Лекция 14. Защита данных в реляционных СУБД: контроль целостности данных.

Каждая СУБД включает инструменты для реализации таких функций:

1) **Функция секретности**, которая состоит в предупреждении несанкционированного доступа к данным, их изменения или разрушения со стороны пользователя.
2) **Функция безопасности**, состоящая в предупреждении изменений или разрушений данных при сбоях аппаратных или программных средств, а также нарушения целостности данных из-за действий пользователя.

Соответственно, защита данных включает такие аспекты, как:

1. **Контроль целостности данных при обновлении**. Требует поддержки некоторых правил целостности по данным.
2. **Восстановление данных после сбоев** аппаратных или программных средств. Приводит базу данных к некоторому согласованному состоянию .
3. **Параллелизм** - предотвращение конфликтов совместного доступа к данным.
4. **Секретность** - предотвращение несанкционированного доступа к данным, их изменению или разрушению со стороны пользователя.

В данной лекции будет рассмотрен вопрос контроля целостности данных при обновлении.

Целостность данных в системах баз данных обеспечивается набором некоторых ограничений на значения данных. В реляционных базах данных существует два общих правила целостности – по сущностям и по связям, а также допускается произвольное число специализированных правил целостности (см. лекцию 9). Все ограничения целостности фиксируются в системном каталоге и имеют собственные имена.

Ограничения целостности данных касаются доменов отдельных атрибутов, либо всего отношения целиком, однако в коммерческих РСУБД есть возможности определения общих ограничений целостности.

14.1 Ограничения на значения атрибута реляционной таблицы.

Ограничения целостности на значения атрибута определяются внутри определения реляционной таблицы так:

```
[CONSTRAINT имя_ограничения]
     {NOT NULL |
      DEFAULT значение |
      UNIQUE |
      CHECK логич.условие}
```

где

`значение` – допустимое значение для атрибута, соответствующее его домену значений;

`логич.условие` – допустимое в SQL выражение, результатом которого может быть одно из значений «Истина» или «Ложь».

Ограничение NOT NULL указывает, что значения данного атрибута являются обязательными для каждого кортежа данной таблицы, и запрещает NULL-значения.

Ограничение DEFAULT указывает значение по умолчанию для данного атрибута таблицы. Это значение должно быть допустимым для домена данного атрибута**.**

Пример 1.
В таблице СОТРУДНИКИ для должности сотрудника по умолчанию ставить «менеджер».

CREATE TABLE СОТРУДНИКИ (
…
должность char(15) DEFAULT "менеджер"
…);

Ограничение UNIQUE требует уникальности значения данного атрибута в данной таблице. Как правило, ограничение UNIQUE при определении атрибута указывает, что этот атрибут является потенциальным ключом.

Ограничение CHECK задает условие, которому должно удовлетворять каждое вводимое значение атрибута в данной таблице. В зависимости от СУБД, это условие может принимать тот или иной вид, например, для SQL Server условие, применяемое в ограничении CHECK, имеет вид:

имя_атрибута IN список_значений_через_запятую

или

имя_атрибута IN нижняя_граница TO верхняя_граница

Пример 2.
В таблице ПОСТАВЩИКИ ограничить значения кода поставщика четырьмя цифрами.

CREATE TABLE ПОСТАВЩИКИ (
…
кодПоставщика char(4) CHECK (кодПоставщика LIKE "####")
…);

14.2 Ограничения на значения домена.

Ограничение целостности домена в общем виде задаются так:

```
CREATE DOMAIN имя_домена тип_данных [DEFAULT значение]
[[CONSTRAINT имя_ограничения]
     CHECK VALUES IN список значений;
```

где

`логич.условие` – допустимое в SQL выражение, результатом которого может быть одно из значений «Истина» или «Ложь».

Пример 3.

Уровень гостиницы определяется многими факторами и выражается количеством «звездочек».
Домен «Звездность_гостиницы» можно задать так:

```
CREATE DOMAIN star INTEGER
     CHECK (VALUES IN 1,2,3,4,5);
```

14.3 Глобальные ограничения на значения кортежей таблицы.

Ограничения целостности таблицы в общем виде задаются так:

```
CREATE TABLE имя_таблицы
(список атрибутов таблицы, разделенный запятыми)
PRIMARY KEY (список атрибутов)                  // первичный ключ
[UNIQUE (список атрибутов)]                     // потенциальные ключи
FOREIGN KEY (список атрибутов)
     REFERENCES имя_родительской таблицы        // внешние ключи
[ON DELETE опция]
[ON UPDATE опция]
     [CHECK (условие)]                          // проверочные условия
```

Ограничения целостности таблицы, заданные в таком виде, являются частью определения таблицы.

14.4. Ограничения целостности в виде правил или предположений

В самом общем виде можно задать ограничения целостности в виде правил, применимых ко всем элементам базы данных. Такие правила являются самостоятельными элементами базы данных, и имеют такой же вес, что и таблицы, индексы.

Правила целостности в SQL (диалект Oracle) имеют вид:

```
CREATE ASSERTION имя_правила CHECK логич.условие;
```

где

`логич.условие` – допустимое в SQL выражение, результатом которого может быть одно из значений «Истина» или «Ложь».

.

В диалекте SQL Server

```
CREATE RULE имя_правила AS @логич.условие1;
```

где

`логич.условие1` – выражение с одной переменной – именем атрибута, к которому применяется ограничение, или допустимое в SQL выражение, результатом которого может быть одно из значений «Истина» или «Ложь».

Пример 4.
«Каждая поставка осуществляется на количество товара, большее нуля» (Oracle)

```
CREATE ASSERTION r1
     CHECK     (NOT EXISTS (
               SELECT кол-во
               FROM ПОСТАВКИ
               WHERE NOT (кол-во > 0))> 0);
```

Пример 5.
«Запрещено оформление заказов на сумму больше 100000 у.е.» (MS SQL Server)

```
CREATE RULE R2
     AS (NOT EXISTS (
                SELECT *
                FROM ORDERS
                WHERE SUM > 100000));
```

Для любого ограничения целостности в SQL можно указать директиву DEFERRABLE (можно отложить проверку ограничения) или NOT DEFERRABLE (нельзя откладывать)

Откладывание ограничения задается таким образом:

```
SET CONSTRAINT имя_правила опция;

     где опция ={IMMEDIATE,          // проверять немедленно
                 DEFERRED}           // отложить проверку
```

Ограничения с NOT DEFERRED проверяются немедленно, все остальные ограничения – при выполнении операции успешного завершения транзакции COMMIT (о транзакциях – см. ниже)

Пример 6.

```
SET CONSTRAINT r1 DEFERRED;
```

Это ограничение будет проверяться лишь при выполнении оператора COMMIT*, а не сразу при обработке запроса.*