Целостность базы данных в СУБД Oracle

Кто отказывается от многого, может многое себе позволить.

Жак Шардон

Содержание

- Понятие целостности базы данных
- Средства обеспечения целостности базы данных данных
  - Декларативные ограничения целостности
  - Триггеры

Целостность базы данных

- *Целостность базы данных (database integrity)* – защита базы данных от санкционированных пользователей.
  - *Целостность столбца таблицы* – NULL-значения, потенциальные, первичные и внешние ключи
  - *Целостность таблицы* – ограничения на значения в нескольких столбцах одной таблицы
  - *Целостность базы данных* – ограничения на значения в нескольких столбцах различных таблиц

Средства обеспечения целостности

- *Декларативное ограничение целостности (integrity constraint)* – предикат столбца таблицы, задаваемый при создании таблицы. При выполнении запросов сначала выполняется оператор SQL, после чего проверяется значение предиката. Если оно ложно, то выполняется откат.
- *Триггер (trigger)* – хранимая процедура, автоматически выполняемая СУБД Oracle при наступлении определенного пользователем события (обновление конкретной таблицы, вход пользователя в систему и др.).

Корпоративные системы баз данных © М.Л. Цымблер 4

Декларативные ограничения целостности

- NOT NULL
  запрет пустых значений в столбце таблицы
- UNIQUE
  запрет совпадающих значений в столбце таблицы
- PRIMARY KEY
  запрет совпадающих и пустых значений в столбце таблицы
- FOREIGN KEY
  запрет значений внешнего ключа таблицы, для которых нет совпадающих значений столбца адресуемой таблицы
- CHECK
  запрет значений столбца таблицы, нарушающих заданное ограничение на один или несколько столбцов таблицы

Корпоративные системы баз данных © М.Л. Цымблер 5

Ограничение NOT NULL

```
-- Именованное ограничение целостности столбца
CREATE TABLE s (
        sid number CONSTRAINT pk_sid PRIMARY KEY,
        sname char(20) CONSTRAINT nn_sname NOT NULL,
        …);

-- Неименованное ограничение целостности столбца
CREATE TABLE s (
        sid number PRIMARY KEY,
        sname char(20) NOT NULL,
        …);
```

Корпоративные системы баз данных © М.Л. Цымблер 6

Ограничение UNIQUE

```
-- Именованное ограничение целостности столбца
CREATE TABLE s (
        sid number CONSTRAINT pk_sid PRIMARY KEY,
        sname char(20) CONSTRAINT unq_sname UNIQUE,
        …);

-- Именованное ограничение целостности таблицы
CREATE TABLE s (
        sid number PRIMARY KEY,
        sname char(20),
        …,
        CONSTRAINT unq_sname UNIQUE (sname));
```


Ограничение UNIQUE

```
-- Создание таблицы
CREATE TABLE emp (
        empno number CONSTRAINT pk_ empno PRIMARY KEY,
        sname char(20) CONSTRAINT nn_sname NOT NULL,
        areacode char(4),
        phoneno char(7),
        CONSTRAINT unq_acph UNIQUE (areacode, phoneno));

-- Добавление записей в таблицу
INSERT INTO emp VALUES (1, 'Иванов', '3512', '420409');
INSERT INTO emp VALUES (2, 'Петров', '3512', NULL);          -- Можно
INSERT INTO emp VALUES (3, 'Сидоров', NULL, '420409');       -- Можно
INSERT INTO emp VALUES (4, 'Коньков', NULL, NULL);           -- Можно
INSERT INTO emp VALUES (5, 'Егоров', '3512', '420409');      -- Нельзя!
```


Комбинация UNIQUE и NOT NULL

```
-- Создание таблицы
CREATE TABLE emp (
        empno number CONSTRAINT pk_empno PRIMARY KEY,
        name char(20) CONSTRAINT nn_name NOT NULL,
        areacode char(4) NOT NULL,
        phoneno char(7) NOT NULL,
        CONSTRAINT unq_acph UNIQUE (areacode, phoneno));

-- Добавление записей в таблицу
INSERT INTO emp VALUES (1, 'Иванов', '3512', '420409');
INSERT INTO emp VALUES (2, 'Петров', '3512', NULL);          -- Нельзя!
INSERT INTO emp VALUES (3, 'Сидоров', NULL, '420409');       -- Нельзя!
INSERT INTO emp VALUES (4, 'Коньков', NULL, NULL);           -- Нельзя!
INSERT INTO emp VALUES (5, 'Егоров', '3512', '420409');      -- Нельзя!
```

Ограничение PRIMARY KEY

```
-- Именованное ограничение целостности столбца
CREATE TABLE s (
        sid number CONSTRAINT pk_sid PRIMARY KEY,
        sname char(20) CONSTRAINT nn_sname NOT NULL,
        …);

-- Именованное ограничение целостности таблицы
CREATE TABLE s (
        sid number,
        sname char(20) CONSTRAINT nn_sname NOT NULL,
        …,
        CONSTRAINT pk_sid PRIMARY KEY (sid));

-- Неименованное ограничение целостности столбца
CREATE TABLE s (
        sid number PRIMARY KEY,
        sname char(20) NOT NULL, …);
```

Корпоративные системы баз данных © М.Л. Цымблер 10

Ограничение FOREIGN KEY

```
-- Именованное ограничение целостности столбца
CREATE TABLE sp (
        -- Обновлять и удалять объект ссылки внешнего ключа
        -- ЗАПРЕЩЕНО
        sid number CONSTRAINT fk_sid REFERENCES s (sid),
        pid number CONSTRAINT fk_pid REFERENCES p (pid),
        qty number,
        CONSTRAINT pk_sidpid PRIMARY KEY (sid, pid));

-- Именованное ограничение целостности таблицы
CREATE TABLE sp (
        sid number,
        pid number,
        qty number,
        CONSTRAINT fk_sid FOREIGN KEY (sid) REFERENCES s (sid),
        CONSTRAINT fk_pid FOREIGN KEY (pid) REFERENCES p (pid),
        CONSTRAINT pk_sidpid PRIMARY KEY (sid, pid));
```

Корпоративные системы баз данных © М.Л. Цымблер 11

Ограничение FOREIGN KEY

```
-- ССЫЛКА НА СЕБЯ
CREATE TABLE emp (
        empno number CONSTRAINT pk_empno PRIMARY KEY,
        name char(20) CONSTRAINT nn_name NOT NULL,
        age number CONSTRAINT nn_age NOT NULL,
        mgr number CONSTRAINT fk_mgr REFERENCES emp (empno));

-- Добавление записей в таблицу
INSERT INTO emp VALUES (1, 'Иванов', 45, NULL);     -- Можно
INSERT INTO emp VALUES (2, 'Петров', 30, NULL);     -- Можно
INSERT INTO emp VALUES (3, 'Сидоров', 33, 1);       -- Можно
INSERT INTO emp VALUES (4, 'Егоров', 55, 2);        -- Можно
INSERT INTO emp VALUES (5, 'Жуков', 40, 5);         -- Можно (но зачем?)
INSERT INTO emp VALUES (6, 'Коньков', 25, 10);      -- Нельзя!
```

Корпоративные системы баз данных © М.Л. Цымблер 12

Ограничение FOREIGN KEY

```
-- Определение действий при удалении объекта ссылки внешнего ключа
CREATE TABLE emp (
        empno number CONSTRAINT pk_empno PRIMARY KEY,
        name char(20) CONSTRAINT nn_name NOT NULL,
        age number CONSTRAINT nn_age NOT NULL,
        -- Каскадное удаление
        deptno number CONSTRAINT fk_dept
                REFERENCES dept (deptno) ON DELETE CASCADE,
        -- Установка в NULL
        mgr number CONSTRAINT fk_mgr
                REFERENCES emp (empno) ON DELETE SET NULL);
```


Ограничение CHECK

```
-- Определение различных ограничений
CREATE TABLE emp (
        empno number CONSTRAINT pk_empno PRIMARY KEY,
        name char(20) CONSTRAINT nn_name NOT NULL
                DEFAULT 'Имя неизвестно',
        age number CONSTRAINT chk_age
                CHECK (age between 18 and 65),
        city char(20) CONSTRAINT chk_city
                CHECK (city in ('Москва', 'Челябинск', 'Пермь', 'Омск')),
        skill number,
                CONSTRAINT chk_skill  CHECK (skill > age-18)),
        accesscode char(8) CONSTRAINT chk_accesscode
                CHECK (accesscode = upper(accesscode)) );
```


Проверка ограничений целостности

1. Выполняется оператор SQL*
2. Результат выполнения проверяется на соответствие всем ограничениям целостности. Если хотя бы одно из них не выполняется, производится откат.

* При вставке строк пропущенные поля заменяются препроцессором на DEFAULT-значения.
Каскадные действия рассматриваются как неотъемлемая часть выполняемого оператора.

Добавление и изменение ограничений

```
-- Добавление ограничения
ALTER TABLE emp
        MODIFY (salary number CONSTRAINT nn_sal NOT NULL);

-- Добавление ограничения, перемещение «неправильных» строк
ALTER TABLE phone_calls
        ADD CONSTRAINT unq_acpn UNIQUE (areaco, phoneno)
        EXCEPTIONS INTO calls_bad_recs;

ALTER TABLE phone_calls
        ADD CONSTRAINT fk_areaco_phoneno
                FOREIGN KEY (areaco, phoneno)
                        REFERENCES clients(areaco, phoneno)
        EXCEPTIONS INTO wrong_numbers;
```

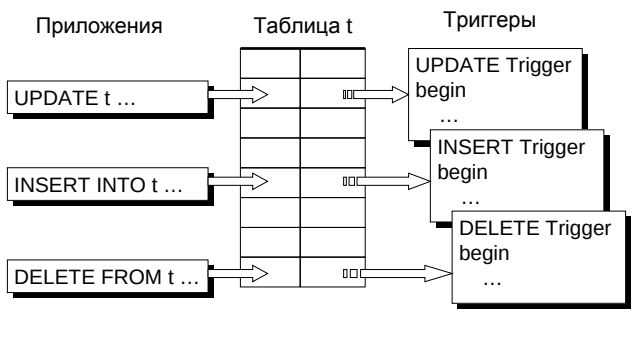
Корпоративные системы баз данных © М.Л. Цымблер 16

Триггеры

- *Триггер (trigger)* – особая хранимая процедура, автоматически выполняемая СУБД Oracle при наступлении определенного пользователем события: создание или модификация указанного объекта схемы, регистрация или выход пользователя, монтирование базы данных, останов экземпляра СУБД.
- Триггер, в отличие от хранимых процедур, не может быть выполнен явно.
- Триггер может быть реализован на PL/SQL, Java, C.

Корпоративные системы баз данных © М.Л. Цымблер 17

Триггеры



Корпоративные системы баз данных © М.Л. Цымблер 18

Применение триггеров

- Реализация сложных ограничений целостности и правил предметной области.
- Реализация ограничений ссылочной целостности в случае распределенной базы данных.
- Аудит, сбор и публикация статистики о выполнении операторов SQL, доступе к таблицам и др. событиях.
- Синхронная репликация таблиц.
- Модификация данных в базовых таблицах необновляемых представлений.

Части триггера

CREATE OR REPLACE TRIGGER check_weight	**Заголовок**
BEFORE INSERT OR UPDATE ON SP	**Событие**
WHEN new.qty > old.qty	**Ограничение**

FOR EACH ROW
DECLARE
p_weight, total_weight number;
BEGIN
select weight into p_weight from P where pid = :new.pid;
total_weight = :new.qty * p_weight;
if total_weight > 1500 then
raise_application_error(-20601, 'Ошибка!');
end if;
END;
	Тело

Типы триггеров

- *Триггер строки* запускается каждый раз при наступлении события триггера.
- *Триггер события* запускается один раз при наступлении события триггера.
- *Триггер BEFORE/AFTER* запускается до/после наступления события триггера.
- *Триггер комбинированного типа* – триггер строки BEFORE, триггер события BEFORE и т.п.
- *Триггер INSTEAD-OF* используется для модификации базовых таблиц необновляемых представлений
- *Триггер системного события* – старт, останов экземпляра СУБД, ошибка сервера.
- *Триггер пользовательского события* – начало и завершение сессии, BEFORE/AFTER CREATE/ALTER/DROP <объекта схемы>.

Выполнение триггеров

1. Выполнить все BEFORE-триггеры события
2. Цикл для каждой измененной триггером строки
   1. Выполнить все BEFORE-триггеры строки
   2. Заблокировать и изменить строку, выполнить проверку ограничений целостности
   3. Выполнить все AFTER-триггеры строки
3. Выполнить отложенную проверку декларативных ограничений целостности
4. Выполнить все AFTER-триггеры события

Пример: триггер строки AFTER

```
-- Обновление вычисляемого поля «Зарплата» таблицы «Отделы»
CREATE TRIGGER total_salary
AFTER DELETE OR INSERT OR UPDATE OF deptno, sal ON emp
FOR EACH ROW
begin
   if deleting or (updating and :old.deptno != :new.deptno) then
      UPDATE dept SET total_sal = total_sal - :old.sal
      WHERE deptno = :old.deptno;
   end if;
   if inserting or (updating and :old.deptno != :new.deptno) then
      UPDATE dept SET total_sal = total_sal + :new.sal
      WHERE deptno = :new.deptno;
   end if;
   if (updating and :old.deptno = :new.deptno and :old.sal != :new.sal ) then
      UPDATE dept SET total_sal = total_sal - :old.sal + :new.sal
      WHERE deptno = :new.deptno;
   end if;
end;
```


Пример: INSTEAD-OF триггер

```
CREATE TABLE dept ( -- Отделы
        deptno NUMBER PRIMARY KEY,
        dname CHAR(10),
        mgr NUMBER );
CREATE TABLE emp (-- Служащие
        empno NUMBER PRIMARY KEY,
        ename CHAR(20),
        deptno NUMBER REFERENCES dept(deptno));
ALTER TABLE dept ADD (
        FOREIGN KEY(mgr) REFERENCES emp (empno));

CREATE VIEW managers AS -- Начальники отделов
        SELECT dept.deptno, dept.sname, emp.empno, emp.ename
        FROM emp, dept
        WHERE emp.empno = dept.mgr;
```

Пример: INSTEAD-OF триггер

```
CREATE TRIGGER managers_insert
        INSTEAD OF INSERT ON mgr_info
        REFERENCING NEW AS n                    -- новая запись
        FOR EACH ROW
DECLARE
        empcnt number;
begin
/* Найдем количество служащих в отделе */
        SELECT COUNT(*) INTO empcnt FROM emp
        WHERE emp.deptno = :n.deptno;
/* Если в отделе есть служащие, то новый может быть начальником*/
        if empcnt >= 1 then
                UPDATE dept SET mgr = :n.empno
                WHERE dept.deptno = :n.deptno;
        end if;
end;
```

Корпоративные системы баз данных © М.Л. Цымблер 25

Пример: триггер системного события

```
CREATE TRIGGER register_shutdown
        ON DATABASE SHUTDOWN
BEGIN
        ...
        DBMS_AQ.ENQUEUE(...);
        ...
END;
```

Корпоративные системы баз данных © М.Л. Цымблер 26

Разработка триггеров

- Не создавать триггеры, которые дублируют функциональность СУБД (декларативные ограничения целостности и проч.).
- Не создавать рекурсивные триггеры (например, триггер AFTER INSERT выполняет оператор INSERT над той же таблицей).
- Не создавать несколько триггеров одного события для одной и той же таблицы (порядок их выполнения неизвестен).
- Использовать хранимые процедуры для уменьшения размера триггеров.

Корпоративные системы баз данных © М.Л. Цымблер 27

Ограничения целостности vs триггеры

- Если правило целостности базы данных не удается реализовать при помощи ограничений целостности, нужно реализовать его при помощи триггеров.
  - Триггер не действует на данные, помещенные в таблицу до его создания.
  - Триггер при выполнении может (неявно) запустить другие триггеры
  - Ограничение целостности не может реализовать сложные правила целостности базы данных
  - Ограничение целостности не может задействовать таблицы удаленных узлов базы данных

Корпоративные системы баз данных © М.Л. Цымблер 28