БАЗЫ ДАННЫХ И ИНФОРМАЦИОННЫЕ СИСТЕМЫ

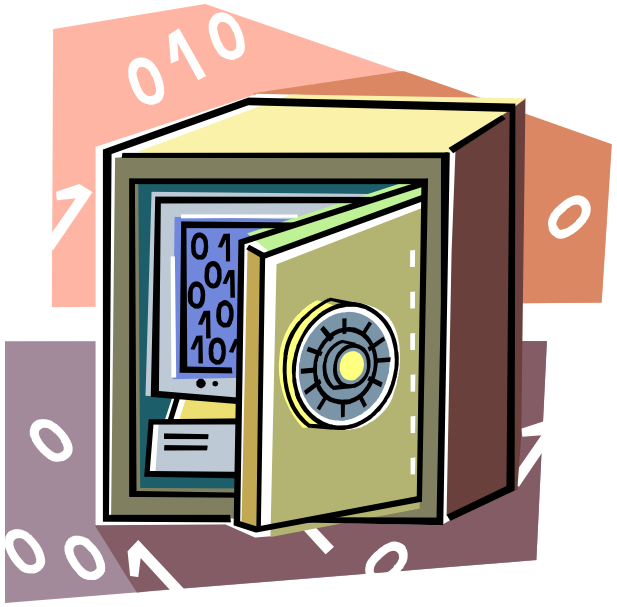
Ст.преп. каф. ИТ Кеберле Наталья Геннадьевна

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

На этой лекции

Мы выясним,

- как СУБД вообще обеспечивает безопасность данных
- какие аспекты «безопасности баз данных» вообще существуют



И рассмотрим один из аспектов безопасности данных – ограничения целостности

Защита данных в СУБД: варианты

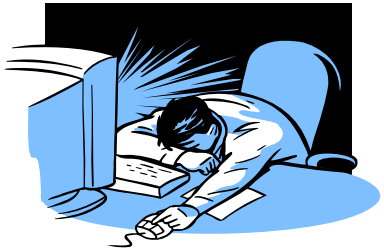
- Каждая СУБД имеет инструменты для реализации таких функций:

Функция секретности – предупреждение несанкционированного доступа к данным, их изменения или разрушения со стороны пользователя.

Защита данных в СУБД: варианты

- Каждая СУБД включает инструменты для реализации таких функций:

- **Функция безопасности –** предупреждение изменений или разрушений данных при сбоях аппаратных или программных средств,



а также нарушения целостности данных из-за действий пользователя.

Аспекты защиты данных

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

Целостность реляционных данных

«Поставщик А сделал заказ на (-100) кондиционеров» - бессмысленное значение.

В любой момент времени любая база данных содержит некоторый набор значений данных

Предполагается, что база данных является моделью реального мира

Однако, набор значений не имеет смысла, если значения **не представляют** определенного состояния реального мира

Целостность реляционных данных

- Определение базы данных нуждается в определении правил целостности.
- Суть правил состоит в том, чтобы сообщать СУБД об ограничениях на сочетания значений в реальном мире. Такие правила целостности называются **специализированными**, т.к. они применяются только в конкретной базе данных.

Целостность реляционных данных

- В реляционной модели данных есть **и общие для всех реляционных баз данных** правила целостности, и **специализированные**, относящиеся к конкретной базе данных.

- Общих ограничений всего два:
  - Правило целостности по сущностям
  - Правило целостности по связям (или ссылочная целостность)

Целостность по сущностям

В первичном ключе не должно быть атрибутов, которые могут принимать неизвестные (неопределенные) значения.

- Т.е. атрибуты первичного ключа должны объявляться как NOT NULL

Целостность по связям

База данных не должна содержать несогласованных значений внешних ключей

- **Несогласованное значение** – это значение внешнего ключа, для которого не существует отвечающего ему значения первичного ключа

- Определить внешний ключ можно оператором SQL:

FOREIGN KEY (список атрибутов внешнего ключа)
REFERENCES имя_таблицы с первичным ключом
(список атрибутов первичного ключа)

Об ограничениях целостности

- Ограничения целостности (**integrity constraints**)
  - отдельные объекты с уникальными именами в пределах одной БД
  - хранятся в отдельной системной БД
  - задаются с помощью логического выражения
    - если оно истинно – значит, ограничение целостности выполнено,
    - если ложно – значит, нет
  - проверяются либо неявно – при каждом изменении БД, либо явно (см. далее)

Классификация ограничений целостности

- Для домена
- Для атрибута
- Для таблицы
- Для всей базы данных

В настоящее время разработчики СУБД такой жёсткой классификации не придерживаются

Ограничение целостности таблицы

```
ALTER TABLE …
ADD CONSTRAINT имя_ограничения
        CHECK (логическое_выражение);
```

Или

```
CREATE TABLE …(
…
CONSTRAINT имя_ограничения
        CHECK (логическое_выражение)
);
```

Ограничение целостности таблицы

Пример: установить для поля «почтовыйИндекс» ограничение на значение «только цифры»

```
ALTER TABLE Покупатель
        ADD CONSTRAINT chkPostalCode
        CHECK (почтовыйИндекс
                LIKE '[0-9][0-9][0-9][0-9][0-9]' );
```

Ограничения целостности домена

Определяет множество значений домена

Абстрактный вид конструкции:

CREATE DOMAIN имя_домена тип_данных
[DEFAULT значение]
[[CONSTRAINT имя_ограничения]
логическое выражение

```
CREATE DOMAIN цвет char(5) DEFAULT ‘красн’
  VALUES (‘красн’, ‘син’, ‘черн’, ‘бел’);
```

Ограничения целостности домена

- Для MS SQL Server вместо понятия домена используется понятие **пользовательский тип данных** (user-defined datatype, UDT)

- Технология создания пользовательского типа данных (актуальна для версий младше 2005, но пока работает и в 2005+):

  - Создать новый UDT с конкретным именем и базовым типом данных
  - Создать правило проверки значений нового UDT
  - Привязать правило к новому UDT

Технология создания домена/UDT

// Создать новый тип данных
// EXEC sp_addtype имя_пользов_типа_данных,
// ‘базовый_тип_данных [(размер)]',
// ‘отношение_к_NULL_значениям'

```
EXEC sp_addtype цвет, ‘char(5)’, ‘NOT NULL’
```

// Создать правило проверки значений нового типа данных
// CREATE RULE имя_правила AS логическое условие

```
CREATE RULE мои_цвета AS @list IN (‘красн’,‘син’,‘бел’,‘черн’);
```

// Привязать правило к типу данных
// USE master;
// EXEC sp_bindrule ‘имя_правила', ‘имя_пользов_типа_данных‘

```
EXEC sp_bindrule ‘мои_цвета’, ‘цвет’;
```

Ограничения целостности атрибута

```
CREATE TABLE …(
имя_атрибута тип_данных CONSTRAINT
    имя_ограничения
    {NOT NULL | DEFAULT значение | UNIQUE |
     CHECK (логическое_выражение)}
);
```

или

```
ALTER TABLE ... ALTER COLUMN …
    ADD CONSTRAINT имя_ограничения
    {NOT NULL | DEFAULT значение | UNIQUE |
     CHECK (логическое_выражение)};
```

Ограничения целостности на базу данных

```
CREATE RULE имя_правила
    CHECK (логическое_выражение);
```

Пример:

```
CREATE RULE chkNetOtrizatZnach
AS
NOT EXISTS (
                    SELECT *
                    FROM ORDERS
                    WHERE sum_stoimost < 0));
```

Проверка ограничений целостности

В MS SQL Server 2003 (с правами db_owner или sysadmin)

DBCC CHECKCONSTRAINTS
[(‘имя_таблицы’ | ‘имя_ограничения’)]
[WITH { ALL_ERRORMSGS | ALL_CONSTRAINTS }]

Пример:

USE aviaraspisanie; // имя БД “Расписание авиарейсов”

// проверяем все ограничения для таблицы “пригодность”

DBCC CHECKCONSTRAINTS (‘пригодность‘);

Проверка ограничений целостности

Пример:

```
USE aviaraspisanie; // имя БД
// проверяем ограничение внешнего ключа для
    таблицы “пригодность”
DBCC CHECKCONSTRAINTS
    (‘FK_рейс_пригодность‘);
```

Ответ может быть таким:

Table name	Constraint name	Where
пригодность	FK_рейс_пригодность	Рейс=1

Т.е. в таблице ‘пригодность’ есть ссылка на рейс 1, которому нет соответствия в таблице ‘рейс’

Итоги

- Ограничения целостности – способ обеспечения непротиворечивости БД в любом состоянии
- С помощью простой конструкции

  ALTER TABLE …ADD CONSTRAINT…

получаем возможность создавать правила проверки ограничений целостности

Вопросы:

Вариант1

1) от чего/кого надо защищать данные?

2) Что означает ALTER TABLE...?

Вариант2

1) что делать, если в БД оказались противоречивые значения?

2) Какая конструкция SQL задает ограничение целостности в виде внешнего ключа?

Вариант3

1) Можно ли в ограничении целостности задать в качестве логического выражения любую конструкцию SQL?

2) Что такое UDT?

Вариант4

1) Какой критерий “правильной” конструкции SQL для логического выражения есть?

2) Какие виды ограничений целостности можно задать с помощью предложения CHECK…?

А ты…

сделал резервные копии?