

# Re: Table Design Advice

---

*Source:*

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.setup/2006-02/msg00220.html>

---

- *From:* "Terry Holland" <Terry.Holland@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
  - *Date:* Fri, 24 Feb 2006 02:35:33 -0800
- 

"David Portas" wrote:

"Terry Holland" <TerryHolland@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message  
<news:8D2407FF-5B67-4C22-A49E-6B307AE4820E@xxxxxxxxxxxxxxxxxxxx>

Im designing a database application that will be built using a SQL Server backend with an ASP.Net front end

My application needs to store data on a variety of different scaffold structures. The different types of structures have different properties. For example an Independent structure will have the following properties:  
Length (Single)  
Width (Single)  
Height (Single)  
NumberOfLifts (Integer)

A Protection will have the following properties  
Length (Single)  
Width (Single)  
WidthOfAttachedStructure (Single)  
BackBoarding (Boolean)  
LiftHeight (Single)

There are thirteen different structure types all having a different set of properties. All of the structures have some common properties these are  
Qty (Integer)  
LabourCost (Currency)  
MaterialCost (Currency)

A project consists of any number of different types of structure. So, for example, a project could have 2 related Independent records, 5

## Re: Table Design Advice

Protection  
Fans and so on.

Im wondering whether to create a Structure table with the following fields

Structure  
=====

StructureID
Qty (Integer)
LabourCost (Currency)
MaterialCost (Currency)

And then create a table for each of the different structure types.

The Independent table would have the following fields and would have a one-to-one relationship to the Structure table

Independent  
=====

StructureID (Long)
Length (Single)
Width (single)
Height (Single)
NumberOfLifts (Byte)

and Protection table would have the following fields and would have a One-to-one relationship to the Structure table

ProtectionFan  
=====

StructureID (Long)
Length (single)
Width (Single)
WidthOfAttachedStructure (Single)
BackBoarding (Boolean)
LiftHeight (Single)

Would this be a worthy setup or should I have something more like this.  
A Structure table containing all of the common fields and a StructureProperties table containing the following fields and having a one-to-many relationship with the Structure table

StructureProperties  
=====

PropertyID (Long)
StructureID (Long)
PropertyName (Text)
PropertyDataType (Byte)
PropertyValue (Text)

## Re: Table Design Advice

This table could then contain records

PropertyID: 1  
StructureID: 1  
PropertyName: Length  
PropertyDataType: 1 (single)  
PropertyValue: 100

PropertyID: 2  
StructureID: 1  
PropertyName: Width  
PropertyDataType: 1 (single)  
PropertyValue: 1.3

PropertyID: 3  
StructureID: 1  
PropertyName: Height  
PropertyDataType: 1 (single)  
PropertyValue: 12

PropertyID: 4  
StructureID: 1  
PropertyName: NumberOfLifts  
PropertyDataType: 1 (integer)  
PropertyValue: 6

This way strikes me as being more normalised, but more complicated to program and perhaps with some performance issues.

Another option is to have a single StructureProperties table containing a field for each of the properties that occur in all of my structure types ie

StructureProperties

=====

PropertyID (Integer) (PK)
StructureID (Integer) (FK)
StructureType
Length (Single)
Width (single)
Height (Single)
NumberOfLifts (Byte)
Length (single)
Width (Single)
WidthOfAttachedStructure (Single)
BackBoarding (Boolean)
LiftHeight (Single)
....

Using this approach I would need to include approx 50 columns to cover all

## Re: Table Design Advice

structure properties but each structure type would only use approx 10 of the fields (the ones that are relevant to the structure type). This is a solution that was offered in the AccessTableDesign newsgroup but it strikes me as being quite wasteful and un-normalised – though it does seem to be the easiest solution

I'd appreciate some opinions on this

Here's a different example but hopefully it will give you the idea. Notice that each product can only be of one or other type – the constraints prevent any anomalies between the multiple type tables.

BTW. Are you sure you are using SQL Server? Your datatypes are not valid ones in SQL.

```
CREATE TABLE products
(sku_code INTEGER NOT NULL
CONSTRAINT pk_products PRIMARY KEY,
product_type CHAR (1) NOT NULL /* Book, CD or DVD */
CONSTRAINT ck1_products CHECK (product_type IN ('B','D')),
product_title VARCHAR (50) NOT NULL
CONSTRAINT ak1_products UNIQUE,
CONSTRAINT ak2_products UNIQUE (sku_code,product_type));
```

```
CREATE TABLE books
(sku_code INTEGER NOT NULL
CONSTRAINT pk_books PRIMARY KEY,
CONSTRAINT fk_books_products
FOREIGN KEY (sku_code,product_type)
REFERENCES products (sku_code,product_type),
product_type CHAR (1) NOT NULL /* = Book ONLY */
DEFAULT ('B')
CONSTRAINT ck1_books CHECK (product_type = 'B'),
isbn CHAR (10) NOT NULL
CONSTRAINT ak1_books UNIQUE);
```

```
CREATE TABLE dvds
(sku_code INTEGER NOT NULL
CONSTRAINT pk_cds PRIMARY KEY,
CONSTRAINT fk_dvds_products
FOREIGN KEY (sku_code,product_type)
REFERENCES products (sku_code,product_type),
product_type CHAR (1) NOT NULL /* = DVD ONLY */
DEFAULT ('D'))
```

Re: Table Design Advice

```
CONSTRAINT ck1_dvds CHECK (product_type = 'D'),  
running_time INTEGER NOT NULL);
```

--

David Portas, SQL Server MVP

Whenever possible please post enough code to reproduce your problem. Including CREATE TABLE and INSERT statements usually helps. State what version of SQL Server you are using and specify the content of any error messages.

SQL Server Books Online:

[http://msdn2.microsoft.com/library/ms130214\(en-US,SQL.90\).aspx](http://msdn2.microsoft.com/library/ms130214(en-US,SQL.90).aspx)

--