CREATE TABLE

Name

CREATE TABLE -- define uma nova tabela

Synopsis

```
CREATE [ [ LOCAL ] { TEMPORARY | TEMP } ] TABLE nome_da_tabela (
    { nome_da_coluna tipo_de_dado [ DEFAULT expressão_padrão ] [ restrição_de_coluna [, ... ] ]
    | restrição_de_tabela }  [, ... ]
)
[ INHERITS ( tabela_ascendente [, ... ] ) ]
[ WITH OIDS | WITHOUT OIDS ]

onde restrição_de_coluna é:

[ CONSTRAINT nome_da_restrição ]
{ NOT NULL | NULL | UNIQUE | PRIMARY KEY |
  CHECK (expressão) |
  REFERENCES tabela_referenciada [ ( coluna_referenciada ) ] [ MATCH FULL | MATCH PARTIAL ]
    [ ON DELETE ação ] [ ON UPDATE ação ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

e restrição_de_tabela é:

[ CONSTRAINT nome_da_restrição ]
{ UNIQUE ( nome_da_coluna [, ... ] ) |
  PRIMARY KEY ( nome_da_coluna [, ... ] ) |
  CHECK ( expressão ) |
  FOREIGN KEY ( nome_da_coluna [, ... ] ) REFERENCES tabela_referenciada [ ( coluna_referenciada [, ... ] )
    [ MATCH FULL | MATCH PARTIAL ] [ ON DELETE ação ] [ ON UPDATE ação ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

Descrição

O comando `CREATE TABLE` cria uma tabela nova, inicialmente vazia, no banco de dados atual. O usuário que executa o comando torna-se o dono da tabela.

O comando `CREATE TABLE` também cria, automaticamente, um tipo de dado que representa o tipo da tupla (tipo estrutura) correspondente a uma linha da tabela. Portanto, uma tabela não pode ter o mesmo nome de um tipo de dado existente.

Uma tabela não pode possuir mais de 1600 colunas (Na prática, o limite efetivo é menor, devido à restrição do comprimento das tuplas). Uma tabela não pode ter o mesmo nome de uma tabela do catálogo do sistema.

As cláusulas opcionais de restrição especificam as restrições (ou testes) que as linhas novas ou modificadas devem satisfazer para a operação de inclusão ou de alteração ser completada. Uma restrição é uma regra com nome: um objeto SQL que ajuda a definir conjuntos válidos de valores, estabelecendo limites nos resultados das operações de inclusão, exclusão e atualização realizadas na tabela.

Existem duas formas para definir restrições: restrições de tabela e restrições de coluna. A restrição de coluna é definida como parte da definição da coluna. A restrição de tabela não está presa a uma coluna em particular, podendo abranger mais de uma coluna. Toda restrição de coluna também pode ser escrita como uma restrição de tabela; a restrição de coluna é somente uma notação conveniente, no caso da restrição afetar apenas uma coluna.

Parâmetros

`[LOCAL] TEMPORARY` ou `[LOCAL] TEMP`

Se for especificado, a tabela é criada como uma tabela temporária. Tabelas temporárias são automaticamente eliminadas no final da sessão. Uma tabela permanente com o mesmo nome, caso exista, não será visível na sessão corrente enquanto a tabela temporária existir. Todo índice criado em tabela temporária também é temporário.

A palavra `LOCAL` é opcional. Veja sob *Compatibilidade*.

nome_da_tabela

O nome da tabela a ser criada.

nome_da_coluna

O nome da coluna a ser criada na nova tabela.

tipo_de_dado

O tipo de dado da coluna. Pode incluir especificadores de `array`. Consulte o *Guia do Usuário* para obter mais informações sobre tipos de dado e sobre `arrays`.

`DEFAULT` *expressão_padrão*

A cláusula `DEFAULT` atribui um valor padrão para o dado da coluna em cuja definição está presente. O valor pode ser qualquer expressão (subconsultas e referências cruzadas para outras colunas da mesma tabela não são permitidas). O tipo de dado da expressão padrão deve corresponder ao tipo de dado da coluna.

A expressão é utilizada em todas as operações de inclusão que não especificam valor para a coluna. Não havendo valor padrão para a coluna, então NULL torna-se o valor padrão.

`INHERITS ( tabela_ascendente [, ... ] )`

A cláusula opcional `INHERITS` (herda) especifica uma lista de tabelas das quais a nova tabela herda, automaticamente, todas as colunas. Se o mesmo nome de coluna existir em mais de uma tabela ascendente um erro é gerado, a menos que o tipo de dado das colunas seja o mesmo em todas as tabelas ascendentes. Se não houver conflito, então as colunas duplicadas são mescladas para formar uma única coluna da nova tabela. Se a lista de nomes de colunas da nova tabela contém uma coluna que também é herdada, da mesma forma o tipo de dado deve ser o mesmo das colunas herdadas, e a definição das colunas será mesclada em uma única coluna. Entretanto, declarações de colunas novas e herdadas com o mesmo nome não precisam especificar restrições idênticas: todas as restrições fornecidas em qualquer uma das declarações são mescladas, sendo todas aplicadas à nova tabela. Se a nova tabela especificar, explicitamente, um valor padrão para a coluna, este valor padrão substitui qualquer valor padrão das declarações herdadas. Fora isso, toda tabela ascendente que especificar um valor padrão para a coluna deve especificar o mesmo valor, ou um erro será gerado.

`WITH OIDS` ou `WITHOUT OIDS`

Esta cláusula opcional especifica se as linhas da nova tabela devem possuir OIDs (identificadores do objeto). O padrão é possuir OIDs (Se a nova tabela herdar de qualquer tabela que possua OIDs, então a cláusula `WITH OIDS` é forçada mesmo que no comando esteja especificado `WITHOUT OIDS`).

A especificação de `WITHOUT OIDS` permite ao usuário suprimir a geração dos OIDs para as linhas da tabela. Este procedimento pode ser interessante em tabelas grandes, porque reduz o consumo de OIDs e, portanto, adia o recomeço deste contador de 32 bits. Quando o contador recomeça, não é mais possível assumir a sua unicidade, o que reduz muito a sua utilidade.

`CONSTRAINT` *nome_da_restrição*

Um nome opcional para a restrição da coluna ou da tabela. Se não for especificado, o nome será gerado pelo sistema.

`NOT NULL`

Valores nulos não são permitidos na coluna. Esta declaração é equivalente à restrição de coluna `CHECK (`*nome_da_coluna* `NOT NULL)`.

`NULL`

Valores nulos são permitidos na coluna. Este é o padrão.

Esta cláusula só está disponível por compatibilidade com bancos de dados SQL fora do padrão. Sua utiização é desestimulada em novas aplicações.

`UNIQUE` (restrição da coluna)
`UNIQUE ( nome_da_coluna [, ... ] )` (restrição da tabela)

A restrição `UNIQUE` especifica a regra onde um grupo de uma ou mais colunas distintas de uma tabela podem conter apenas valores únicos. O comportamento da restrição de unicidade para tabelas é o mesmo da restrição de unicidade para colunas, porém com a capacidade adicional de abranger várias colunas.

Para a finalidade da restrição de unicidade, valores nulos não são considerados iguais.

Cada restrição de unicidade da tabela deve abranger um conjunto de colunas diferente do conjunto de colunas abrangido por qualquer outra restrição de unicidade e da chave primária definida para a tabela (Senão, seria apenas a mesma restrição declarada duas vezes).

`PRIMARY KEY` (restrição da coluna)
`PRIMARY KEY ( nome_da_coluna [, ... ] )` (restrição da tabela)

A restrição de chave primária especifica que a coluna, ou colunas, da tabela pode conter apenas valores únicos (não duplicados) e não nulos. Tecnicamente a chave primária (`PRIMARY KEY`) é simplesmente uma combinação de unicidade (`UNIQUE`) com não nulo (`NOT NULL`), mas identificar um conjunto de colunas como chave primária também fornece metadados sobre o projeto do esquema, porque chaves primárias indicam que outras tabelas podem depender deste conjunto de colunas como um identificador único para linhas.

Somente uma chave primária pode ser especificada para uma tabela, seja como uma restrição de coluna ou como uma restrição de tabela.

A restrição de chave primária deve abranger um conjunto de colunas que seja diferente de outro conjunto de colunas abrangido por uma restrição de unicidade definida para a mesma tabela.

`CHECK (expressão)`

A cláusula `CHECK` especifica restrições de integridade, ou testes, que as linhas novas ou atualizadas devem atender para que uma operação de inserção ou de atualização complete. Cada restrição deve ser uma expressão que produza um resultado booleano. Uma condição declarada na definição da coluna deve fazer referência apenas ao valor desta coluna, enquanto uma condição declarada como uma restrição da tabela pode fazer referência a várias colunas.

Atualmente, as expresões de `CHECK` não podem conter subconsultas, nem fazer referência a variáveis que não sejam colunas da linha atual.

`REFERENCES tabela_referenciada [ ( coluna_referenciada ) ] [ MATCH tipo_corresp ] [ ON DELETE ação ] [ ON UPDATE ação ]` (restrição da coluna)
`FOREIGN KEY ( nome_da_coluna [, ... ] ) REFERENCES tabela_referenciada [ ( coluna_referenciada [, ... ] ) ] [ MATCH tipo_corresp ] [ ON DELETE ação ] [ ON UPDATE ação ]` (restrição da tabela)

A restrição `REFERENCES` especifica que um grupo de uma ou mais colunas da nova tabela deve conter somente valores correspondentes aos valores das colunas referenciadas *`coluna_referenciada`* da tabela referenciada *`tabela_referenciada`*. Se a *`coluna_referenciada`* for omitida, a chave primária da *`tabela_referenciada`* é utilizada. As colunas referenciadas devem pertencer a uma restrição de chave primária ou de unicidade da tabela referenciada.

Os valores adicionados a estas colunas são comparados com os valores das colunas referenciadas da tabela referenciada utilizando o tipo de comparação. Existem 3 tipos de comparação: `MATCH FULL`, `MATCH PARTIAL` e o tipo de comparação padrão se nada for especificado. `MATCH FULL` não permite que uma coluna de uma chave estrangeira com várias colunas seja nula, a menos que todas as colunas da chave estrangeira sejam nulas. O tipo de comparação padrão permite que algumas colunas da chave estrangeira sejam nulas enquanto outras colunas da chave estrangeira não são nulas. `MATCH PARTIAL` ainda não está implementado.

Adicionalmente, quando os dados das colunas referenciadas são modificados, certas ações são realizadas nos dados das colunas desta tabela. A cláusula `ON DELETE` especifica a ação a ser executada quando uma linha referenciada da tabela referenciada é excluída. Da mesma maneira, a cláusula `ON UPDATE` especifica a ação a ser executada quando uma coluna referenciada da tabela referenciada muda de valor. Se a linha é atualizada, mas a coluna referenciada não muda de valor, nenhuma ação é executada. São possíveis as seguintes ações para cada cláusula:

`NO ACTION`

Gera um erro indicando que a exclusão ou a atualização criaria uma violação de chave estrangeira. Esta é a ação padrão.

`RESTRICT`

O mesmo que `NO ACTION`.

`CASCADE`

Exclui qualquer linha que faça referência à linha excluída, ou atualiza o valor da coluna que faz referência usando o novo valor da coluna referenciada, respectivamente.

`SET NULL`

Atribui nulo aos valores das colunas que fazem referência.

`SET DEFAULT`

Atribui o valor padrão às colunas que fazem referência.

Se a coluna da chave primária é atualizada freqüentemente, pode ser útil adicionar um índice às colunas da cláusula `REFERENCES`, para que as ações `NO ACTION` e `CASCADE` associadas às colunas da cláusula `REFERENCES` sejam executadas mais rapidamente.

`DEFERRABLE` ou `NOT DEFERRABLE`

Estas cláusulas controlam se as restrições podem ser postergadas. Uma restrição que não pode ser postergada é verificada imediatamente após cada comando. A verificação das restrições que são postergáveis pode ser adiada para o final da transação (usando o comando *SET CONSTRAINTS*). O padrão é `NOT DEFERRABLE`. Somente restrições de chave estrangeira aceitam esta cláusula no momento. Todos os outros tipos de restrição não são postergáveis.

`INITIALLY IMMEDIATE` ou `INITIALLY DEFERRED`

Se uma restrição é postergável, esta cláusula especifica o momento padrão para verificar a restrição. Se a restrição é `INITIALLY IMMEDIATE`, então é verificada após cada declaração. Este é o padrão. Se a declaração é `INITIALLY DEFERRED`, então é verificada apenas no final da transação. O momento de verificação da restrição pode ser alterado pelo comando *SET CONSTRAINTS*.

Diagnósticos

`CREATE`

Mensagem retornada se a tabela for criada com sucesso.

`ERROR`

Mensagem retornada se a criação da tabela falhar. Esta mensagem é normalmente acompanhada por algum texto descritivo, como: `ERROR: Relation 'nome_da_tabela' already exists`, que ocorre quando a tabela especificada existe no banco de dados.

Notas

- Sempre que uma aplicação faz uso dos OIDs para identificar linhas específicas de uma tabela, é recomendado criar uma restrição de unicidade para a coluna `oid` da tabela, para garantir que os OIDs na tabela realmente identificam unicamente uma linha, mesmo após o contador recomeçar. Evite assumir que os OIDs são únicos entre tabelas; se for necessário um identificador único para todo o banco de dados, use uma combinação do `tableoid` (OID da tabela) com o OID da linha para esta finalidade (é provável que nas versões futuras do PostgreSQL existam contadores OID separados para cada tabela, então será *necessário*, e não opcional, incluir o `tableoid` para ter-se um identificador único para todo o banco de dados).

  Tip: O uso de `WITHOUT OIDS` não é recomendado para tabelas sem chave primária porque sem um OID, e sem uma chave de dados única, fica difícil identificar uma linha específica.

- O PostgreSQL cria, automaticamente, um índice para cada restrição de unicidade e de chave primária para garantir a sua unicidade. Portanto, não é necessário criar um índice explícito para as colunas da chave primária. (Consulte o comando *CREATE INDEX* para obter mais informações)
- O padrão SQL92 especifica que as restrições `CHECK` de colunas podem referenciar apenas a coluna à qual se aplica; somente restrições `CHECK` de tabelas podem fazer referências a várias colunas. O PostgreSQL não impõe esta restrição; as restrições de tabela e de colunas são tratadas da mesma maneira.
- As restrições de unicidade e de chave primária não são herdadas na implementação atual, tornando o comportamento da combinação de herança com restrição de unicidade diferente do esperado.

Exemplos

Criar a tabela `filmes` e a tabela `distribuidores`:

```
CREATE TABLE filmes (
    cod         CHARACTER(5) CONSTRAINT pk_filmes PRIMARY KEY,
    titulo      CHARACTER VARYING(40) NOT NULL,
    did         DECIMAL(3) NOT NULL,
    data_prod   DATE,
    tipo        CHAR(10),
    duracao     INTERVAL HOUR TO MINUTE
);

CREATE TABLE distribuidores (
     did    DECIMAL(3) PRIMARY KEY DEFAULT NEXTVAL('serial'),
     nome   VARCHAR(40) NOT NULL CHECK (nome <> '')
);
```

Criar uma tabela com uma matriz de 2 dimensões

```
CREATE TABLE matriz2d (
    matriz  INT[][]
);
```

Definir uma restrição de unicidade para a tabela filmes. Restrições de unicidade de tabela podem ser definidas usando uma ou mais colunas da tabela:

```
CREATE TABLE filmes (
    cod         CHAR(5),
    titulo      VARCHAR(40),
    did         DECIMAL(3),
    data_prod   DATE,
    tipo        VARCHAR(10),
    duracao     INTERVAL HOUR TO MINUTE,
    CONSTRAINT producao UNIQUE(data_prod)
);
```

Definir uma restrição de coluna para verificação:

```
CREATE TABLE distribuidores (
    did     DECIMAL(3) CHECK (did > 100),
    nome    VARCHAR(40)
);
```

Definir uma restrição de tabela para verificação:

```
CREATE TABLE distribuidores (
    did     DECIMAL(3),
    nome    VARCHAR(40)
    CONSTRAINT chk_dist CHECK (did > 100 AND nome <> '')
);
```

Definir uma restrição de chave primária para a tabela `filmes`. Restrições de chave primária da tabela podem ser definidas usando uma ou mais colunas da tabela.

```
CREATE TABLE filmes (
    cod         CHAR(5),
    titulo      VARCHAR(40),
    did         DECIMAL(3),
```

```
    data_prod   DATE,
    tipo        VARCHAR(10),
    duracao     INTERVAL HOUR TO MINUTE,
    CONSTRAINT pk_filmes PRIMARY KEY(cod,titulo)
);
```

Definir a restrição de chave primária para a tabela `distribuidores`. Os dois exemplos abaixo são equivalentes, o primeiro utiliza a sintaxe de restrição de tabela, e o segundo utiliza a notação de restrição de coluna.

```
CREATE TABLE distribuidores (
    did     DECIMAL(3),
    nome    CHAR VARYING(40),
    PRIMARY KEY(did)
);

CREATE TABLE distribuidores (
    did     DECIMAL(3) PRIMARY KEY,
    nome    VARCHAR(40)
);
```

O comando abaixo especifica uma constante literal como valor padrão da coluna `nome`; faz com que o valor padrão da coluna `did` seja gerado como o próximo valor de um objeto de seqüência; faz com que o valor padrão da coluna `data_ins` seja a data e hora em que a linha é inserida.

```
CREATE TABLE distribuidores (
    nome      VARCHAR(40) DEFAULT 'luso filmes',
    did       INTEGER DEFAULT NEXTVAL('seq_distribuidores'),
    data_ins  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Definir duas restrições de coluna `NOT NULL` na tabela `distribuidores`, sendo que uma delas recebe um nome explicito:

```
CREATE TABLE distribuidores (
    did     DECIMAL(3) CONSTRAINT nao_nulo NOT NULL,
    nome    VARCHAR(40) NOT NULL
);
```

Definir uma restrição de unicidade para a coluna `nome`:

```
CREATE TABLE distribuidores (
    did     DECIMAL(3),
    nome    VARCHAR(40) UNIQUE
);
```

O comando acima é equivalente ao mostrado abaixo, especificado através de uma restriçao de tabela:

```
CREATE TABLE distribuidores (
    did     DECIMAL(3),
    nome    VARCHAR(40),
    UNIQUE(nome)
);
```

Compatibilidade

O comando `CREATE TABLE` está em conformidade com o "SQL92 Intermediate" e com um subconjunto do SQL99, com excessões listadas abaixo e nas descrições acima.

Tabelas temporárias

Além da tabela temporária local, o SQL92 também define a declaração `CREATE GLOBAL TEMPORARY TABLE`. Tabelas temporárias globais também são visíveis por outras sessões.

Para as tabelas temporárias existe uma cláusula opcional `ON COMMIT`:

```
CREATE { GLOBAL | LOCAL } TEMPORARY TABLE nome_da_tabela ( ... ) [ ON COMMIT { DELETE | PRESERVE } ROWS ]
```

A cláusula `ON COMMIT` especifica se a tabela temporária deve ou não ser esvaziada quando o comando `COMMIT` é executado. Se a cláusula `ON COMMIT` for omitida, o SQL92 especifica como padrão `ON COMMIT DELETE ROWS`. Entretanto, o comportamento do PostgreSQL é sempre `ON COMMIT PRESERVE ROWS`.

"Restrição" `NULL`

A "restrição" `NULL` (na verdade uma não restrição) é uma extensão do PostgreSQL ao SQL92, incluída para manter a compatibilidade com alguns outros SGBDRs (e por simetria com a restrição `NOT NULL`). Sendo que este é o padrão para qualquer coluna, sua presença é simplesmente informativa.

Asserções

Uma asserção (assertion) é um tipo especial de restrição de integridade e compartilha o mesmo espaço de nomes como outras restrições. Entretanto, uma asserção não é necessriamente dependente de uma tabela em particular como as retrições são, por isso o SQL92 fornece a declaração `CREATE ASSERTION` como uma forma alternativa para definir restrições:

```
CREATE ASSERTION nome CHECK ( condição )
```

O PostgreSQL não implementa asserções atualmente.

Herança

Heranças múltiplas através da cláusula `INHERITS` é uma extensão da linguagem do PostgreSQL. O SQL99 (mas não o SQL92) define herança única utilizando uma sintaxe diferente e semânticas diferente. O estilo de herança do SQL99 ainda não é suportado pelo PostgreSQL.

Identificadores do Objeto (Object IDs)

O conceito de OIDs (identificadores de objeto) do PostgreSQL não é padrão.

Consulte também

ALTER TABLE , DROP TABLE

Prev CREATE SEQUENCE | Home Up | Next CREATE TABLE AS