# Coздание таблицы, ссылающейся на саму себя

Иногда возникает необходимость задать в ограничении столбец, находящийся не в другой таблице, а непосредственно в той же таблице, в которой создается ссылка на исходное ограничение. Это означает, что одна и та же таблица применительно к некоторому ограничению может играть роль и ссылающейся таблицы, и таблицы, указанной в ссылке. Разумеется, подобные ситуации встречаются не очень часто, но достаточно регулярно.

Рассмотрим пример таблицы сотрудников. Вот код создания таблицы:

CREATE TABLE Employees

(

"EmpioyeeID" serial PRIMARY KEY,

"FirstName" varchar (25),

"LastName" varchar (25),

"Title" varchar (25) ,

"SSN" varchar (11),

"Salary" money,

"PriorSalary" money,

"HireDate" date,

"ManagerEmpID" int,

"Department" varchar (25) NOT NULL

);

В данном случае рассматривается таблица, в которой будет иметься ссылка на столбец, представляющий собой столбец идентификации – ManagerEmpID (ИД менеджера ссылается на другого работника). Если мы добавим ограничение не во время создания, а при помощи оператора ALTER, то необходимо перед этим ввести в таблицу хотя бы одну первичную строку и только после этого задавать ограничение. Иначе мы получим ошибку:



Рисунок рпрп – Попытка ввода первой записи с установленной ссылкой

Уже после ввода первичной строки, можно приступить к заданию внешнего ключа. В этом варианте создания таблицы, ссылающейся на саму себя, в котором используется оператор ALTER, осуществляемые действия аналогичны тем, которые вы­полняются при уточнении любого другого определения внешнего ключа. Проверим действие следующего оператора на практике:

ALTER TABLE Employees

ADD CONSTRAINT "FK\_EmployeeHasManager"

FOREIGN KEY ("ManagerEmpID") REFERENCES Employees("EmpioyeeID")

Если бы речь шла о создании таблицы с самого начала, то на данном этапе можно было бы применить следующий сценарий:

CREATE TABLE Employees

(

"EmpioyeeID" serial PRIMARY KEY,

"FirstName" varchar (25),

"LastName" varchar (25),

"Title" varchar (25) ,

"SSN" varchar (11),

"Salary" money,

"PriorSalary" money,

"HireDate" date,

"TerminationDate" date,

"ManagerEmpID" int

REFERENCES Employees("EmpioyeeID"),

"Department" varchar (25) NOT NULL

)

## Ограничения UNIQUE

По су­ществу эти ограничения почти полностью соответствуют ограничениям первичного ключа, поскольку требуют наличия уникальных значений во всем указанном в них столбце таблицы. Ограничения UNIQUE часто называют ограничениями альтернативных ключей.Основные различия между ограничениями UNIQUE и ограничениями первичного ключа состоят в том, что в качестве ограниче­ний UNIQUE обычно не применяются уникальные идентификаторы строк таблицы, а, кроме того, на таблице может быть задано несколько ограничений UNIQUE. После того как будет задано ограничение UNIQUE, должно соблюдаться такое усло­вие, чтобы все значения в столбцах, указанных в этом ограничении, были уникальны­ми. При обнаружении попытки обновить или вставить строку со значением, которое уже имеется в столбце с ограничением уникальности, СУБД SQL Server вырабатывает сообщение об ошибке и отклоняет попытку ввести такую строку. В отличие от ограничения первичного ключа, применение ограничения UNIQUE не приводит к тому, что автоматически исключается возможность вставить в соответствующий столбец NULL – значение. В данном случае возможность применения NULL – значений зависит от того, как задана опция null для указанного столбца таблицы. Но следует учитывать, что даже если вставка NULL – значений разрешена, фактически допускается вставка в столбец толь­ко одного такого значения.

Создадим таблицу поставщиков и назовем ее Shippers, у разных поставщиков не может быть один и тот же номер телефона:

CREATE TABLE Shippers (

"ShipperID" serial PRIMARY KEY,

"ShipperName" varchar(30)

"Address" varchar(30)

"City" varchar(25)

"State" char(2)

"Zip" varchar(10)

"PhoneNo" varchar(14) UNIQUE

)

Как уже было сказано, ограничение UNIQUE действует в основном по таким же принципам, как ограничения первичных и внешних ключей. Рассмотрим следующий пример создания ограничения UNIQUE на таблице Employees:

ALTER TABLE Employees

ADD CONSTRAINT "AK\_EmployeeSSN" UNIQUE ("SSN");

Отметим, что в приведенном выше примере аббревиатура АК, используемая в имени ограни­чения, расшифровывается как Alternate Key (альтернативный ключ). С другой стороны, для обозначения первичных ключей и внешних ключей применяются аббревиатуры РК (Primary Key) и FK (Foreign Key). А в именах ограничений UNIQUE часто используется префикс UQ, или просто U.

## Ограничения CHECK

Ключевое слово CHECK задает ограничение для диапазона значений, которые могут храниться в столбце. Для этого после слова CHECK указывается в скобках условие, которому должен соответствовать столбец или несколько столбцов. Например, зарплата сотрудников не может быть меньше 0 или больше 200000$, а дата увольнения ранее даты приема на работу:

CREATE TABLE Employees

(

"EmpioyeeID" serial PRIMARY KEY,

"FirstName" varchar (25),

"LastName" varchar (25),

"Title" varchar (25) ,

"SSN" varchar (11),

"Salary" money CHECK("Salary" > '0$' AND "Salary" < '200000$'),

"PriorSalary" money CHECK("PriorSalary" > '0$' AND "PriorSalary" < '230000$'),

"HireDate" date,

"TerminationDate" date CHECK("TerminationDate" > "HireDate" OR "TerminationDate" = NULL),

"ManagerEmpID" int

REFERENCES Employees("EmpioyeeID"),

"Department" varchar (25) NOT NULL

)

Удобным свойством ограничений CHECK является то, что эти ограничения не обя­зательно должны применяться только к какому-то конкретному столбцу. Безусловно, указанные ограничения могут относиться лишь к некоторому столбцу, но также до­пускается их распространение по существу на всю таблицу, в том смысле, что с их помощью может осуществляться проверка значений в одном столбце на основании значений другого столбца. С помощью ограничений CHECK может также осуществляться проверка того, соответствует ли некоторое сочетание значений столбцов заданному критерию. Ограничения CHECK определяются на основании таких же правил, которые рас­пространяются на операции проверки, используемые в конструкции WHERE. Примеры критериев, которые могут применяться в ограничении CHECK, приведены в таблице №4.

Таблица №fddf – Примеры применения ограничения CHECK

|  |  |
| --- | --- |
| **Назначение** | **Код SQL** |
| Обеспечение применения в столбце Month только допустимых значений | between 1 and 12 |
| Правильное форматирование номера карточки социального обеспечения | Like ‘ [0-9] [0-9] [0-9]-[0-9] [0-9]-[0-9] [0-9] ‘ |
| Регламентация списка допустимых значений в поле Shippers | in (‘UPS’, ‘Fed Еx’, ‘USPS’) |
| Регламентация применения только положительных значений цены | UnitPrice >= О |
| Ссылка на другое поле в той же строке | ShipDate >= OrderDate |

В таблице приведена лишь незначительная часть всех возмож­ных примеров, поскольку количество вариантов применения различных операций сравнения в конструкции CHECK является практически бесконечным. В ограничении CHECK может быть задано почти любое такое же выражение, которое допускается за­давать в конструкции WHERE. К тому же ограничения CHECK позволяют достичь гораз­до более высокой производительности по сравнению с альтернативными средствами проверки допустимости данных.