

Re: Unique PK across two tables?

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-11/1966.html>

From: Richard R (*rrose_at_informsnospam.co.uk*)

Date: 11/08/04

Date: Mon, 8 Nov 2004 10:40:15 -0000

Joe,

This example is neater than the example I use for teaching in my organisation.

Aside from being good example of type & sub-type , it also uses CASCADEs properly, which is something I've struggled to find a good example of.

Mind if I pinch it?

Richard

"Joe Celko" <jcelko212@earthlink.net> wrote in message news:einM5dUxEHA.2788@TK2MSFTNGP15.phx.gbl...

> >> *Is it possible to have two tables where the primary keys are unique across both of them, ..<<*

>

> *Here is a "cut & paste" that the regulars have seen before:*

>

> ===

> *The classic scenario calls for a root class with all the common attributes and then specialized sub-classes under it. As an example, let's take the class of Vehicles and find an industry standard identifier (VIN), and add two mutually exclusive sub-classes, Sport utility vehicles and sedans ('SUV', 'SED').*

>

> *CREATE TABLE Vehicles*
> *(vin CHAR(17) NOT NULL PRIMARY KEY,*
> *vehicle_type CHAR(3) NOT NULL*
> *CHECK(vehicle_type IN ('SUV', 'SED')),*
> *UNIQUE (vin, vehicle_type),*
> *..);*

>

> *Notice the overlapping candidate keys. I then use a compound candidate key (vin, vehicle_type) and a constraint in each sub-class table to assure that the vehicle_type is locked and agrees with the Vehicles table. Add some DRI actions and you are done:*

>

```
> CREATE TABLE SUV
> (vin CHAR(17) NOT NULL PRIMARY KEY,
> vehicle_type CHAR(3) DEFAULT 'SUV' NOT NULL
> CHECK(vehicle_type = 'SUV'),
> UNIQUE (vin, vehicle_type),
> FOREIGN KEY (vin, vehicle_type)
> REFERENCES Vehicles(vin, vehicle_type)
> ON UPDATE CASCADE
> ON DELETE CASCADE,
> ..);
>
> CREATE TABLE Sedans
> (vin CHAR(17) NOT NULL PRIMARY KEY,
> vehicle_type CHAR(3) DEFAULT 'SED' NOT NULL
> CHECK(vehicle_type = 'SED'),
> UNIQUE (vin, vehicle_type),
> FOREIGN KEY (vin, vehicle_type)
> REFERENCES Vehicles(vin, vehicle_type)
> ON UPDATE CASCADE
> ON DELETE CASCADE,
> ..);
>
> I can continue to build a hierarchy like this. For example, if I had a
> Sedans table that broke down into two-door and four-door sedans, I could
> a schema like this:
>
> CREATE TABLE Sedans
> (vin CHAR(17) NOT NULL PRIMARY KEY,
> vehicle_type CHAR(3) DEFAULT 'SED' NOT NULL
> CHECK(vehicle_type IN ('2DR', '4DR', 'SED')),
> UNIQUE (vin, vehicle_type),
> FOREIGN KEY (vin, vehicle_type)
> REFERENCES Vehicles(vin, vehicle_type)
> ON UPDATE CASCADE
> ON DELETE CASCADE,
> ..);
>
> CREATE TABLE TwoDoor
> (vin CHAR(17) NOT NULL PRIMARY KEY,
> vehicle_type CHAR(3) DEFAULT '2DR' NOT NULL
> CHECK(vehicle_type = '2DR'),
> UNIQUE (vin, vehicle_type),
> FOREIGN KEY (vin, vehicle_type)
> REFERENCES Sedans(vin, vehicle_type)
> ON UPDATE CASCADE
> ON DELETE CASCADE,
> ..);
>
> CREATE TABLE FourDoor
> (vin CHAR(17) NOT NULL PRIMARY KEY,
> vehicle_type CHAR(3) DEFAULT '4DR' NOT NULL
```

> CHECK(vehicle_type = '4DR'),
> UNIQUE (vin, vehicle_type),
> FOREIGN KEY (vin, vehicle_type)
> REFERENCES Sedans (vin, vehicle_type)
> ON UPDATE CASCADE
> ON DELETE CASCADE,
> ..);
>
> The idea is to build a chain of identifiers and types in a UNIQUE()
> constraint that go up the tree when you use a REFERENCES constraint.
> Obviously, you can do variants of this trick to get different class
> structures.
>
> If an entity doesn't have to be exclusively one subtype, you play with
> the root of the class hierarchy:
>
> CREATE TABLE Vehicles
> (vin CHAR(17) NOT NULL,
> vehicle_type CHAR(3) NOT NULL
> CHECK(vehicle_type IN ('SUV', 'SED')),
> PRIMARY KEY (vin, vehicle_type),
> ..);
>
> Now start hiding all this stuff in VIEWS immediately and add an INSTEAD
> OF trigger to those VIEWS.
> =====
>
> You also have some serious problems when you confuse an OID from your OO
> background with a key in SQL. Proper keys are not generated so much as
> discovered if your schema is designed correctly.
>
> --CELKO--
> Please post DDL, so that people do not have to guess what the keys,
> constraints, Declarative Referential Integrity, datatypes, etc. in your
> schema are. Sample data is also a good idea, along with clear
> specifications.
>
>
> *** Sent via Developersdex <http://www.developersdex.com> ***
> Don't just participate in USENET...get rewarded for it!