London';
```

Для обновления значений наиболее часто применяется поисковый оператор модификации, который имеет следующий формат

```
UPDATE <имя_таблицы>
SET <установка> [{,<установка>}...]
[WHERE <спецификация выбора записей>]
```

где <установка>представляет собой выражение вида
<имя_поля> ={ <значение> | NULL }

1)Всем поставщикам из Лондона изменить статус на 30

```
UPDATE   S   SET   status = 30
WHERE city='London';
```

При выполнении данного запроса ограничения ссылочной целостности не проверяются, так как информация в поле связи (SC) не изменяется

2)Увеличить цену всех деталей в 1.5 раза

```
UPDATE   P  SET   cost = cost*1.5;
```

При выполнении данного запроса ограничения ссылочной целостности не проверяются, так как информация в поле связи (PC) не изменяется

3)Уменьшить на 10 количество деталей во всех поставках поставщика S1, сделанных 21.01.2010 г.

```
UPDATE   SP  SET   qty = qty-10
WHERE  SC='S1' and
            date_sp='2010-01-21';
```

При выполнении данного запроса ограничения ссылочной целостности не проверяются, так как информация в полях связи (SC, PC) не изменяется

4)Изменить код детали S1 на SS1

```
UPDATE   S  SET   SC = 'SS1'
WHERE  SC='S1';
```

При выполнении данного запроса изменения в главной таблице S в той же транзакции передаются в подчиненную таблицу SP (т.е. осуществляется каскадное изменение кода поставщика на SS1 во всех поставках поставщика S1.

5)Изменить код детали P1 на PP1

UPDATE P SET PC = 'PP1'
WHERE PC='P1';

Изменения в главной таблице P будут блокированы, если в подчиненной таблице SP есть поставки детали P1

6)Изменить код поставщика S1 на S6 во всех поставках поставщика S1.

UPDATE SP SET SC = 'S6'
WHERE SC='S1';

Изменения в подчиненной таблице SP будут блокированы, если в главной таблице S нет поставщика с кодом S6 (запись подчиненной таблицы может сменить родителя, но остаться без него не должна).

Поисковый **оператор удаления** имеет формат:

DELETE FROM <ИД>
[WHERE <спецификация выбора записей>]

Если не указана опция отбора записей, то удаляются все строки ИД.

1)Удалить информацию о поставщике с кодом s3

DELETE
FROM S
WHERE SC='s3';

При выполнении данного запроса в той же транзакции происходит каскадное удаление всех поставок поставщика s3 из таблицы SP (для внешнего ключа SC при создании таблицы SP было прописано каскадное удаление и обновление (см. далее).

```
CREATE TABLE SP (
sc varchar(5) not null references
  S (sc) on delete cascade on update
  cascade ,
pc varchar(5) not null references
  P (pc),
date_sp date,
qty integer default 100,
primary key (sc, pc, date_sp),
check (qty between 100 and 1000)
);
```

2)Удалить информацию обо всех деталях, которые производятся в Лондоне

DELETE
FROM P

WHERE city='London';

Попытка удаления строк, соответствующих деталям из Лондона будет блокирована (с выдачей сообщения), если есть поставки с кодами этих деталей в таблице SP (см. далее). Режим «Запретить» действует по умолчанию, так как для внешнего ключа PC при создании таблицы SP не было прописано каскадирование.

3) Удалить все поставки поставщика S1 с объемом поставки больше 200 деталей

DELETE
FROM SP
WHERE SC='S1' AND QTY>200

Строки, соответствующие условию, из подчиненной таблицы удаляются бесконтрольно.

ВЫВОДЫ

- *Внешние ключи задаются, чтобы при выполнении запросов на добавление, изменение и удаление данных СУБД автоматически отслеживала ограничения ссылочной целостности.*
- *Механизм внешних ключей не используется при выполнении запросов на выборку данных!*