Изменение определения базовой таблицы

ALTER TABLE table_name action

Возможны следующие действия:
ADD COLUMN column_name { datatype | domain } [DEFAULT value]
[constraint_list]
ALTER COLUMN column_name SET DEFAULT value
ALTER COLUMN column_name DROP DEFAULT
DROP COLUMN column_name { RESTRICT | CASCADE }
ADD CONSTRAINT [constraint_name] constraint_expression
DROP CONSTRAINT constraint_name [{ RESTRICT | CASCADE }]

Добавление или удаление столбцов таблицы

ADD COLUMN column_name { datatype | domain } [DEFAULT value] [constraint_list]
Отличие от определения столбца при создании таблицы: для существующих строк в таблице значением данного столбца является значение по умолчанию
→ добавляемый столбец обязан иметь явно или неявно определенное значение по умолчанию
Пример: ALTER TABLE EMPLOYEE ADD COLUMN EMP_BONUS SALARY DEFAULT NULL CONSTRAINT BONUS CHECK(VALUE < EMP_SALARY);

DROP COLUMN column_name { RESTRICT | CASCADE }
RESTRICT – действие отвергается, если столбец является единственным в таблице или имя столбца используется в определениях ограничений (внешних по отношению к данной таблице) или виртуальных таблиц (представлений)
CASCADE – действие выполняется, если столбец не является единственным в таблице, при этом отменяются определения ограничений и представлений, в которых задействован удаляемый столбец
Пример: ALTER TABLE DEPARTMENT DROP COLUMN DEPT_EMP_NUM CASCADE;

Изменение набора табличных ограничений

ADD CONSTRAINT [constraint_name] constraint_expression
Если к моменту исполнения ALTER TABLE ADD CONSTRAINT в таблице существуют строки, противоречащие новому ограничению, то СУБД должна отвергнуть этот оператор
Пример: ALTER TABLE DEPARTMENT ADD CONSTRAINT DEPT_EMP_NUMBER CHECK((SELECT COUNT(*) FROM EMPLOYEE WHERE DEPT_ID = EMPLOYEE.DEPT_ID) <= 100);

DROP CONSTRAINT constraint_name [{ RESTRICT | CASCADE }]
RESTRICT и CASCADE имеют смысл только для ограничений первичного или возможного ключей (RESTRICT – действие отвергается, если существует хотя бы один внешний ключ, ссылающийся на отменяемый; CASCADE – действие выполняется всегда, при этом отменяются определения всех ссылающихся внешних ключей)

Отмена определения базовой таблицы

DROP TABLE table_name { RESTRICT | CASCADE }

DROP TABLE RESTRICT отвергается, если таблица задействована в определении представлений или ограничений целостности (не считая собственных табличных ограничений)

DROP TABLE CASCADE выполняется всегда по следующим правилам:

- Уничтожаются все ограничения целостности и представления, в определениях которых задействована данная таблица
- Уничтожаются все строки, хранящиеся в данной таблице, а также определения ее столбцов и табличных ограничений (таблица перестает существовать)

Ограничения целостности БД

Иерархия ограничений целостности языка SQL

Дополнительные ограничения базы данных – общие ограничения целостности

Определение:

CREATE ASSERTION constraint_name constraint_expression

Отмена:

DROP ASSERTION constraint_name

Пример:

CREATE ASSERTION DEPT_MNG_CONSTR
CHECK(NOT EXISTS(SELECT * FROM EMPLOYEE EMP, DEPARTMENT DEPT WHERE EMP.EMP_ID = DEPT.DEPT_MANAGER AND CURRENT_DATE - EMP.EMP_BDATE < INTERVAL ‘30’ YEAR));

Базовые средства манипулирования данными

- Средства манипулирования данными присутствуют в прямом, встраиваемом и динамическом SQL
- К базовым средствам манипулирования данными относятся:
  - Оператор вставки строк (INSERT)
  - Оператор модификации строк (UPDATE)
  - Оператор удаления строк (DELETE)

Оператор вставки строк

INSERT INTO table_name [(column_list)] VALUES row_constructor_list

• Если список столбцов пропущен, то предполагается, что используется список из имен всех столбцов таблицы, указанных в том порядке, в котором они были описаны в операторе CREATE TABLE

• В списке столбцов должны быть обязательно указаны те столбцы, для которых не задано значение по умолчанию (явно или неявно)

• Порядок перечисления значений столбцов в конструкторе строки должен совпадать с порядком перечисления столбцов в соответствующем списке

• В качестве значений столбцов могут использоваться литералы, DEFAULT, NULL, а также скалярные запросы, результат выполнения которых состоит из единственной строки, включающей единственный столбец

Пример:

```
INSERT INTO EMPLOYEE VALUES
  ROW( 319, ‘Иванов И.И.’, ‘1980-07-19’, DEFAULT, NULL, NULL ),
  ROW( 320, ‘Петров П.П.’, ‘1983-05-09’, (SELECT EMP_SALARY FROM EMPLOYEE
WHERE EMP_ID = 200), 12, NULL );
INSERT INTO EMPLOYEE( EMP_ID, EMP_NAME, EMP_BDATE ) VALUES
  ROW( 321, ‘Сидоров С.С.’, ‘1962-12-10’ );
```

Оператор вставки строк – вставка результата запроса

Определим таблицу, хранящую информацию о временных сотрудниках организации, проходящих испытательный срок:

```
CREATE TABLE EMPLOYEE_TEMP (
        EMPT_ID INTEGER PRIMARY KEY,
        EMPT_NAME VARCHAR( 200 ) NOT NULL,
        EMPT_BDATE DATE NOT NULL,
        EMPT_ENTRY_DATE DATE NOT NULL
);
```

Переведем временных сотрудников, чей испытательный срок превысил 3 месяца, в основной штат:

```
INSERT INTO EMPLOYEE( EMP_ID, EMP_NAME, EMP_BDATE ) ( SELECT EMPT_ID, EMPT_NAME, EMPT_BDATE  FROM EMPLOYEE_TEMP WHERE CURRENT_DATE – EMPT_ENTRY_DATE >= INTERVAL ‘3’  MONTH );
```

Оператор модификации строк

UPDATE table_name SET column_value_assignment_list WHERE logical_expression
column_value_assignment := column_name = value_expression

• Для всех строк таблицы с указанным именем вычисляется логическое условие. Строки, для которых значением этого условия является TRUE, считаются подлежащими модификации (S_m – множество модифицируемых строк)

• Каждая строка $s \in S_m$ подвергается модификации таким образом, что значение каждого столбца этой строки, указанного в списке модификации, заменяется значением, указанным в правой части соответствующего элемента списка. Если новое значение задается оператором запроса, в который входят имена модифицируемых столбцов, то под значениями этих столбцов в запросе понимаются их значения до модификации

Пример. Перевести всех служащих, выполняющих проект с номером 5, в отдел 12 и повысить им заработную плату на 5000 руб.

```
UPDATE EMPLOYEE SET DEPT_ID = 12, EMP_SALARY = EMP_SALARY + 5000.00
WHERE PRO_ID = 5;
```

Оператор удаления строк

DELETE FROM table_name WHERE logical_expression

• Для всех строк таблицы с указанным именем вычисляется логическое условие. Строки, для которых значением этого условия является TRUE, считаются подлежащими удалению (S_d – множество удаляемых строк)

• Каждая строка $s \in S_d$ удаляется из указанной таблицы.

Пример. Уволить всех служащих, размер заработной платы которых превышает размер заработной платы начальников их отделов.

```
DELETE FROM EMPLOYEE WHERE EMP_SALARY >
            (SELECT EMP.EMP_SALARY
              FROM EMPLOYEE EMP, DEPARTMENT DEPT
              WHERE EMPLOYEE.DEPT_ID = DEPT.DEPT_ID
               AND DEPT.DEPT_MANAGER = EMP.EMP_ID);
```

Учебный пример

```
CREATE TABLE EMPLOYEE (
…
DEPT_ID INTEGER DEFAULT
NULL REFERENCES
DEPARTMENT ON DELETE SET
NULL,
…
);
```

```
CREATE TABLE DEPARTMENT (
        DEPT_ID INTEGER PRIMARY KEY,
        DEPT_EMP_NUM INTEGER NOT NULL
CHECK( VALUE <= 100 ),
        CHECK ( DEPT_EMP_NUM = ( SELECT
COUNT(*) FROM EMPLOYEE WHERE DEPT_ID =
EMPLOYEE.DEPT_ID )),
…
);
```

Операторы модификации таблицы EMPLOYEE вида:
INSERT INTO EMPLOYEE (…, DEPT_ID, …) VALUES ROW (…, X, …);
UPDATE EMPLOYEE SET DEPT_ID = X WHERE …;
DELETE FROM EMPLOYEE WHERE …;
где X – значение DEPT_ID, отличное от NULL, а во множество удаляемых строк включается хотя бы одна строка, в которой значение столбца DEPT_ID отличается от NULL, нарушают выделенное ограничение целостности

???

Триггеры в SQL

Триггер – хранимая в БД процедура, автоматически вызываемая СУБД при возникновении определенных условий. В языке обеспечиваются возможности определения триггеров, которые вызываются при модификации указанной базовой таблицы (определение триггеров над представлениями не допускается).
Предметная таблица – базовая таблица, с которой связывается определение триггера.
Инициирующий оператор – оператор SQL, выполнение которого приводит к срабатыванию триггера.
Основные области применения триггеров:

- Согласование и очистка данных
- Журнализация и аудит
- Операции, не связанные с изменением БД (генерация отчетов, печать документов, рассылка почты)

Определение триггера

Оператор CREATE TRIGGER (отмена – DROP TRIGGER). В операторе указываются:

1. Имя триггера
2. Момент срабатывания: BEFORE (непосредственно до выполнения инициирующего оператора) или AFTER (сразу после его выполнения)
3. Тип инициирующего оператора: INSERT, UPDATE [OF column_list] или DELETE
4. Имя предметной таблицы: ON table_name [REFERENCING OLD {ROW | TABLE} AS name NEW {ROW | TABLE} AS name]
5. Количество срабатываний триггера: FOR EACH ROW (вызывается для каждой вставляемой, модифицируемой или удаляемой строки) или FOR EACH STATEMENT (один раз для всего выполнения инициирующего оператора)
6. Дополнительное (опциональное) условие применимости триггера: WHEN (logical_expression), выражение вычисляется для каждой обновляемой строки (триггер FOR EACH ROW) или для всей таблицы (триггер FOR EACH STATEMENT), и триггер срабатывает только в том случае, если результат вычисления выражения равен TRUE
7. Тело триггера: одиночный оператор или процедура на языке SQL/PSM

Триггеры для учебного примера

```
CREATE TRIGGER CHANGE_DEPT_I AFTER INSERT ON EMPLOYEE FOR EACH ROW
WHEN( EMPLOYEE.DEPT_ID IS NOT NULL )
        UPDATE DEPARTMENT SET DEPT_EMP_NUM = DEPT_EMP_NUM + 1
WHERE DEPARTMENT.DEPT_ID = EMPLOYEE.DEPT_ID;

CREATE TRIGGER CHANGE_DEPT_D AFTER DELETE ON EMPLOYEE FOR EACH ROW
WHEN( EMPLOYEE.DEPT_ID IS NOT NULL )
        UPDATE DEPARTMENT SET DEPT_EMP_NUM = DEPT_EMP_NUM – 1
WHERE DEPARTMENT.DEPT_ID = EMPLOYEE.DEPT_ID;

CREATE TRIGGER CHANGE_DEPT_U AFTER UPDATE OF DEPT_ID ON EMPLOYEE
REFERENCING OLD ROW AS OLD_EMP NEW ROW AS NEW_EMP FOR EACH ROW
        BEGIN ATOMIC
         UPDATE DEPARTMENT SET DEPT_EMP_NUM = DEPT_EMP_NUM – 1
WHERE DEPARTMENT.DEPT_ID = OLD_EMP.DEPT_ID;
        UPDATE DEPARTMENT SET DEPT_EMP_NUM = DEPT_EMP_NUM + 1
WHERE DEPARTMENT.DEPT_ID = NEW_EMP.DEPT_ID;
        END;
```

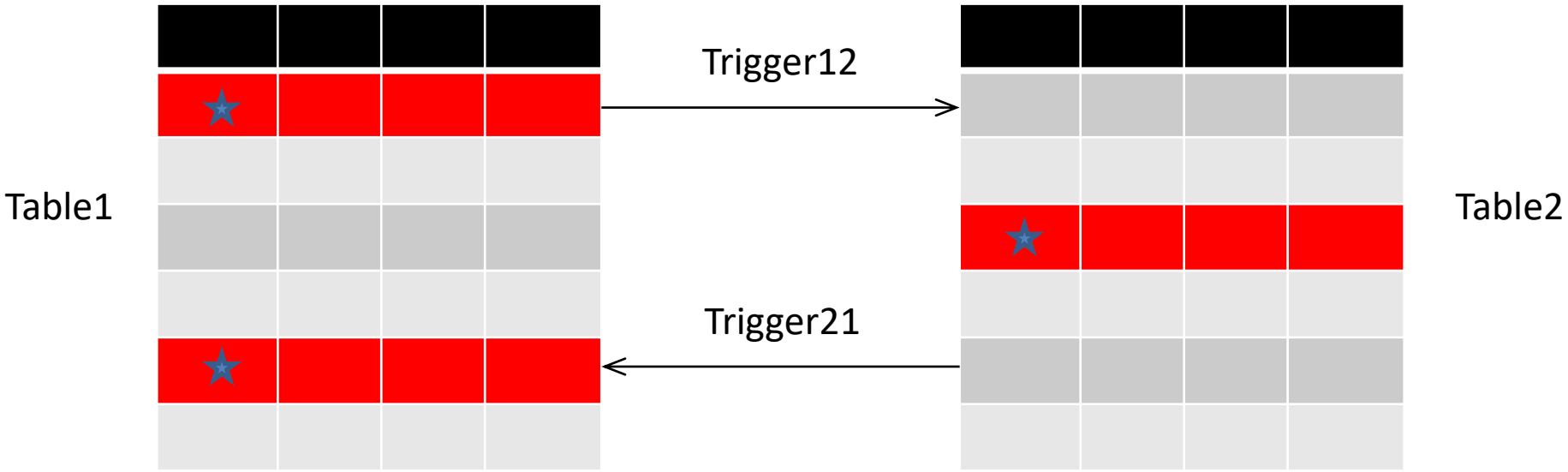
Выполнение триггеров

При выполнении каждого триггера система устанавливает контекст выполнения триггера. Контекст выполнения триггера всегда является атомарным, т.е. инициируемый SQL-оператор либо успешно завершается, либо результаты его действия гарантированно отсутствуют в базе данных. Контекст выполнения триггера включает следующие данные:

- триггерное событие (INSERT, UPDATE или DELETE)
- имя предметной таблицы триггера
- имена столбцов предметной таблицы, специфицированных в определении триггера (для триггеров по UPDATE)
- набор переходов: представление всех вставляемых, модифицируемых или удаляемых строк, а также список уже выполненных триггеров и представлений строк, над которыми эти триггеры выполнялись.

Отслеживание уже выполненных триггеров ведется для предотвращения зацикливания выполнения системы триггеров. Набор переходов изначально пуст, и переходы добавляются при каждом обновлении предметных таблиц выполняемой системы триггеров.

Выполнение триггеров



Набор переходов:

Table1(Row1) -> Trigger 12 -> Table2(Row3)

Table2(Row3) -> Trigger21 -> Table1(~~Row1~~, Row5)

Table1(Row5) -> …