Оглавление

Создание объектов базы данных

Что такое объекты базы данных

База данных состоит из различных объектов, таких как таблицы, виды (view), домены, хранимые процедуры, триггеры и т.п.

Примеры объектов базы данных

- Таблицы (Tables)
- Столбцы (Columns)
- Типы данных (Data types)
- Домены (Domains)
- Ограничения (Constraints)
- Индексы (Indexes)
- Представления (Views)
- Хранимые процедуры (Stored procedures)
- Триггеры (Triggers)

Для манипулирования объектами базы данных применяется подмножество языка SQL которое называют DDL.

Основные команды DDL

Команды языка определения данных DDL (**Data Definition Language**, язык определения данных) — это подмножество **SQL**, используемое для определения и модификации различных структур данных. К данной группе относятся команды предназначенные для создания, изменения и удаления различных объектов базы данных. Команды

CREATE (создание)
ALTER (модификация)
DROP (удаление)

имеют большинство типов объектов баз данных (таблиц, представлений, процедур, триггеров, табличных областей, пользователей и др.). Т.е. существует множество команд **DDL**, например, **CREATE TABLE**, **CREATE VIEW**, **CREATE PROCEDURE**, **CREATE TRIGGER**, **CREATE USER**, **CREATE ROLE** и т.д.

Создание таблицы в базе данных

Таблицы создаются при помощи команды CREATE TABLE. Полный синтаксис команды

- MS SQL Server
- MySQL Server

Упрощенный вид можно представить так:

```
CREATE TABLE table_name (
    column1 datatype [default value] [nullable],
    column2 datatype [default value] [nullable],
    column3 datatype [default value] [nullable],
   ....
);
```

Имена объектов базы данных (включая имена таблиц и столбцов) должны подчиняться правилам именования объектов базы данных для данного конкретного сервера БД. Правила именования включают набор допустимых символов, длину имен, требования к уникальности имен объектов БД и т.п.

Типы данных для столбцов должны быть указаны обязательно и являются специфичными для каждого сервера БД (MS SQL, MySQL, Oracle…).

Пример:

```
CREATE TABLE LECTURERS(
       ID int PRIMARY KEY NOT NULL,
       SURNAME nvarchar(50) NOT NULL,
       NAME nvarchar(30),
       CITY nvarchar(50),
       UNIV_ID int
);
```

Для MySQL

```
CREATE TABLE lecturers (
  id int(11) PRIMARY KEY  NOT NULL,
  surname varchar(50),
  name varchar(30),
  city varchar(50),
  univ_id int(11)
);
```

```
create table TBL_EXAMPLE
(
       id int PRIMARY KEY
       ,fld1 char(50) default 'name' NOT NULL
       ,fld2 int
       ,fld3 datetime
);
```

Изменение таблицы

Переименование таблиц и колонок (MS SQL)

Каждое переименование таблицы следует тщательно планировать. Если существующие запросы, представления, определяемые пользователем функции, хранимые процедуры или программы ссылаются на таблицу, изменение имени таблицы сделает эти объекты недействительными.

Переименование таблиц и колонок в MS SQL Server выполняется при помощи хранимой процедуры sp_rename:

```
sp_rename 'TBL_EXAMPLE', 'TABLE_EXAMPLE';
```

```
Переименование столбца
sp_rename 'TABLE_EXAMPLE.fld2', 'field2', 'COLUMN';
```

Переименование таблиц и колонок MySQL

Для переименования таблиц и столбцов в MySQL используется команда ALTER TABLE:

ALTER TABLE t1 **RENAME** t2;

ALTER TABLE TBL_EXAMPLE RENAME TABLE_EXAMPLE;

Переименование столбца:

ALTER TABLE table_name CHANGE COLUMN old_name TO new_name;

ALTER TABLE TABLE_EXAMPLE CHANGE COLUMN fld2 field2 int;

Добавление столбца в таблицу базы данных

ALTER TABLE <table name> **ADD** <*имя столбца*> <*тип данных*>**[**<*размер*>**][,...]**;

Значение колонки для всех существующих строк устанавливается в **NULL.**

```
alter table TABLE_EXAMPLE add new_col integer;
```

Модификация колонки

ALTER TABLE <*имя таблицы*> **ALTER COLUMN** <*имя столбца*> <*тип данных*>**[**<*размер*>**]**

```
alter table TABLE_EXAMPLE alter column new_col integer NOT NULL;
```

MySql

ALTER TABLE TABLE_EXAMPLE MODIFY new_col int NOT NULL;

В каких случаях это возможно:

- изменение типа данных возможно только в том случае, если столбец пуст
- для незаполненного столбца можно изменять размер/точность
- для заполненного столбца размер/точность можно только увеличить
- ограничение **NOT NULL** может быть установлено, если ни одно значение в столбце не содержит **NULL**.

Опцию **NOT NULL** всегда можно отменить, разрешается изменять значения, установленные по умолчанию

Создание индексов

Индекс – это структура данных, постоянно сохраняющаяся в базе данных и предназначенная для ускорения операций поиска данных в таблице. Индекс может быть построен по одной или нескольким колонкам таблицы базы данных и обладать (или не обладать) свойством уникальности. Индекс, обладающий свойством уникальности, гарантирует то, что в таблице, для которой он построен, в поле по которому проведена индексация ВСЕГДА будут содержаться уникальные значения. Свойство уникальности распространяется только на колонку таблицы, по которой построен уникальный индекс.

Индексы также бывают кластерные и некластерные. Кластерные индексы обеспечивают физическое хранение информации в колонке в соответствии с индексом.

Индексы создаются при помощи команды CREATE INDEX (MS SQL, My SQL). Упрощенно команда выглядит следующим образом:

```
CREATE [UNIQUE] INDEX index_name
ON table_name (column1, column2, ...);
```

Пример:

```
CREATE INDEX FK_Universities ON LECTURERS( UNIV_ID ASC );
```

Модификация и удаление индекса

Изменение индекса выполняется командой ALTER INDEX, а удаление командой DROP INDEX

Удаление объектов базы данных

Удаление объектов базы данных выполняется командой DROP. Например для удаления таблицы необходимо выполнить команду

DROP TABLE <table_name>,

а для удаления VIEW команду

DROP VIEW <view_name>

Ограничения (CONSTAINTS)

Ограничение в базе данных — декларативно определенное правило, ограничивающее значения, которые могут быть введены в столбец в таблице. Ограничения базы данных являются декларативными, так как определяют ограничения как часть структуры таблицы при ее создании или изменении. Будучи однажды ассоциировано с таблицей, ограничение всегда поддерживается, если его явно не удалить или деактивировать. Размещение ограничений в базе данных имеет следующие преимущества:

- *Централизация.* Ограничение базы данных определяется только один раз и может автоматически использоваться всеми клиентами, обращающимися к базе данных. Определение ограничения в базе данных освобождает разработчика от необходимости вносить одни и те же ограничения в каждую форму, которая использует данную информацию. Кроме того, при необходимости модифицировать ограничение изменения вносятся только в один объект.

- *Защита.* Ограничения базы данных всегда поддерживаются, *независимо* от того, какой инструмент доступа к данным используется. С другой стороны, ограничения, определенные в приложении, могут быть нарушены пользователем, использующим для доступа к тем же таблицам другое приложение или инструмент.

- *Простота.* Ограничения базы данных просты в определении и не требуют никакого программного кода.

Ограничение NOT NULL

По умолчанию колонки в таблице создаются со свойством NULLABLE, т.е. такими, которые могут содержать значение NULL.

Ограничение NOT NULL запрещает ввод в колонку таблицы пустых значений. Оно всегда применяется к отдельным колонкам. Ограничения NOT NULL гарантируют, что для важных данных всегда имеются значения.

Ограничение DEFAULT

Ограничение DEFAULT используется, чтобы обеспечить значение по умолчанию для столбца. Значение по умолчанию добавляется ко всем новым записям, если не указано другое значение.

```
Пример с использованием default значения

create table TBL_EXAMPLE
(
	id int PRIMARY KEY
	,fld1 char(50) default 'name' NOT NULL
	,fld2 int
	,fld3 datetime
);
```

Ограничение UNIQUE

Ограничение UNIQUE запрещает пользователю ввод в столбец или набор столбцов дублированных значений. Ограничение UNIQUE может активироваться для отдельного столбца или для комбинации столбцов. В последнем случае ограничение UNIQUE иногда называется составным ограничением UNIQUE. Ограничения UNIQUE используются, чтобы гарантировать, что в таблице не будет дублированных значений столбцов.

Ограничение PRIMARY KEY

Ограничение PRIMARY KEY гарантирует, что каждая строка в таблице будет уникально идентифицирована значением в столбце или наборе столбцов первичного ключа. Ограничение по первичному ключу объединяет функции ограничения UNIQUE и ограничения NOT NULL.
Обычно рекомендуется определение PRIMARY KEY в каждой создаваемой таблице. Использование первичного ключа может значительно повысить быстродействие доступа к строкам таблицы. Ограничение PRIMARY KEY также используется для поддержания ссылочной целостности, когда в базе данных определены отношения "один ко многим". Установка ссылочной целостности позволяет поддерживать соответствие между главной и подчиненной таблицами. Для поддержания ссылочной целостности ограничения PRIMARY KEY используются в комбинации с ограничениями FOREIGN KEY, описанными ниже.

Ограничение FOREIGN KEY

Ограничение FOREIGN KEY (внешний ключ) гарантирует, что каждое значение, введенное в столбец, уже существует в некотором другом столбце (обычно, в другой таблице). Ограничения FOREIGN KEY обычно используются для поддержания ссылочной целостности, когда в базе данных определены отношения "один-ко-многим". Ограничения FOREIGN KEY всегда используются вместе с ограничениями PRIMARY KEY. В отношении "один-ко-многим" внешний ключ — столбец в подчиненной таблице, которая содержит идентификатор строки в главной таблице. Значение в столбце внешнего ключа равно значению в столбце первичного ключа в другой таблице.
В отношении "один-к-одному" каждая строка в подчиненной таблице соответствует уникальной строке в главной таблице. В отношении "один-ко-многим" одной строке в главной таблице может соответствовать любое количество строк в подчиненной таблице.

Ограничение CHECK

Гарантирует, что каждое значение, введенное в столбец, будет принадлежать некоему предопределенному множеству допустимых значений. Например, стипендия студента не может быть отрицательной, а значение курса, на котором обучается студент должно принадлежать множеству {1, 2, 3, 4, 5}.

```
ALTER TABLE dbo.STUDENTS ADD  CONSTRAINT CK_STUDENTS_COURSEDOMAIN CHECK ( COURSE IN (1, 2, 3, 4, 5))
GO
```

Ограничение CHECK не поддерживается MySQL.

Поддержка ссылочной целостности при помощи ограничений PRIMARY/FOREIGN KEY

При поддержании ссылочной целостности между главной и подчиненной таблицами используются следующие правила:

- подчиненная строка не может быть вставлена, пока не существует главная строка; однако в поле внешнего ключа возможен ввод пустых значений, показывающих, что записи только подготавливаются и пока не являются связанными;
- главная строка не может быть удалена до удаления всех подчиненных строк;
- если значение первичного ключа в главной строке изменено, все значения внешнего ключа, которые обращаются к этому значению первичного ключа, должны быть также обновлены, и,

наоборот, нельзя изменить значение ключевого поля в главной таблице, если существуют связанные записи.

Каскадное удаление и обновление

Если для связей, определена целостность данных, пользователь имеет возможность указать, следует ли автоматически выполнять для связанных записей операции каскадного обновления и каскадного удаления. Если включить данные параметры, станут возможными операции удаления и обновления, в противном случае запрещенные условиями целостности данных. Чтобы обеспечить целостность данных при удалении записей или изменении значения ключевого поля в главной таблице, автоматически вносятся необходимые изменения в связанные таблицы.

Пример:

```
alter table EXAM_MARKS add  constraint FK_EXAM_MARKS_STUDENT foreign key(student_id)
references STUDENTS (id)
-- on delete cascade


declare @newId int;
select @newId = max(id) + 1 from STUDENTS;

insert into STUDENTS(id,surname, name, gender, stipend, course, city, birthday ,univ_id)
values (@newId, 'Куприй', 'Владимир','m' ,650 ,2 ,'Черновцы' ,'19890723', 3);


insert into EXAM_MARKS (student_id, subj_id, mark ,exam_date)
values      (@newId, 2, 4, getdate()),
                (@newId, 3, 3, getdate()),
                (@newId, 4, 5, getdate());


select * from STUDENTS where id = @newId;
select * from EXAM_MARKS where student_id = @newId;

delete from STUDENTS where id = @newId;
select * from EXAM_MARKS where student_id = @newId;

delete from EXAM_MARKS where student_id = @newId;
delete from STUDENTS where id = @newId;

alter table EXAM_MARKS drop constraint FK_EXAM_MARKS_STUDENT;
```

Для MySQL

```
alter table EXAM_MARKS add  constraint FK_EXAM_MARKS_STUDENT foreign key(student_id)
references STUDENTS (id)
-- on delete cascade
;

select @newId := max(id) + 1 from STUDENTS;

insert into STUDENTS(id,surname, name, gender, stipend, course, city, birthday ,univ_id)
values (@newId, 'Куприй', 'Владимир','m' ,650 ,2 ,'Черновцы' ,'19890723', 3);


insert into EXAM_MARKS (student_id, subj_id, mark ,exam_date)
values      (@newId, 2, 4, current_date()),
                (@newId, 3, 3, current_date()),
                (@newId, 4, 5, current_date());
```

```
select * from STUDENTS where id = @newId;
select * from EXAM_MARKS where student_id = @newId;

delete from STUDENTS where id = @newId;
select * from EXAM_MARKS where student_id = @newId;

delete from EXAM_MARKS where student_id = @newId;
delete from STUDENTS where id = @newId;

alter table EXAM_MARKS drop foreign key FK_EXAM_MARKS_STUDENT;
```

Другие ограничения PRIMARY/FOREIGN KEY

При использовании команд модификации **UPDATE** и **DELETE** возможны следующие варианты действий:

- Ограничение **NO ACTION** или **RESTRICT** (по умолчанию). Любые попытки изменения значений родительского ключа запрещаются. Записи в родительской таблице не могут быть удалены, пока не удалены соответствующие записи в подчиненной таблице.
- Ограничение **CASCADE** - каскадное изменение/удаление. Изменения значений родительского ключа разрешаются, но при этом автоматически осуществляется коррекция всех значений внешних ключей, ссылающихся на модифицируемое значение родительского ключа. Удаление родительских записей приводит к удалению записей в подчиненной таблице.
- Ограничение **SET NULL.** Изменения значений родительского ключа разрешаются, но при этом соответствующие значения внешнего ключа автоматически удаляются, то есть заменяются значением NULL.

- Ограничение **SET DEFAULT.** Изменения значений родительского ключа разрешаются, но при этом соответствующие значения внешнего ключа автоматически заменяются значением по умолчанию.

Литература:

1. К. Дж. Дейт «Введение в системы баз данных»
2. Гектор Гарсиа-Молина, Джефри Д. Ульман, Дженифер Уидом «Системы баз данных. Полный курс»
3. Джо Селко «SQL для профессионалов. Программирование»
4. Майкл Дж. Хернандес, Джон Л. Вьескас «SQL-запросы для простых смертных»