İSİM

CREATE CAST – iki veri türü arasında yeni bir dönüşüm tanımlar

KULLANIM

```
CREATE CAST (kaynak_tür AS hedef_tür)
    WITH FUNCTION işlev_ismi (arg_türü)
    [ AS ASSIGNMENT | AS IMPLICIT ]

CREATE CAST (kaynak_tür AS hedef_tür)
    WITHOUT FUNCTION
    [ AS ASSIGNMENT | AS IMPLICIT ]
```

Açıklama

CREATE CAST iki veri türü arasında yeni bir dönüşüm tanımlar. Örnek:

```
SELECT CAST(42 AS text);
```

Burada 42 tamsayı sabiti evvelce tanımlanmış bir işlev (`text(int4)` işlevi) çağrılarak `text` türüne dönüştürülmektedir. (Eğer böyle bir tür dönüşümü tanımlanmamışsa, dönüşüm başarısız olur.)

İki tür *ikilik olarak uyumlu* olabilir, yani herhangi bir işlev çağrılmadan serbestçe bir türden diğer türe dönüştürülebilir. Bu ilgili değerlerin aynı dahili gösterime sahip olmalarını gerektirir. Örneğin, `text` ve `varchar` türleri ikilik olarak uyumlu türlerdir (ya da dahili gösterimleri aynı olan türlerdir).

Öntanımlı olarak, bir tür dönüşümü sadece **CAST**(*x* **AS** *tür_ismi*) veya *x*::*tür_ismi* gibi doğrudan bir tür dönüşüm isteği ile yapılabilir.

Eğer tür dönüşümü **AS ASSIGNMENT** ile tanımlanmışsa, bir sütuna hedef veri türünde atama şeklinde örtük olarak çağrılabilir. Örneğin, `foo.f1`'nin `text` türünde bir sütun olduğunu varsayalım,

```
INSERT INTO foo (f1) VALUES (42);
```

eğer `integer` türünden `text` türüne dönüşüm **AS ASSIGNMENT** imli olarak oluşturulmuşsa, bu dönüşüm mümkün olur, yoksa olmaz. (Böyle tür dönüşümlerine genellikle, ***atamalı tür dönüşümü*** diyoruz.)

Eğer tür dönüşümü **AS IMPLICIT** ile oluşturulmuşsa, herhangi bir bağlam içinde, sadece atama ile değil, bir ifadenin içinde kullanıldığında bile dolaylı olarak böyle bir tür dönüşümü yapılabilir. Örneğin, `||` işleci `text` türünde terimler aldığından,

```
SELECT 'The time is ' || now();
```

eğer `timestamp` türünden `text` türüne dönüşüm **AS IMPLICIT** imli olarak oluşturulmuşsa, bu dönüşüm mümkün olur. Aksi takdirde tür dönüşümünün doğrudan yazılması gerekir. Örnek:

```
SELECT 'The time is ' || CAST(now() AS text);
```

(Böyle tür dönüşümlerine genellikle, ***dolaylı tür dönüşümü*** diyoruz.)

Tür dönüşümlerini dolaylı olarak yapılması için imlerken biraz tutucu olmakta fayda vardır. Dolaylı dönüşümde bir aşırı bolluk, çok sayıda yorumun mümkün olması nedeniyle, PostgreSQL'in şaşırtıcı komut yorumları seçmesine ya da komutları tamamen çözümleyememesine sebep olabilir. En iyisi sadece aynı genel tür kategorisindeki türler arasında bilgi koruyan dönüşümler için dolaylı olarak çağrılabilen bir tür dönüşümü yapmaktır. Örneğin, `int2`'den `int4`'e tür dönüşümünün dolaylı olması kabul edilebilir, fakat

`float8`'den `int4`'e dönüşüm şüphesiz atama yoluyla yapılmalıdır. `text`'den `int4`'e tür dönüşümü gibi farklı tür kategorileri arasındaki tür dönüşümünü açıkça belirterek yapmak en iyisidir.

Bir tür dönüşümünün mümkün olması için kendi kaynak ve hedef veri türleriniz olmalıdır. İkilik olarak uyumlu tür dönüşümü oluşturabilmek için ise ayrıcalıklı kullanıcı olmalısınız. (Bu kısıtlamanın sebebi, hatalı yapılan ikilik uyumlu tür dönüşümlerinin sunucunun kolayca çökmesine sebep olmasıdır.)

Parametreler

kaynak_tür
: Tür dönüşümünün kaynak veri türünün ismi.

hedef_tür
: Tür dönüşümünün hedef veri türünün ismi.

işlev_ismi(*arg_türü*)
: Tür dönüşümünü gerçekleştirecek işlev. İşlevin ismi şema nitelemeli olabilir; değilse, işlev şema arama yolunda aranacaktır. İşlevin sonuç veri türü tür dönüşümünün hedef veri türü ile eşleşmelidir. Argümanlar aşağıda açıklanmıştır.

`WITHOUT FUNCTION`
: Kaynak ve hedef türlerin ikilik olarak uyumlu olduğunu belirtir, böylece tür dönüşümünü gerçekleştirmek için bir işleve gerek kalmaz.

`AS ASSIGNMENT`
: Tür dönüşümünün atama bağlamında dolaylı olarak uygulanabileceğini belirtir.

`AS IMPLICIT`
: Tür dönüşümünün herhangi bir bağlamda dolaylı olarak uygulanabileceğini belirtir.

Tür dönüşümü gerçekleme işlevleri bir, iki ya da üç argümanlı olabilir. İlk argümanın türü tür dönüşümünün kaynak veri türü ile aynı olmalıdır. Varsa, ikinci argüman `integer` türünde olmalıdır; hedef türle ilgili tür dönüştürücüyü, yoksa `-1` değerini alır. Varsa, üçüncü argüman `boolean` türünde olmalıdır; dönüşüm açıkça belirtilerek uygulanacaksa `true`, aksi takdirde `false` değerini alır. (Tuhaf bir biçimde, SQL belirtimi bazı durumlarda, doğrudan ve dolaylı tür dönüşümleri için farklı davranışlar talep eder. Bu argüman böyle tür dönüşümlerini gerçekleştirmesi istenen işlevler için sağlanmıştır. Kendi veri türlerinizi buna konu olacak şekilde tasarlamanız tavsiye edilmez.)

Normal olarak bir tür dönüşümü farklı kaynak ve hedef veri türlerine sahip olmalıdır. Yine de, dönüşümü gerçekleştirecek işlevin birden fazla argümanı olması durumunda, kaynak ve hedef veri türleri aynı olan tür dönüşümü bildirimlerine izin verilir. Bu, sistem kataloglarında türe özel uzunluk zorlama işlevlerini ifade etmekte kullanılır. İsimli işlev, tür değerini ikinci argümanında belirtilen tür değiştirici değerine zorlamak için kullanılır. (Dil kuralları sadece belli yerleşik veri türlerinin tür değiştiricilere sahip olmasına izin verdiğinden, bu özellik kullanıcı tanımlı hedef türlerde kullanmak için değildir, burada bütünlüğü sağlamak için bahsettik.)

Bir tür dönüşümü farklı kaynak ve hedef türleri ile birden fazla argüman alan bir işleve sahip olduğunda, bir türden diğer türe dönüşümü yaparken aynı adımda bir uzunluk zorlaması da yapılıyor demektir. Böyle bir girdi mevcut olmadığı zaman, bir tür dönüşümü kullanan bir tür zorlaması iki adım gerektirir; birincisinde veri türleri arasında dönüşüm yapılır, ikincisinde tür değiştirici uygulanır.

Ek Bilgi

Kullanıcı tanımlı bir tür dönüşümünü kaldırmak için **`DROP CAST`** [`drop_cast(7)`] kullanılır.

Türler arasındaki dönüşümün her iki yönde de yapılmasını istiyorsanız, her iki tür dönüşümünü açıkça bildirmeniz gerektiğini unutmayın.

PostgreSQL 7.3 öncesinde, her işlev bir veri türü ile aynı ismi alır, bu veri türünü döndürür ve farklı türde bir argüman alıp özdevinimli olarak bir tür dönüşüm işlevi olurdu. Bu uzlaşım şemaların tanıtımıyla ve sistem kataloglarında ikilik uyumlu tür dönüşümlerinin ifade edilebilmesi için terkedildi. Yerleşik tür dönüşüm işlevleri hala bu isimleme şemasını kullanıyor olsa da, pg_cast sistem kataloğunda da tür dönüşümleri olarak gösterilmiş olması gerekir.

Gerekli olmadığında, tür dönüşüm gerçekleme işlevlerinin hedef veri türünde isimlendirilmesi uzlaşımını kullanmaya devam etmenizi öneririz. Çoğu kullanıcı, *türismi*(*x*) biçiminde veri türlerini dönüştürmeye alışmıştır. Bu yazım şekli aslında bir tür dönüşümü gerçekleştirme işlevi çağrısından ne eksik ne de fazladır; bir tür dönüşümü olarak özellikle ele alınmaz. Eğer sizin dönüşüm işlevleriniz bu uzlaşıma uygun olarak isimlendirilmemişse, siz şaşırtıcı kullanıcılarsınız demektir. PostgreSQL farklı argüman türlerine sahip işlevlere aynı ismin verilmesine izin verdiğinden farklı veri türlerine dönüşüm için hepsi hedef veri türünde isimlendirilmiş çok sayıda dönüşüm işlevine sahip olmanın bir zorluğu yoktur.

Bilgi

Önceki paragrafta küçük bir zararsız yalan vardır: pg_cast'ın bir görünüşte işlev çağrısının anlamını çözümlemekte kullanılacağı bir durum hala vardır. Eğer *isim*(*x*) gibi bir işlev çağrısı aslında bir işlevle eşleşmiyorsa, fakat *isim* bir veri türünün ismiyse ve pg_cast *x* türünden bu türe bir ikilik uyumlu tür dönüşümünü gösteriyorsa, çağrı bir açıkça tür dönüşümü olarak yorumlanacaktır. İkilik olarak uyumlu tür dönüşümleri bir işleve karşılık olmasalar bile işlevsel sözdizimi kullanılarak çağrılabileceğinden böyle bir olağandışılık vardır.

Örnekler

int4(text) işlevini kullanarak text türünden int4 türüne bir tür dönüşümü oluşturmak için:

```
CREATE CAST (text AS int4) WITH FUNCTION int4(text);
```

(Bu tür dönüşümü sistemde zaten önceden tanımlanmıştır.)

Uyumluluk

SQL:1999'un ikilik uyumlu türler veya gerçekleme işlevlerinin ek argümanlarını hesaba katmaması dışında, **CREATE CAST** SQL:1999 ile uyumludur. **AS IMPLICIT** de bir PostgreSQL oluşumudur.

See Also

CREATE FUNCTION [create_function(7)], **CREATE TYPE** [create_type(7)], **DROP CAST** [drop_cast(7)].

Çeviren

Nilgün Belma Bugüner <nilgun (at) belgeler·gen·tr>, Nisan 2005

YASAL UYARI

Bu dosya (man7-create_cast.pdf), belgenin XML biçiminin TEXLive ve belgeler-xsl paketlerindeki araçlar kullanılarak PDF biçimine dönüştürülmesiyle elde edilmiştir.

31 Ocak 2007