Zeitabhängige Stammdaten (Period Master Data Pattern)

Absicht

Eine Entität enthält Attribute, die zeitbezogen gespeichert werden müssen.

Problem

Ein Beispiel: Ein Programm zur Rechnungserstellung soll unterschiedliche Steuersätze behandeln können. Bei jedem Produkt wird ein Steuersatz hinterlegt. Ab einem definierten Datum soll ein neuer MWST-Satz gelten. Das Rechnungsprogramm soll je nach Rechnungsdatum den aktuellen MWST-Satz ermitteln.

Forces

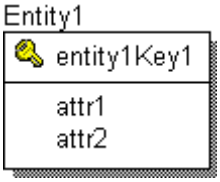
In einer Entität existiert mindestens ein Attribut, dass zeitbezogen gespeichert werden muss. D.h. es müssen für unterschiedliche Zeiträume auch unterschiedliche Werte für das Attribut gespeichert werden.

Lösung

Die Lösung besteht darin, dass für die zeitabhängigen Daten eine neue Entität erstellt wird, die alle zeitabhängigen Attribute enthält.
Damit enthält die ursprüngliche Entität nur diejenigen Attribute, die nicht zeitabhängig sind. Die Entitäten werden durch eine identifizierende Beziehung miteinander verbunden. Die zeitabhängige Entität enthält einen neuen Schlüssel, der das Datum enthält, ab dem der Datensatz gültig ist.

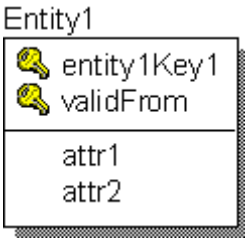
Diese Lösung kann auch auf Grund der zweiten Normalform hergeleitet werden. Betrachten wir die Lösung schrittweise.

Abbildung 1 - Ausgangssituation



Die Entity1 enthält ein Attribut, dass zeitabhängig ist (siehe Abbildung 1). Also muss ein zweiter Primäreschlüssel eingeführt werden, damit mehrere Version von attr2 gespeichert werden können. Dies führt zunächst zu folgender Zwischenlösung.

Abbildung 2 - Lösung mit 2 Schlüssel

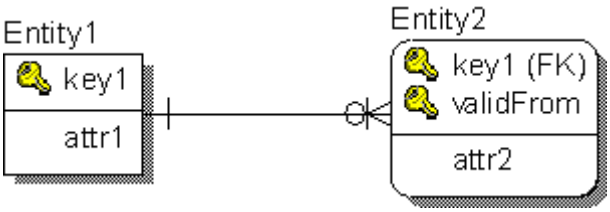


Nun entspricht die Entität aber nicht mehr der zweiten Normalform, weil attr1 nicht zeitabhängig ist, und somit auch nicht abhängig vom identifizierenden Attribut validFrom. Eine Verletzung der zweiten Normalform wird aufgehoben, in dem eine neue Entität erstellt wird in der diejenigen Attribute ausgelagert werden, die nicht vom ganzen Schlüssel abhängen. Dies führt uns zur Lösung.

Struktur

In Abbildung 3 wird die Lösung dargestellt. Die *Entity2*, die durch eine identifying Relationaship mit *Entity1* in Beziehung steht, enthält das zeitabhängige Attribut2. Außerdem enthält es einen neuen Schlüssel validFrom, der angibt, ab wann dieser Eintrag gültig ist.

Abbildung 3 Lösung



Beteiligte und Verantwortlichkeiten

Entity1

Entity1 enthält alle Attribute, die nicht zeitabhängig gespeichert werden müssen.

Entity2

Entity1 enthält alle Attribute, die zeitabhängig gespeichert werden müssen.

ClientEntity

ClientEntity enthält als Verweis auf die *Entity1* ein Foreign Key. Dieser FK verweist je nach Bedarf auf *Entiy1* oder *Entity2*.

Begründung

- Durch die zeitabhängigen Felder wird die zweite Normalform nicht verletzt, da sowohl zeitabhängige als auch nicht zeitabhängige Felder voll funktional vom jeweiligen Schlüssel abhängen.

Strategien

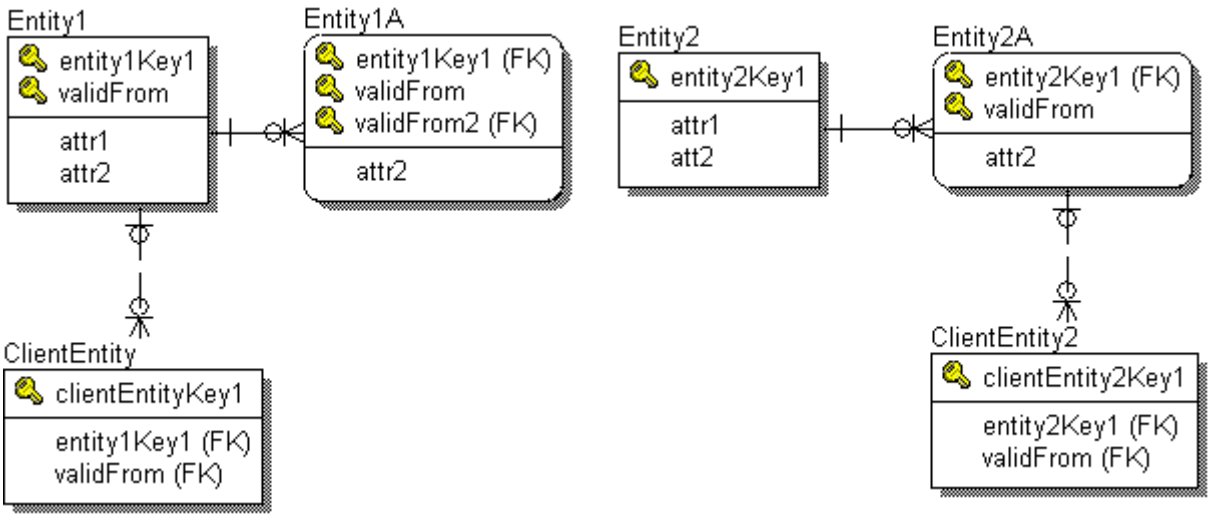
Ermittlung eines zeitabhängigen Attributs

SELECT mwst, steuerSatz FROM MWST_ZEIT
WHERE mwst = ?
and validFrom >= '20.05.2004'
and validTo < '20.05.2004'

Relationen zur zeitabhängigen Entität

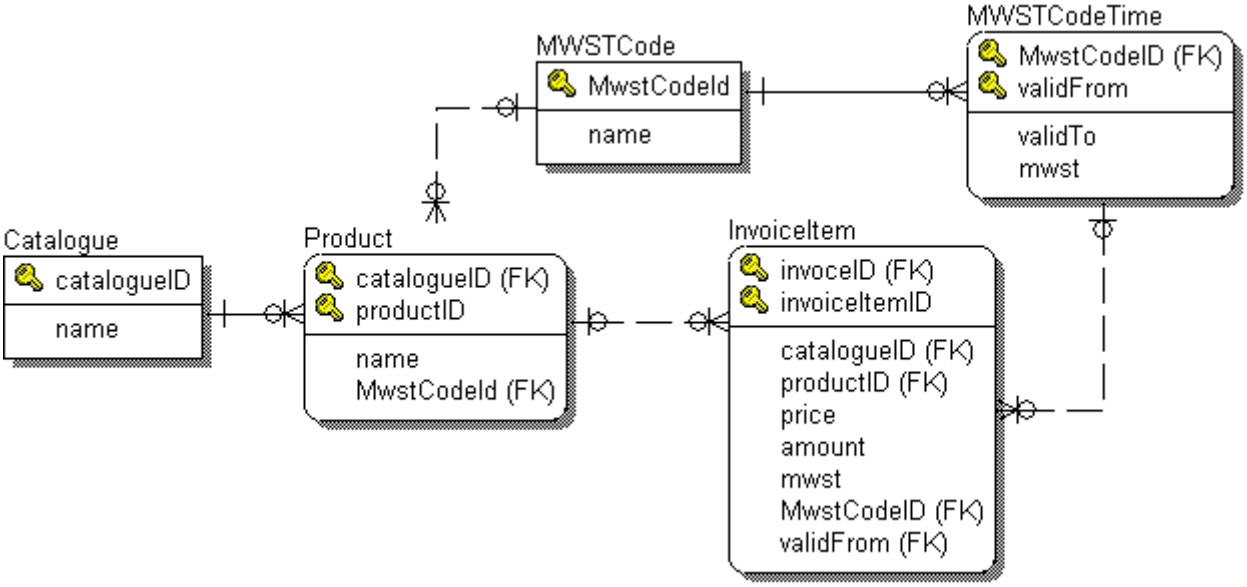
Es stellt sich nun die Frage, in welchen Fällen eine Relation zur Entität mit Zeitbezug und wann zur Entität ohne Zeitbezug erstellt werden soll.

Abbildung 4 Client benutzt entweder Entität mit oder ohne Zeitbezug



Dies soll an Hand des folgenden Beispiels erläutert werden.

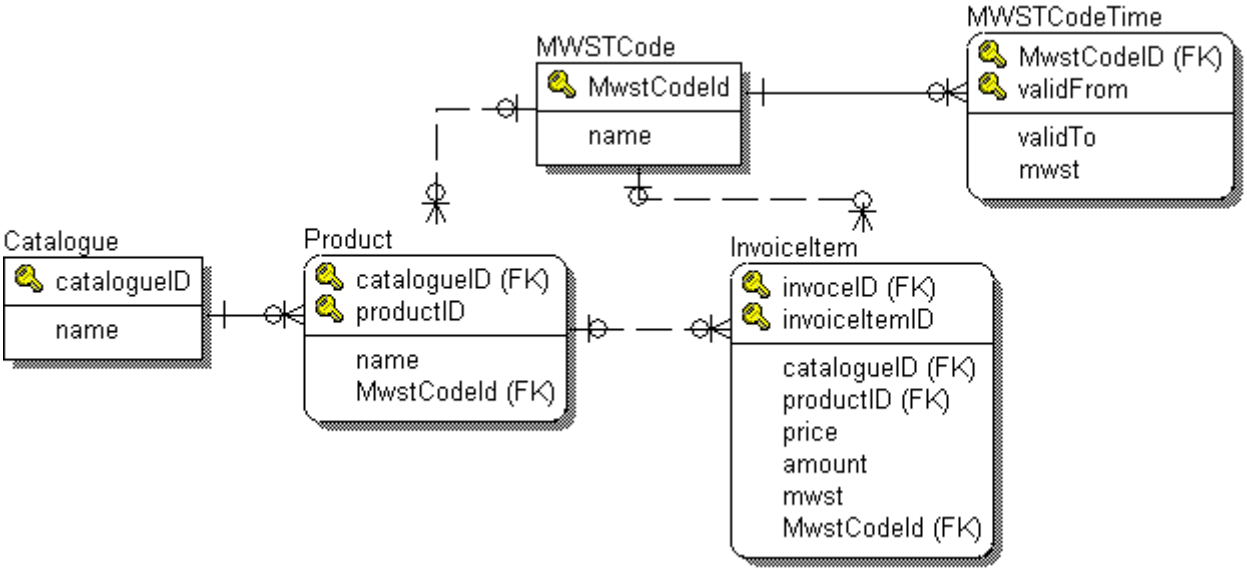
Abbildung 5 - Beispiel



Sowohl Product als auch *InvoiceItem* sind Client der zeitabhängigen Entität *MWSTCode*. Die Entität *Product* hält eine Beziehung zu *MWSTCode*. Damit wird nur festgelegt, dass z.B. das Produkt "Bananen" den MWSTCode "Lebensmittel" verwenden soll. Das Programm zur Rechnungserstellung muss den aktuellen MWST-Satz an Hand des Rechnungsdatums bzw. des Buchungsdatums ermitteln. Die Tabelle *InvoceItem* ist eine typische Bewegungsdatentabelle. Sie enthält Daten über verrechnete Produkte, die in einer Rechnung zusammengefasst werden. (Die Entität *Invoice* fehlt in der Abbildung)
Vorgabe für diese Tabelle ist es, dass die Nachvollziehbarkeit gewährleistet werden muss. d.h. es muss zu einem späteren Zeitpunkt eindeutig festgestellt werden können mit welchem MWSTSteuersatz dieses Produkt abgerechnet wurde. Deshalb wird hier direkt auf die zeitabhängige Entität verwiesen, in der der MWST-Satz gespeichert ist.

Es wäre in diesem Beispiel jedoch folgende Lösung praktikabel.

Abbildung 6 - Variante 2 des Beispiels

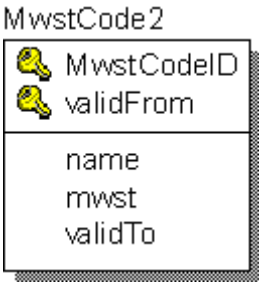


In diesem Fall wird in *InvoideItem* nur der MWSTCode gespeichert. Damit ist man von Änderungen in der *MWSTCodeTime* unabhängig. In diesem Fall müssen jedoch alle Daten, die aus der zeitabhängigen Entität ermittelt werden in den Bewegungsdaten gespeichert werden. In diesem Fall das Feld mwst. Was zunächst Redundant aussieht ist in Wirklichkeit die sicherste Methode der Nachvollziehbarkeit.

Eine Entität mit validFrom im Primärschlüssel

Die vermeintlich einfachere Lösung um zeitabhängige Daten zu speichern besteht zunächst darin die Entität *MWST* mit einem zusätzlichen Primary Key zu versehen. Dadurch können mehrere Datensätze für einen MWST-Code. (z.B. Lebensmittel) erstellt werden. Die Entität *MWSTCode2* enthält nun zwei identifizierende Attribute.

Abbidlung 7 - Lösung mit einer Entität aber zwei Primärschlüssel



Die Entität entspricht jedoch nicht der zweitern Normalform.
Aus der Problemstellung geht hervor, dass nur das Attribut *mwst* zeitabhängig gespeichert werden muss. *name* ist nur abhängig vom MwstCode. Der zweite Normalform besagt, dass jedes Attribut voll funktional vom gesamten Schlüssel abhängig sein muss. Dies gilt hier für mwst, aber nicht für name. Überarbeitet man das Modell, kommt man automatisch wieder auf das "Period Master Data" Pattern.

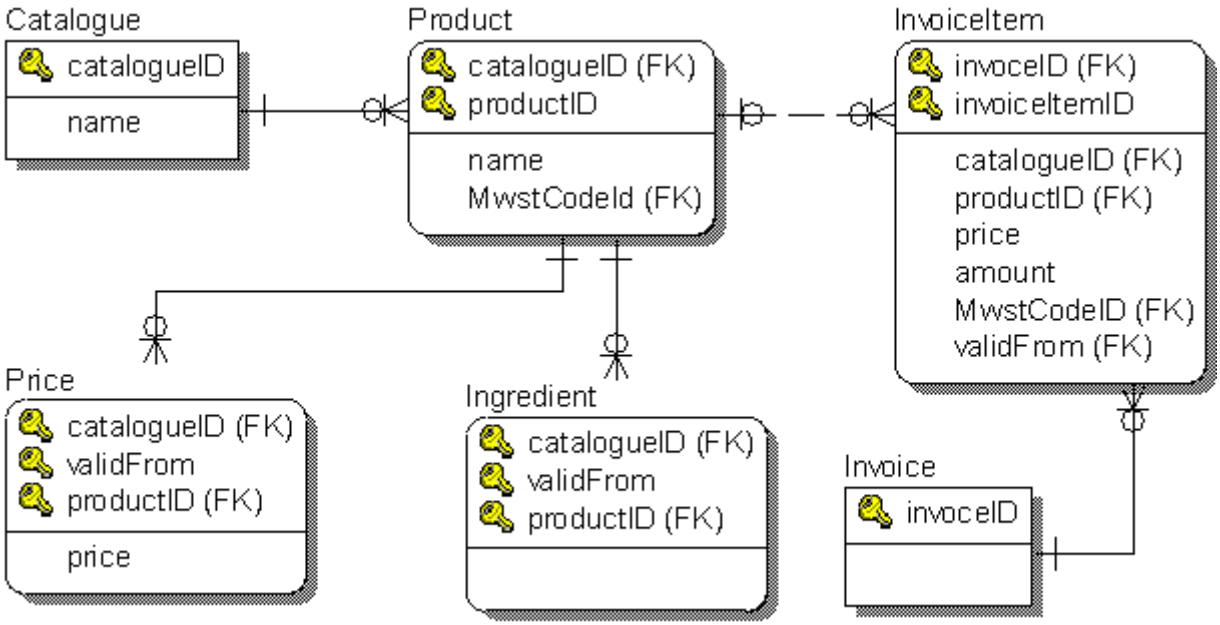
Damit soll nur verdeutlicht werden, dass die hier beschriebene Strategie, nämlich der entsprechenden Entität einfach eine zweiten Schlüssel zu verpassen, keine Lösung darstellt, weil damit die zweite Normalform verletzt wird.

Diese Variante ist also nur eine sinnvolle Lösung, wenn alle Attribute der Entität auch vom Gültigkeitsdatum abhängen.

Beispiele

Produktkatalog Beispiel

Abbildung 8 - Beispiel Produktkatalog



In diesem Beispiel ist das Produkt die Entität, für die zeitabhängige Daten gespeichert werden sollen.

Der Produktkatalog besteht aus verschiedenen Katalogen. z.B. Ersatzteile, Elektro, Lebensmittel usw.. In der *Produkt* Entity werden nur Attribute gespeichert, die funktional vom Produkt abhängen. Attribute, die zeitabhängig gespeichert werden müssen, wie z.B. Preise, werden in Kindentitäten von *Produkt* abgelegt. Es gibt in diesem Beispiel zwei solche Kindentitäten: Preise und Bestandteile.
Neue Preise bewirken lediglich neue Einträge in der Tabelle *Price*. Änderunen an den Bestandteilen bewirken wiederum neue Einträge in der Tabelle *Ingredient*.
Die Tabelle *InvoiceItem*, in der die in Rechnung gestellten Produkte aufgeführt sind, enthält das Produkt und den ermittelten Preis. Dies ist notwendig, damit auch nach möglichen Änderung an den Stammdatentabellen auf jeden Fall der in Rechnung gestellte Betrag für das Produkt ermittelt werden kann.

Verwandte Muster

- *Validity Period* (*Zeitliche Gültigkeit*). Dieses Pattern wird angewandt, wenn nicht mehrere Versionen von Daten gespeichet werden müssen, sondern nur die Gültigkeit der Entität beschränkt werden soll.