İSİM

CREATE TABLE – yeni bir tablo tanımlar

KULLANIM

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE tablo_ismi (
  { sütun_ismi veri_türü [ DEFAULT öntanımlı_ifade ]
    [ sütun_kısıtı [ ... ] ]
    | tablo_kısıtı
    | LIKE ana_tablo [ { INCLUDING | EXCLUDING } DEFAULTS ] }  [, ... ]
)
[ INHERITS ( ana_tablo [, ... ] ) ]
[ WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tabloalanı ]

Buradaki sütun_kısıtı:

[ CONSTRAINT kısıt_ismi ]
{ NOT NULL |
  NULL |
  UNIQUE [ USING INDEX TABLESPACE tabloalanı ]
  | PRIMARY KEY [ USING INDEX TABLESPACE tabloalanı ]
  | CHECK (ifade)
  | REFERENCES başvuru_tablosu [ ( başvuru_sütunu ) ]
    [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]
    [ ON DELETE eylem ] [ ON UPDATE eylem ] }
[ DEFERRABLE | NOT DEFERRABLE ]
[ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

ve tablo_kısıtı:

[ CONSTRAINT kısıt_ismi ]
{ UNIQUE ( sütun_ismi [, ... ] )
   [ USING INDEX TABLESPACE tabloalanı ]
   | PRIMARY KEY ( sütun_ismi [, ... ] )
      [ USING INDEX TABLESPACE tabloalanı ]
      | CHECK ( ifade )
      | FOREIGN KEY ( sütun_ismi [, ... ] )
         REFERENCES başvuru_tablosu [ ( başvuru_sütunu [, ... ] ) ]
         [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]
         [ ON DELETE eylem ] [ ON UPDATE eylem ] }
[ DEFERRABLE | NOT DEFERRABLE ]
[ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

Açıklama

CREATE TABLE o anki veritabanınında yeni bir tabloyu ilk olarak boş bir tablo olarak oluşturacaktır. Tablonun sahibi komutu çalıştıran kullanıcı olacaktır.

Eğer bir şema ismi belirtilmişse (**CREATE TABLE** *myschema.mytable ...* gibi), tablo belirtilen şemada oluşturulur. Aksi takdirde o an geçerli olan şemada oluşturulur. Geçici tablolar bir özel şema içinde mev-

cuttur, dolayısıyla bir geçici tablo oluştururken bir şema ismi belirtilmeyebilir. Tablo ismi şema içindeki diğer tabloların, indekslerin ve sanal tabloların isimlerinden farklı olmalıdır.

CREATE TABLE ayrıca, özdevinimli olarak tablonun bir satırına karşılık olarak karma bir veri türü oluşturur. Bu nedenle, aynı şema içindeki tablolar mevcut veri türleriyle aynı isme sahip olamazlar.

İsteğe bağlı kısıt deyimleri ile yeni veya güncellenen satırların yerleştirme ve güncelleme işlemlerinin sağlaması gereken kısıtlar (sınamalar) belirtilir. Bir kısıt, tablo içinde çeşitli yollarla geçerli değerlerin tanımlanmasına yardımcı olacak bir SQL nesnesidir.

Kısıtları tanımlamanın iki yolu vardır: tablo kısıtları ve sütun kısıtları. Bir sütun kısıtı bir sütun tanımının parçası olarak tanımlanır ve birden fazla sütunu kuşatabilir. Her sütun kısıtı ayrıca, bir tablo kısıtı olarak da yazılabilir; bir kısıt sadece bir sütunu etkilediğinde bir sütun kısıtı sayılabilir.

Parametreler

TEMPORARY veya **TEMP**

Belirtilmişse tablo bir geçici tablo olarak oluşturulur. Geçici tablolar bir oturumun sonunda özdevinimli olarak silinir, bazan isteğe bağlı olarak bir hareketin sonunda da silinebilir (aşağıdaki **ON COMMIT** açıklamasına bakınız). şema nitelemeli ismlerle başvurulmadıkça, aynı isimle varolan kalıcı tablolar geçici tablonun varlığı durumunda o anki oturuma görünür olmazlar. Bir geçici tablo üzerinde oluşturulan indeksler de özdevinimli olarak geçici olacaktır.

İsteğe bağlı olarak, **TEMPORARY** veya **TEMP**'den önce **GLOBAL** veya **LOCAL** yazılabilir. Bu, PostgreSQL'de bir şey farkettirmez, ama yine de *Uyumluluk* (sayfa: 9) bölümüne bakın.

tablo_ismi

Oluşturulacak tablonun ismi (şema nitelemeli olabilir).

sütun_ismi

Yeni tabloda oluşturulacak bir sütunun ismi.

veri_türü

Sütunun veri türü. Bu, dizi belirteçleri içerebilir. PostgreSQL tarafından desteklenen veri türleri hakkında bilgi edinmek için http://www.postgresql.org/docs/8.0/static/datatype.html adresine bakınız.

DEFAULT *öntanımlı_ifade*

DEFAULT deyimi sütun tanımının içinde görünerek sütun için bir öntanımlı veri değeri atar. Değer, bir değişken içrmeyen bir ifadedir (tablo içinde diğer sütunlara çağraz başvurulara ve altsorgulara izin verilmez). Öntanımlı ifadenin veri türü sütunun veri türü ile eşleşmelidir.

Önanımlı ifade, sütun için bir değer belirtmeyen bir veri girme işleminde sütun değeri olarak kullanılacaktır. Bir sütun için öntanımlı bir değer belirtilmezse, NULL öntanımlıdır.

INHERITS (*ana_tablo* [, ...])

İsteğe bağlı **INHERITS** deyimi, yeni bir tablonun tüm sütunlarını miras alacağı tabloların bir listesini belirtmek için kullanılır.

INHERITS kullanımı yeni çocuk tablo ile onun abeveyni olan tablo arasında kalıcı bir birliktelik oluşturur. Ebeveynler üzerindeki şema değişiklikleri normalde çocukları da etkiler ve öntanımlı olarak çocuk tablonun verisi ebeveynler tarafından paylaşılır.

Eğer aynı sütun ismi birden fazla ebeveyn tabloda mevcutsa ve bu tabloların bu sütunları veri türü bakımından eşleşmediği takdirde bir hata raporlanacaktır. Eğer böyle bir durum yoksa, yinelenen sütunlar yani tabloda tek bir sütun oluşturacak şekilde katıştırılırlar. Eğer yeni tablonun sütun isimleri listesi aynı zamanda miras alınmış bir sütun da içeriyorsa, veri türü miras alınan sütunlara benzer şekilde eşleşmeli ve sütun tanımları tek bir sütun olarak katıştırılmalıdır. Ancak, miras alınan ve yeni sütun bildirimlerinin eşdeğer kısıtla belirtmesi gerekmez: bildirimlerden toplanan tüm kısıtlar birlikte katıştırılır ve tümü yeni tabloya uygulanır. Eğer yeni tablo sütun için açıkça bir öntanımlı değer belirtiyorsa, bu öntanımlı değer, miras alınan sütun bildirimlerin gelen öntanımlıların yerine geçer. Aksi takdirde, sütun için öntanımlı değer belirten her ebeveyn aynı öntanımlı değeri belirtmelidir, yoksa bir hata raporlanacaktır.

LIKE *ana_tablo* [{ **INCLUDING** | **EXCLUDING** } **DEFAULTS**]

LIKE deyimi, yeni bir tabloya veri türleri ve boş olmayan kısıtları ile birlikte tüm sütun isimlerinin özdevinimli kopyalanacağı bir tablo belirtir.

INHERITS'in tersine, yeni tablo ve özgün tablo oluşturma işleminden sonra tamamen birbirlerinin kopyası olurlar. Özgün tabloda yapılacak değişiklikler yeni tabloya uygulanmayacak ve yeni tabloda özgün tablodan alınmış veri olmayacaktır.

Kopyalanan sütun tanımları için öntanımlı olan ifadeler sadece **INCLUDING DEFAULTS** belirtilmişse kopyalanacaktır. Öntanımlı davranış öntanımlı olan ifadelerin dışlanması ve sonuç olarak yeni tablonun tüm sütunlarının boş öntanımlılara sahip olmasıdır.

WITH OIDS
WITHOUT OIDS

Bu isteğe bağlı deyimler, yeni tablonun satırlarının onlara atanacak OID'lere (nesne belirteçlerine) sahip olup olmayacağını belirtmekte kullanılır. Eğer ne **WITH OIDS** ne de **WITHOUT OIDS** belirtilmişse, öntanımlı değer `default_with_oids` yapılandırma parametresinin değerine bağlıdır. (Eğer yeni tablo nesne belirteçlerine sahip bir tabloyu miras alıyorsa, tabloyu oluşturan cümle **WITHOUT OIDS** içerse bile **WITH OIDS** geçerli olur.)

Eğer **WITHOUT OIDS** belirtilmiş ve uygulanmışsa, yeni tablo nesne belirteçlerini saklamaz ve girilen her yeni satır için bir nesne belirteci atanmaz. Nesne belirteci tüketimi azalacağından ve bu suretle 32 bitlik OID sayacının başa dönmesi erteleneceğinden genelde edilen zahmete değer. Sayaç bir kere başa döndü mü, nesne belirteçlerinin artık eşsiz olmayacağı varsayılır ve bu onları nispeten daha az yararlı hale getirir. Nesne belirteçlerinin dışlanması ek olarak, her satır için 4 bayt (çoğu makinede) olmak üzere tablonun disk üzerinde kaplayacağı alanı düşürecek ve başarımı oldukça arttıracaktır.

Tablo oluşturulduktan sonra nesne belirteçlerini kaldırmak için **ALTER TABLE** [`alter_table(7)`] kullanın.

CONSTRAINT *kısıt_ismi*

Bir sütun ya da tablo kısıtı için isteğe bağlı bir isim. Belirtilmezse ismi sistem üretir.

NOT NULL

Sütunun boş değer içeremeyeceğini belirtir.

NULL

Sütunun boş değer içermesine izin verilir. bu öntanımlıdır.

Bu deyim, sadece standartdışı SQL veritabanlarıyla uyumluluk için vardır. Yeni uygulamalarda kullanımından vazgeçilmelidir.

UNIQUE (*sütun_kısıtı*)
UNIQUE (*sütun_ismi* [, ...]) (*tablo_kısıtı*)

UNIQUE kısıtı, bir tablonun bir veya daha fazla sütunundan oluşan bir grubunun sadece eşsiz değerler içerebileceğini belirtir. Eşsiz tablo kısıtının davranışı, çok sayıda sütun belirtilebilmesi dışında sütun kısıtınınki ile aynıdır.

Eşsizlik kısıtının amacına uygun olarak, boş değerlerin eşit olmadıkları varsayılır.

Her eşsiz tablo kısıtı, aynı tabloda tanımlanmış diğer eşsizlik veya birincil anahtar kısıtı tarafından isimlendirilmiş sütun grubundan farklı bir sütun grubunu isimlendirmelidir. (Aksi takdirde, yalnızca, aynı kısıt iki kere listelenmiş olur.)

PRIMARY KEY (*sütun_kısıtı*)
PRIMARY KEY (*sütun_ismi* [, ...]) (*tablo_kısıtı*)

Birincil anahtar kısıtı bir tablonun bir sütununun ya da sütunlarının sadece elsiz ve boş olmayan değerler içerebileceğini belirtir. Teknik olarak, **PRIMARY KEY** sadece **UNIQUE** ile **NOT NULL** deyiminin birleşimidir, fakat bir sütun grubunun birincil anahtar olarak belirtilmesi ayrıca, şema tasarımı hakkında hamveri sağlar; bir birincil anahtar uygulanmış gibi diğer tablolar, bu tablonun satırları için bir eşsiz belirteç olarak bu sütun grubuna bel bağlayayabilir.

İster sütun ister tablo kısıtı olarak belirtilsin, bir tablo için sadece bir birincil anahtar belirtilebilir.

Birincil anahtar kısıtı, aynı tabloda tanımlanmış bir eşsizlik kısıtı tarafından isimlendirilmiş diğer sütun gruplarından farklı bir sütun grubunu isimlendirmelidir.

CHECK (*ifade*)

CHECK deyimi, bir veri girme veya güncelleme işlemi sonucunda yeni ya da güncellenmiş satırların sağlaması gereken bir mantıksal sonuç üreten bir ifadeyi belirtmek için kullanılır. TRUE veya UNKNOWN olarak sonuçlanan ifadeler başarılı kabul edilir. Bir veri girme veya güncelleme işleminin ürettiği bir FALSE sonucunda bir hata olağandışılığı oluşur ve bu veri girme veya güncelleme işlemi veritabanını değiştirmez. Bir tablo kısıtı içinde görünen bir ifade çok sayıda sütunun değeri ile ilgili olabilirken, bir sütun kısıtı olarak belirtilen bir sınama kısıtı sadece bu sütunun değeriyle ilgili olmalıdır.

Şimdilik, **CHECK** ifadeleri ne altsorgu içerebilir ne de üzerinde çalışılan satır dışındaki sütunların değerleriyle ilgili olabilir.

REFERENCES *başvuru_tablosu* [(*başvuru_sütunu*)]
[**MATCH FULL** | **MATCH PARTIAL** | **MATCH SIMPLE**]
[**ON DELETE** *eylem*] [**ON UPDATE** *eylem*
FOREIGN KEY (*sütun_ismi* [, ...])
REFERENCES *başvuru_tablosu* [(*başvuru_sütunu* [, ...])]
[**MATCH FULL** | **MATCH PARTIAL** | **MATCH SIMPLE**]
[**ON DELETE** *eylem*]
[**ON UPDATE** *eylem*] } (*tablo_kısıtı*)

Bu deyimler, başvurulan tablonun bir satırının başvurulan sütunlarındaki değerlerle eşleşmesi gereken değerler içermesi istenen yeni tablonun bir ya da daha fazla sütunundan oluşan bir sütun grubunu gerektiren bir yabancı anahtar kısıtı belirtir. Eğer *başvuru_sütunu* belirtilmezse, *başvuru_tablosu*'nun birincil anahtarı kullanılır. Başvurulan sütunlar, başvurulan tablodaki bir eşsizlik veya bir birincil anahtar kısıtınının sütunları olmalıdır.

Bşvurulan sütunlara girilen bir değer, başvurulan tablonun ve başvurulan sütunların değerleriyle belirtilen eşleşme türü kullanılarak eşleşmelidir. Üç eşleşme türü vardır: **MATCH FULL**, **MATCH PARTIAL** ve aynı zamanda öntanımlı olan **MATCH SIMPLE**. **MATCH FULL**, tüm yabancı anahtar sütunları boş olmadıkça, bir çok sütunlu yabancı anahtarın tek sütununun boş olmasına izin vermeyecektir. **MATCH SIMPLE**, diğer yabancı anahtar sütunları boş değer içermezken, bazı yabancı anahtar sütunlarının boş değer içermesine izin verecektir. **MATCH PARTIAL** ise henüz gerçeklenmemiştir.

Ek olarak, başvurulan sütunlardaki veri değiştiğinde, bu tablonun sütunlarının verisi üzerinde bazı eylemler uygulanır. **ON DELETE** deyimi, başvurulan tablodaki başvurulan satır silindiğinde uygulanacak eylemi belirtmekte kullanılır. Benzer şekilde, **ON UPDATE** deyimi, başvurulan tablodaki başvurulan sütun yeni bir değerle güncellendiğinde uygulanacak eylemi belirtmekte kullanılır. Eğer başvurulan satır güncellenmeksizin bu sütunu içeren satır güncellenmişse, bir işlem yapılmaz. **NO ACTION** sınaması dışında hiçbir göreli eylem, kısıtta ertelenebileceği belirtilmiş olsa bile ertelenemez. Her deyim için olası eylemler şunlardır:

NO ACTION
: Silme veya güncelleme işlemini bir yabancı anahtar kısıtı çelişkisi ürettiğinde bunu belirten bir hatanın üretilmesini sağlar. Eğer kısıt ertelenmişse ve başvurulan satırlar hala mevcutsa, bu hata kısıtın sınanması sırasında üretilecektir. Bu öntanımlı eylemdir.

RESTRICT
: Silme veya güncelleme işlemini bir yabancı anahtar kısıtı çelişkisi ürettiğinde bunu belirten bir hatanın üretilmesini sağlar. Sınamanın ertelenebilir olmaması dışında **NO ACTION**'a benzer.

CASCADE
: Silinen satıra başvuran bir satırın silinmesini ya da başvuran sütunun değerinin başvurulan sütunun yeni değerine güncellenmesini sağlar.

SET NULL
: Başvuran sütunların boş olmasını sağlar.

SET DEFAULT
: Başvuran sütunlara öntanımlı değerlerinin atanmasını sağlar.

Eğer başvurulan sütunlar sıkça değişmiyorsa, yabancı anahtar sütununa bir indeks eklemek akıllıca olur, böylece yabancı anahtar sütunu ile ilgili göreli eylemler daha verimli uygulanabilir.

DEFERRABLE
NOT DEFERRABLE
: Kısıtın ertelenip ertelenmeyeceğini belirler. Ertelenebilir olmayan (not deferrable) bir kısıt her deyimden sonra anında sınanacaktır. Ertelenebilir bir kısıt ise, **SET CONSTRAINTS** [set-constraints(7)] deyimi kullanılarak hareketin sonuna kadar ertelenebilir. **NOT DEFERRABLE** öntanımlıdır. Şimdilik sadece yabancı anahtar kısıtları bu deyimleri kabul etmektedir. Diğer tüm kısıt türleri ertelenebilir değildir.

INITIALLY IMMEDIATE
INITIALLY DEFERRED
: Eğer bir kısıt ertelenebilirse, bu deyim kısıtın öntanımlı sınama zamanını belirtmek için kullanılabilir. Eğer kısıt **INITIALLY IMMEDIATE** ise, her deyimden sonra sınanacaktır. Eğer kısıt **INITIALLY DEFERRED** ise, sadece hareketin sonunda sınanır. Kısıtın sınama zamanı **SET CONSTRAINTS** [set_constraints(7)] cümlesi ile değiştirilebilir.

ON COMMIT

Geçici tabloların hareket kümesinin sonundaki davranışı, **ON COMMIT** kullanılarak denetlenebilir. Üç seçeneği vardır:

PRESERVE ROWS
Hareketin sonunda özel bir eylem yapılmaz. Bu öntanımlı davranıştır.

DELETE ROWS
Her hareket kümesinin sonunda geçici tablonun satırları silinir. Aslında, her teslimde (commit) özdevinimli bir **TRUNCATE** [truncate(7)] yapılır.

DROP
Geçici tablo o anki hareket kümesinin sonunda silinecektir.

TABLESPACE *tabloalanı*
tabloalanı, yeni tablonun içinde oluşturulacağı tablo alanının ismidir. Belirtilmezse, default_tablespace yapılandırma parametresinin değeri, bu parametrenin değeri boş dizge ise veritabanının öntanımlı tablo alanı kullanılır.

USING INDEX TABLESPACE *tabloalanı*
Bir **UNIQUE** veya **PRIMARY KEY** kısıtı ile ilişkili olarak oluşturulacak indeksdeki tablo alanının seçimini mümkün kılar. Belirtilmezse, default_tablespace yapılandırma parametresinin değeri, bu parametrenin değeri boş dizge ise veritabanının öntanımlı tablo alanı kullanılır.

Ek Bilgi

Nesne belirteçlerinin (OID) yeni uygulamalarda kullanılması önerilmez: mümkünse, bir **SERIAL** veya başka bir kayıt listesi üretecinin, tablonun birincil anahtarı olarak kullanılması önerilir. Yine de uygulamanız, bir tablonun belli bir satırını belirtmek için nesne belirteçlerini kullanıyorsa, tablodaki nesne belirteçlerinin sayaç başa döndükten spnra bile satırları eşsiz olarak belirttiğinden emin olmak için tablonun oid sütununda bir eşsizlik kısıtı oluşturmanızı öneririz. Bu nesne belirteçlerinin veritabanı çapında eşsiz olduğunu kabulden kaçının; eğer veritabanı çapında eşsiz bir belirtece ihtiyacınız varsa, tableoid ile tablonun oid sütununu birlikte kullanın.

İpucu

Birincil anahtarı olmayan tablolar için, hem bir OID hem de bir eşsiz veri anahtarı olmaksızın, satırları belirtmek zor olacağından, **WITHOUT OIDS** kullanımı önerilmez.

PostgreSQL eşsizliği güçlendirmek için her eşsizlik kısıtı ve her birincil anahtar kısıtı için bir indeksi özdevinimli olarak oluşturur. Bu suretle, birincil anahtar ütunu için açıkça bir indeks oluşturmak gerekmez. (Daha fazla bilgi için **CREATE INDEX** [create_index(7)] kılavuz sayfasına bakınız.)

Eşsizlik kısıtları ve birincil anahtarlar, şimdiki gerçeklenimde miras alınmamaktadır. Bu, eşsizlik kısıtları ile miras almanın birleşimini tersine işlevsiz yapar.

Bir tablo 1600 sütundan fazla sütun içeremez. (Uygulamada, demet uzunluğu kısıtlarından dolayı etkin sınır daha düşüktür.)

Örnekler

films ve distributors tablolarını oluşturmak için:

```
CREATE TABLE films (
    code        char(5) CONSTRAINT firstkey PRIMARY KEY,
    title       varchar(40) NOT NULL,
```

```
    did         integer NOT NULL,
    date_prod   date,
    kind        varchar(10),
    len         interval hour to minute
);

CREATE TABLE distributors (
    did    integer PRIMARY KEY DEFAULT nextval('serial'),
    name   varchar(40) NOT NULL CHECK (name <> '')
);
```

2 boyutlu bir dizi ile bir tablo oluşturmak için:

```
CREATE TABLE array_int (
    vector  int[][]
);
```

`films` tablosu için bir eşsiz tablo kısıtının tanımlanması. Eşsiz tablo kısıtları tablonun bir veya daha fazla sütunu için tanımlanabilir:

```
CREATE TABLE films (
    code        char(5),
    title       varchar(40),
    did         integer,
    date_prod   date,
    kind        varchar(10),
    len         interval hour to minute,
    CONSTRAINT production UNIQUE(date_prod)
);
```

Bir sınama sütunu kısıtı tanımlamak için:

```
CREATE TABLE distributors (
    did     integer CHECK (did > 100),
    name    varchar(40)
);
```

Bir sınama tablo kısıtı tanımlamak için:

```
CREATE TABLE distributors (
    did     integer,
    name    varchar(40)
    CONSTRAINT con1 CHECK (did > 100 AND name <> '')
);
```

`films` tablosu için bir birincil anahtar tablo kısıtının tanımlanması. Birincil anahtar tablo kısıtları tablonun bir ya da daha fazla sütununda tanımlanabilir:

```
CREATE TABLE films (
    code        char(5),
    title       varchar(40),
    did         integer,
    date_prod   date,
```

```
    kind        varchar(10),
    len         interval hour to minute,
    CONSTRAINT code_title PRIMARY KEY(code,title)
);
```

`distributors` tablosu için bir birincil anahtar kısıtının tanımlanması. Aşağıdaki iki örnek eşdeğerdir, ilki tablo kısıtı sözdizimini, ikincisi sütun kısıtı sözdizimini kullanır:

```
CREATE TABLE distributors (
    did     integer,
    name    varchar(40),
    PRIMARY KEY(did)
);

CREATE TABLE distributors (
    did     integer PRIMARY KEY,
    name    varchar(40)
);
```

Aşağıdaki örnekte, `name` sütunu için öntanımlı değer olarak bir dizge sabiti atanmakta; `did` sütununun öntanımlı değeri bir kayıt listesinin sonraki değeri seçilerek üretilmekte; `modtime` sütununun öntanımlı değeri ise satıra verinin girildiği zaman olmaktadır.

```
CREATE TABLE distributors (
    name      varchar(40) DEFAULT 'Luso Films',
    did       integer DEFAULT nextval('distributors_serial'),
    modtime   timestamp DEFAULT current_timestamp
);
```

`distributors` tablosunda açıkça bir isim belirterek iki **`NOT NULL`** sütun kısıtının tanımlanması:

```
CREATE TABLE distributors (
    did     integer CONSTRAINT no_null NOT NULL,
    name    varchar(40) NOT NULL
);
```

`name` sütunu için bir eşsizlik kısıtının tanımlanması:

```
CREATE TABLE distributors (
    did     integer,
    name    varchar(40) UNIQUE
);
```

Bu örnek, yukarıdaki örneğe eşdeğerdir:

```
CREATE TABLE distributors (
    did     integer,
    name    varchar(40),
    UNIQUE(name)
);
```

`diskvol1` tablo alanında `cinemas` tablosunun oluşturulması:

```
CREATE TABLE cinemas (
        id serial,
        name text,
        location text
) TABLESPACE diskvol1;
```

Uyumluluk

CREATE TABLE cümlesi SQL–92 ve SQL:1999'un bir alt kümesi ile aşağıda belirtilenler dışında uyumludur.

Geçici tablolar

CREATE TEMPORARY TABLE sözdizimi bakımından SQL standardına benzese de etkisi aynı değildir. Standartta, geçici tablolar bir defada tanımlanır ve onlara ihtiyaç oldukça he oturumda özdevinimli olarak mevcut olurlar (başlangıçta boş içerikle). PostgreSQL'de ise, bir geçici tablonun her gerekişinde oturumda **CREATE TEMPORARY TABLE** cümlesinin açıkça çalıştırılması gerekir. Bu, farklı oturumların aynı geçici tabloyu farklı amaçlarla kullanabilmesine olanak sağlar. Halbuki standardın yaklaşımı, belirtilen geçici tablo ismiyle erişilen her kopyanın aynı tablo yapısına sahip olması şeklinde bir sınırlama içerir.

Standardın geçici tabloların davranışıyla ilgili tanımı geniş çapta yoksayılır. Bu noktada, PostgreSQL'in davranışı çeşitli başka SQL veritabanlarınınkine benzerlik gösterir.

Standardın yerel ve genel geçici tabloları ayrımsama yöntemi, PostgreSQL'in sahip olmadığı modül kavramına oturtulduğundan, PostgreSQL aynı yöntemi kullanmaz. Uyumluluk uğruna, PostgreSQL bir geçici tablo bildiriminde **GLOBAL** ve **LOCAL** anahtar sözcüklerini kabul eder, ama bunların bir etkisi yoktur.

Geçici tablolar için **ON COMMIT** deyimi SQL standardıyla benzerlik gösterse de, bazı farklar vardır. Eğer **ON COMMIT** belirtilmezse, SQL öntanımlı davranış olarak **ON COMMIT DELETE ROWS** belirtiken, PostgreSQL öntanımlı davranış olarak **ON COMMIT PRESERVE ROWS** belirtir. **ON COMMIT DROP** seçeneği ise SQL standardında yoktur.

Sütun Sınama Kısıtları

SQL standardı, **CHECK** sütun kısıtlarının sadece uygulandığı sütun ile ilgili olabileceğini söyler; sadece **CHECK** tablo kısıtları çok sayıda sütun ile ilgili olabilir. PostgreSQL böyle bir sınırlamayı zorunlu kılmaz; sütun ve tablo kısıtlarının farksız olduğunu kabul eder.

NULL Kısıtı

NULL kısıtı (aslında bir kısıt değildir) (ve simetriği olan **NOT NULL** kısıtı), bir PostgreSQL oluşumudur ve bazı başka veritabanı sistemleriyle uyumluluk adına vardır. Bir sütun için zaten öntanımlı olduğundan varlığı basitçe kuru gürültüdür.

Kalıtım

INHERITS deyimi ile çok sayıda miras alınması bir PostgreSQL oluşumudur. SQL:1999 standardı (ama, SQL–92 değil), farklı bir sözdizimi ve farklı sözcüklerle tek bir miras alma deyimi tanımlar. SQL:1999 tarzı kalıtım henüz PostgreSQL tarafından desteklenmemektedir.

Nesne Kimlikleri

PostgreSQL'in OID kavramı standart değildir.

Sıfır Sütunluk Tablolar

PostgreSQL bir tablonun hiç sütun içermeksizin oluşturulmasına izin verir (**CREATE TABLE foo();** gibi). Bu bir PostgreSQL oluşumudur. SQL standardı sıfır sütunluk tablolara izin vermez. Sıfır sütunluk tablolar kendi başlarına çok kullanışlı olmamakla beraber, buna izin verilmemesi, **ALTER TABLE DROP COLUMN** için tuhaf özel durumlar oluşmasına yol açar, dolayısıyla standardın bu sınırlamasını yoksaymak daha temiz görünür.

Tablo Alanları

PostgreSQL'in tablo alanı kavramı standardın bir parçası değildir. Dolayısıyla, **TABLESPACE** ve **USING INDEX TABLESPACE** deyimleri birer PostgreSQL oluşumudur.

İlgili Belgeler

ALTER TABLE [alter_table(7)], **DROP TABLE** [drop_table(7)], **CREATE TABLESPACE** [create_tablespace(7)].

Çeviren

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

YASAL UYARI

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

31 Ocak 2007