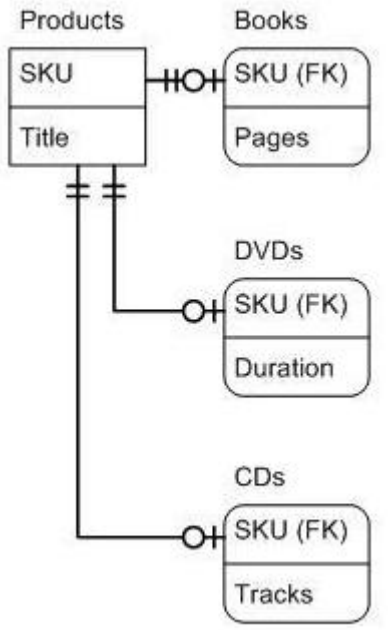
Distributed Keys and Disjoint Subtypes

Entity subtyping is a very common data modelling scenario that gets extensive coverage in books on relational design but it isn't necessarily well supported by SQL DBMSs. The following is a simple technique I have found useful for implementing some kinds of entity subtypes. Specifically it applies to the case of *Disjoint Subtypes* that require what Hugh Darwen calls a *Distributed Key* – a uniqueness constraint spanning multiple tables.

First, here's an example of a disjoint subtype problem. A retailer sells three types of product: Books, CDs and DVDs. In their product database all these product types have two attributes in common: SKU and Title. They also have other attributes that are unique to one type or the other. The unique attributes are: Number of Pages (for Books), Number of Tracks (for CDs) and Duration (DVDs). Here's an E/R model prototype. SKU is the common candidate key and is also the identifying foreign key in each of the referencing tables:



So far so good. But the constraints implied by this E/R diagram are not enough to enforce the business rule that Books, CDs and DVDs are *disjoint* subtypes – i.e. that no SKU can be of more than one type. Most textbook examples don't seem to address the problem of how to enforce such a constraint in SQL. The solution I like to use is to add a ProductType attribute to both the supertype and the subtypes and then include a compound foreign key on (SKU, ProductType). Here are the additional constraints for the Books table:

```
ALTER TABLE Books ADD CONSTRAINT Books_Products_ProductType_FK
FOREIGN KEY (SKU, ProductType) REFERENCES Products (SKU, ProductType);

ALTER TABLE Books ADD CONSTRAINT Books_ProductType_CK
CHECK (ProductType ='B');
```

Similar constraints apply to CDs and DVDs. The following is the DDL for the complete schema. To aid readability I've omitted the constraint names. I've also added some defaults:

```
CREATE TABLE Products
 (SKU INT NOT NULL PRIMARY KEY,
  ProductType CHAR(1) NOT NULL
  CHECK (ProductType IN ('B','C','D' /* Book, CD or DVD */)),
  Title VARCHAR(50) NOT NULL,
  UNIQUE (SKU,ProductType));

CREATE TABLE Books
 (SKU INT NOT NULL PRIMARY KEY,
  ProductType CHAR(1) DEFAULT 'B' NOT NULL, CHECK (ProductType ='B'),
  Pages SMALLINT NOT NULL,
  FOREIGN KEY (SKU,ProductType) REFERENCES Products (SKU,ProductType));

CREATE TABLE CDs
 (SKU INT NOT NULL PRIMARY KEY,
  ProductType CHAR(1) DEFAULT 'C' NOT NULL, CHECK (ProductType ='C'),
  Tracks SMALLINT NOT NULL,
  FOREIGN KEY (SKU,ProductType) REFERENCES Products (SKU,ProductType));

CREATE TABLE DVDs
 (SKU INT NOT NULL PRIMARY KEY,
  ProductType CHAR(1) DEFAULT 'D' NOT NULL, CHECK (ProductType ='D'),
  Duration SMALLINT NOT NULL,
  FOREIGN KEY (SKU,ProductType) REFERENCES Products (SKU,ProductType));
```

That's it. The disjoint requirement is always enforced by the combination of the compound foreign key constraints and check constraints. Although it's extremely simple I think this is worth documenting here because I have never seen such a solution mentioned in print. I don't claim any originality for the above and I know of others who have used it. If anyone is aware of a good reference that describes the same technique then I'd be grateful to hear about it so that I can attribute a source in future.

One other tiny point of interest. A colleague of mine once objected to this design because he said that the table I've called Products violated Boyce-Codd Normal Form. I am confident that he was wrong however - there is no violation of BCNF. In fact this schema comfortably satisfies 5NF. For the moment I'll leave further analysis open to anyone who wants to comment on it here...

--

 See: Halpin - Information Modeling and Relational Databases; Pascal - Practical Issues in Database Management; Date - An Introduction to Database Systems.

In the final paragraph above I was wrong to ignore the dependency {}->{ProductType} in the Books, CDs, DVDs tables. At the time I didn't recognise that this dependency on the empty set would be a violation of 2NF.

DP, 2016

Published 08 January 2007 22:35 by David.Portas
Filed under: SQL, Data Modelling