عنوان: **گذری بر مفاهیم relationship**
نویسنده: محمد سلیم ابادی
تاریخ: ۱۳۹۱/۱۱/۰۶ ۲۲:۵
آدرس: www.dotnettips.info
گروه‌ها: Referential Integrity, foreign key, relationship, table, SQL Server

متاسفانه کاربران زیادی وجود دارند که هنوز درک صحیحی از جامعیت داده‌های ارجاعی (referential Integrity) ندارند. نمی‌دانند که relationship چیزی جز قید کلید خارجی (foreign key) نیست. در ادامه مفاهیم زیر را در حد آشنایی توضیح خواهم داد:

کلید خارجی ترکیبی (composite foreign key)

خود ارجاعی (self referencing)

اعمال تغییرات به صورت آبشاری (cascade)

چندین مسیر برای اعمال (multiple cascading path)

جدول اتصال (junction table)- ارتباط یک به یک

توسط دستور create table به دو شکل می‌توانیم بر روی ستون‌ها قید (کلید اولیه، check، کلید خارجی، کلید یونیک...) تعریف نمود:

قید ستونی

قید جدولی

syntax مربوط به قید کلید خارجی در مدل ستونی به صورت زیر است:

```
<column_constraint> ::=
[ CONSTRAINT constraint_name ]
{    ...

  | [ FOREIGN KEY ]
       REFERENCES [ schema_name . ] referenced_table_name [ ( ref_column ) ]
       [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
       [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
       [ NOT FOR REPLICATION ]

  ...
}
```

نکته: بطور پیش فرض برای کلید خارجی اعمال update و delete روی وضعیت no action تنظیم شده است. به این معنا که اگر سعی کنیم کلید اولیه جدول مرجع را بروز رسانی یا حذف کنیم ممانعت به عمل خواهد آمد. برای رفع این مشکل هم میتوانید از طریق design اقدام کنید و هم در هنگام ساخت جدول توسط DDL (همانطور که در دستورات فوق مشاهده میشود).

کلید خارجی ترکیبی

زمانی که در جدول والد (parent) کلید اولیه ترکیبی باشد، هر جدولی که بخواهد به کلید جدول والد ارجاعی داشته باشد باید از ترکیب دو ستون برای ساخت کلید خارجی استفاده کند.

فرض کنید جدول parent به این صورت است (ترکیب دو ستون col1 و col2 کلید اولیه است)

```
create table parent
(
col1 int not null,
col2 int not null,
col3 char(1) null,
-- Composite Primary Key
primary key(col1, col2)
);
```

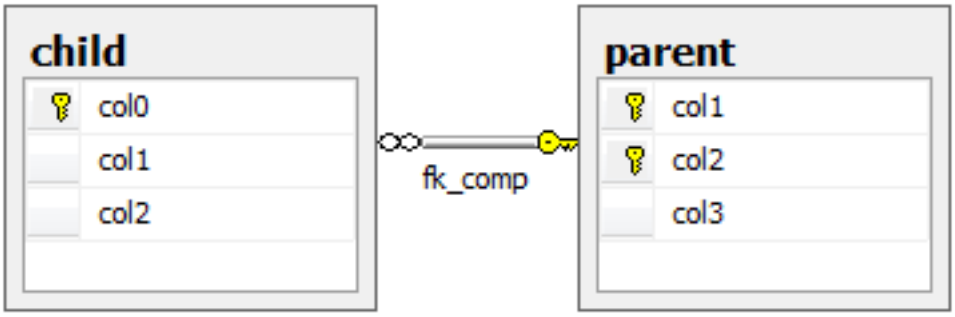
در اینجا چون ترکیب دو ستون کلید اولیه هست باید توسط "قید جدولی" اقدام به تعریف کلید کرد

و جدول child که دارای قید کلید خارجی ترکیبی به نام fk_comp است و به جدول parent ارجاع داده است:

```
create table child
(
col0 int primary key,
col1 int null,
col2 int null,
-- Composite Foreing Key Constraint
constraint fk_comp
foreign key (col1, col2)
references parent(col1, col2)
);
```

در این DDL هم از قید جدولی برای تعریف کلید خارجی ترکیبی استفاده شده است.

نمودار این دو جدول:



پس به عنوان نتیجه گیری، هرگاه جدول اصلی دارای کلید ترکیبی بود در جداول child نیز باید از کلید خارجی ترکیبی برای ایجاد relationship استفاده نمود.

اما این دو جدول را به یک شیوه دیگر نیز می‌توان طراحی نمود. در جدول parent ترکیب دو ستون col1 و col2 را منحصربفرد (unique) گرفته و ستونی دیگر (مثلا از نوع identity) را به عنوان کلید اولیه در نظر گرفت (یا یک ستون از نوع محاسباتی تعریف کرده و آن را کلید قرار داد)

```
create table parent
(
col0 int not null primary key identity,
col1 int not null,
col2 int not null,
col3 char(1) null,
-- Composite Unique Key
unique(col1, col2)
);

create table child
(
col0 int primary key,
col1 int null references parent
);
```

خود ارجاعی و multiple cascading path

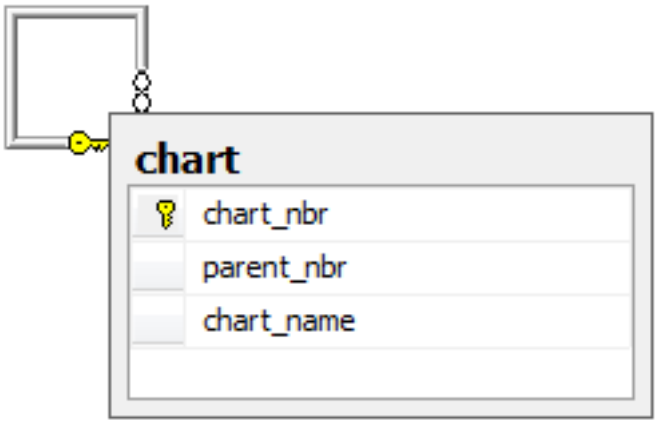
فرض کنید بخش‌های مختلف یک سازمان که بصورت چارت است را توسط جدول پیاده سازی کردیم. ستون‌های جدول به این شرح هستند:

کد بخش
نام بخش
کد بخش بالایی

ستون "کد بخش بالایی" نیز خود یک بخش است. برای پیاده سازی این چنین ساختارهایی از جدول زیر کمک گرفته می‌شود:

```
create table chart
(
chart_nbr int not null primary key,
parent_nbr int null references chart,
chart_name varchar(5) null
);
```

تصویر نمودار جدول chart



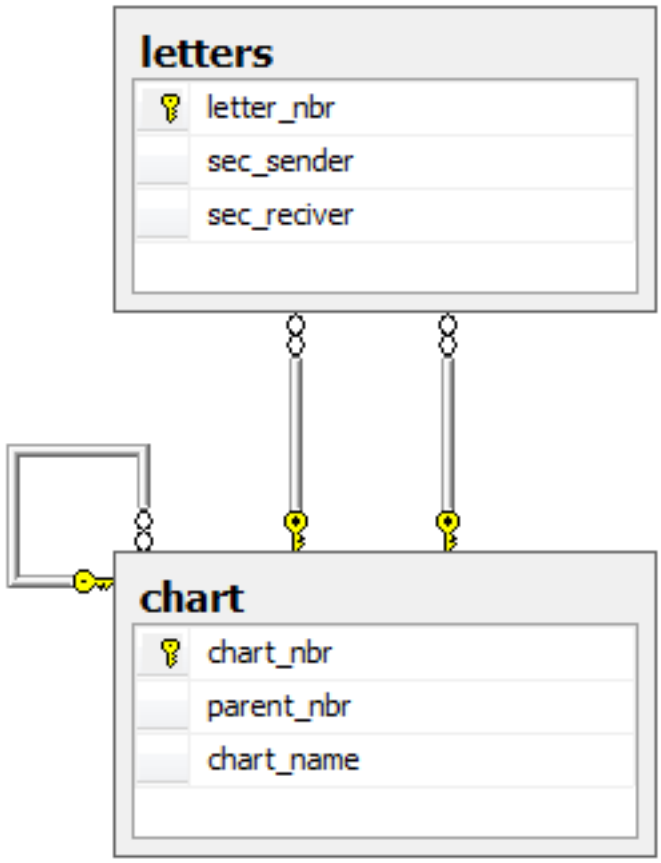
حالا فرض کنید میخواهیم اطلاعات نامه هایی که بین بخش‌ها رد و بدل میشود را در یک جدول ذخیره کنیم. جدول دارای ستون‌های زیر خواهد بود:

شماره نامه
کد بخش فرستنده
کد بخش گیرنده

ستون شماره نامه کلید اولیه و دو ستون دیگه کلیدهای خارجی هستند که به جدول chart مراجعه می‌کنند:

```
create table letters
(
letter_nbr int primary key,
sec_sender int not null references chart,
sec_reciver int not null references chart
);
```

نمودار جدول نامه‌ها و چارت:



نکته ای که در اینجا وجود دارد این است که اگر کلید جدول chart بروز شود آنگاه SQL Server از دو راه می‌تواند جدول letters را بروز رسانی کند، به این علت پیغام خطایی با عنوان multiple cascading paths صادر می‌شود. برای رفع این مشکل باید از trigger کمک گرفت.

جدول اتصال (junction table)

برای پیاده سازی رابطه N-N از جدول واسط کمک گرفته می‌شود. برای این منظور رابطه N-N را باید به دو رابطه N-1 تجزیه کرد.

فرض کنید یک جدول مربوط به خلبانان و جدول دیگر مربوط به مسیرهای پروازی (مثل مسیر ایران-ترکیه، ایران-عربستان...) است. یک خلبان ممکن است در چند مسیر پروازی هواپیما را هدایت کرده باشد و یا بالعکس یک مسیر پروازی ممکن است توسط N خلبان طی شده باشد.

برای پیاده سازی اینگونه سیستم هایی باید یک جدول ایجاد نمود که دارای دو کلید خارجی باشد یکی آنها به جدول خلبانان و دیگری به مسیرهای پروازی مرتبط است.

می‌توان ترکیب دو کلید خارجی جدول واسط را کلید اولیه در نظر گرفت.

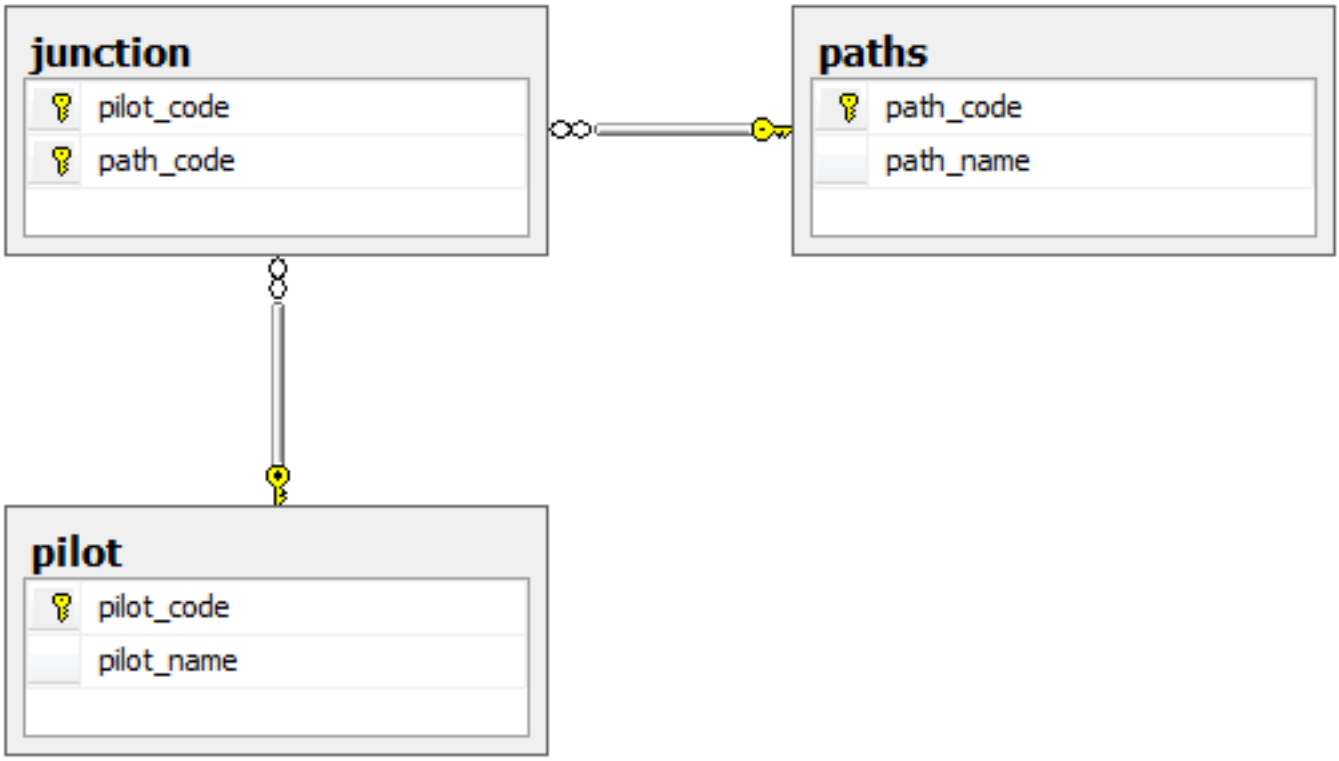
پس خواهیم داشت:

```
create table pilot
(
pilot_code int primary key,
pilot_name varchar(20)
);

create table paths
(
path_code int primary key,
path_name varchar(20)
);
```

```
create table junction
(
pilot_code int references pilot,
path_code int references paths,
primary key (pilot_code, path_code)
);
```

و نمودار آن:



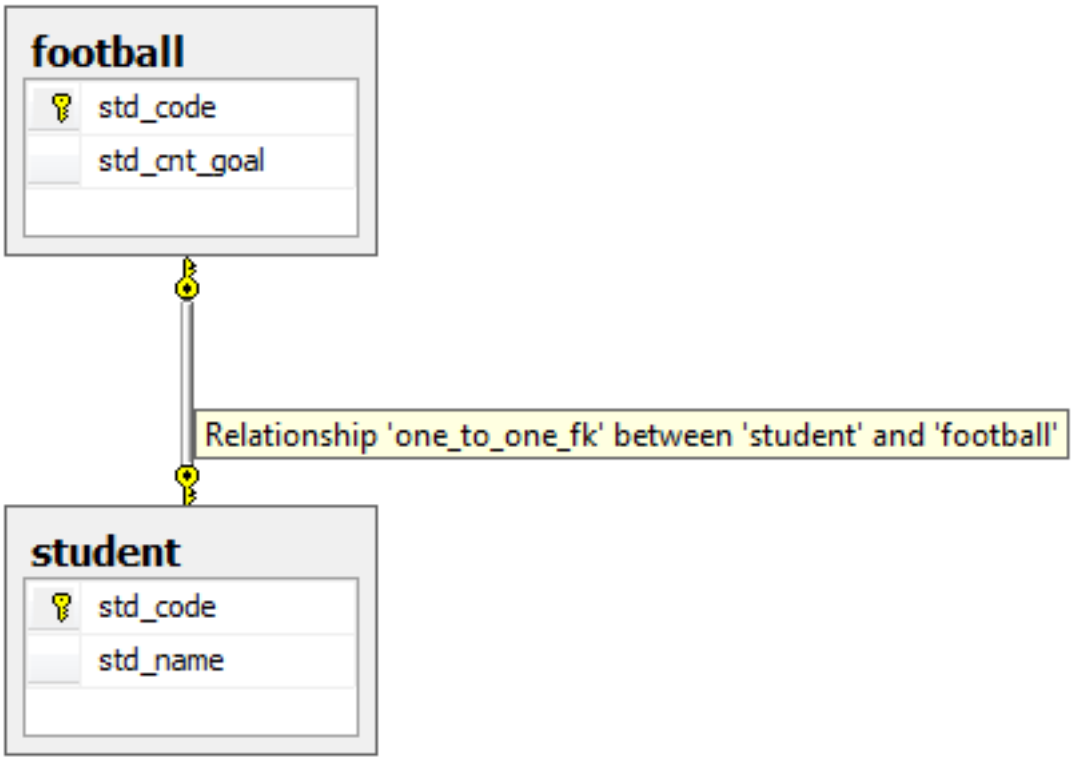
رابطه یک به یک

زمانی که نمونه‌های محدودی از یک موجودیت دارای مقدار برای یکسری خصیصه هستند بهتر است جدول به دو جدول تجزیه شود تا فضای اضافی صرف جدول نشود. مثلا در مدرسه تنها 10 درصد دانش آموزان جزء تیم فوتبال هستند حال اگر بخواهیم اطلاعات مربوط به تیم فوتبال مثل تعداد گل زده، تعداد بازی ... در جدول اصلی ذخیره کنیم برای 90 درصد دانش آموزان مقداری نخواهیم داشت. برای حل این مساله ارتباط یک به یک پیشنهاد می‌شود.

```
create table student
(
std_code int primary key,
std_name varchar(25) not null
);

create table football
(
std_code int primary key
  constraint one_to_one_fk
  references student,
std_cnt_goal int not null
  default (0)
);
```

توجه داشته باشید که ستون std_code هم کلید اولیه هست و هم کلید خارجی که به جدول student ارجاع داده شده است.



نتیجه گیری

یک ستون همزمان می‌تواند کلید اولیه باشد و هم کلید خارجی (مثلا در ارتباط یک به یک)

همانطور که کلید اولیه ترکیبی داریم به همان شکل هم کلید خارجی ترکیبی داریم.

یک جدول می‌تواند به خودش ارجاع دهد که به آن اصطلاحا self-referencing می‌گویند

relationship چیزی جز کلید خارجی نیست و کلید خارجی نیز چیزی جز یک قید برای جامعیت داده‌ها نیست

جامعیت داده ارجاعی را می‌توان توسط trigger پیاده سازی کرد

اگر SQL Server بیش از یک مسیر برای تغییر جدول child داشته باشد با مشکل مواجه خواهید شد

نظرات خوانندگان

نویسنده: Mohsen
تاریخ: ۱۳۹۱/۱۱/۰۷ ۸:۱۵

بسیار عالی بود.مفاهیم پایگاه داده برای جامعه نرم افزار ایران بسیار حیاتی هستند.

نویسنده: mohammad
تاریخ: ۱۳۹۱/۱۱/۰۸ ۲۰:۴۰

با سلام.ممنون از مطلبتون. لطفا توضیحی راجع به اینکه چطور میتوان با دادن یک id تمام ریشه‌های اون رو درحالت خودارجاعی درآورد بدید.

نویسنده: mohammad
تاریخ: ۱۳۹۱/۱۱/۰۸ ۲۰:۵۵

ببخشید منظورم برگ‌ها بود که اشتباه نوشتم ریشه. با تشکر

نویسنده: محمد سلم ابادی
تاریخ: ۱۳۹۱/۱۱/۰۸ ۲۱:۵۶

سلام،
برای این منظور باید از recursive cte کمک گرفت.
فرض کنید درختی به شکل زیر داریم:

```
/*         A
         /   \
        B     C
        |    /|\
        D   E F G
        |
        H

*/
```

و هدف بدست آوردن تمام زیر شاخته‌های گره A است.
ابتدا باید تمام گره هایی که مقدار گره پدرشان برابر با A است را بدست بیاریم یعنی گره‌های B و C
حالا باید تمام گره هایی که گره پدرشان B و یا C است را بدست بیاریم یعنی گره‌های D E F G
و در مرحله بعد باید تمام گره هایی را بدست بیاریم که گره پدرشان برابر با یکی از مقادیر بدست آمده در مرحله قبل (یعنی D E F G) یعنی H

این الگوریتم را توسط Recursive CTE پیاده میکنیم:

```
declare @t table
(
   id char(1) primary key not null,
   pid char(1) null --references @t
);

insert @t
values ('A', null), ('B','A'),('C','A'),
('D','B'), ('H','D'),('E','C'),('F','C'),('G','C');

with cte as
(
select id
from @t
where pid = 'A'
```

```
union all

select t.id
from cte c
join @t t
on t.pid = c.id
)
select * from cte
```

موفق باشید

نویسنده: mohammad
تاریخ: ۱۳۹۱/۱۱/۰۹ ۱۰:۵۳

ممنون خیلی هم خوب.
اما یک سوال؟ آیا این روش فشار زیادی روی سرور ایجاد نمیکنه. مثلا در یک برنامه تحت وب با تعداد زیاد کاربر؟

نویسنده: محمد سلم ابادی
تاریخ: ۱۳۹۱/۱۱/۰۹ ۱۲:۱۵

این یک روش معمولی و رایج هست که همگان دارن ازش استفاده می‌کنند. بعید می‌دانم که در عملکرد مشکلی داشته باشد.
اگر فرضا سرعت اجرای query اتان پایین بود یک اندیس (index) روی ستون "کد والد" تعریف کنید
به این شکل

```
create nonclustered index ix_parent on table_name (parent_id)
```