Vererbung in Datenbanken

Wechseln zu:Navigation, Suche

Dieser Artikel erfüllt die GlossarWiki-Qualitätsanforderungen **nur teilweise**:

Korrektheit: 3 (zu größeren Teilen überprüft)	**Umfang**: 3 (einige wichtige Fakten fehlen)	**Quellenangaben**: 2 (wichtige Quellen fehlen)	**Quellenarten**: 2 (befriedigend)	**Konformität**: 2 (befriedigend)

Es gibt verschiedene Möglichkeiten wie man Vererbungen, sogenannte IS-A-Beziehungen, in Datenbanken realisieren kann. Die unterschiedlichen Implementierungsmöglichkeiten bringen jeweils Vor- und Nachteile, insbesondere beim Zugriff auf die jeweiligen Tabellen, mit sich.

Inhaltsverzeichnis

1 Manuelle Definition

Voraussetzung: `f isa e`, `f` erbt alle Attribute von `e` und kann weitere Attribute besitzen.

1.1 Implementierungsvarianten Relationenschema

1. Möglichkeit

```
e: id, e1, e2
f: id, e1, e2, f1, f2 (id kommt nicht in e vor)
```

Hierbei wird jedes Objekt in einer seperaten Tabelle gespeichert.

2. Möglichkeit

```
e: id, e1, e2
f: id, f1, f2 (id -> E: id)
```

Hierbei ist die Auflistung aller Objekte der Tabelle e einfacher, für alle Objekte und f und deren zugehörigen von e geerbten Attribute ist ein einfacher Join möglich. Die zweite Variante ist die sauberste aller Implementierungsmöglichkeiten und wird empfohlen.

3. Möglichkeit

```
e: id, e1, e2, f1*, f2*
```

Es ist theoretisch möglich, für die gesamte Vererbungs-Hierachie nur eine einzige Tabelle zu definieren. Diese enthält dann alle Attribute der Basisklasse sowie aller Subklassen. Die Attribute der Subklassen dürfen NULL werden.

Der große Nachteil dieser Variante: Es können Inkonsistenzen auftreten. Attribute wie f1 und f2, welche bei einem Objekt der Klasse f nicht undefiniert sein dürften; können nun den WERT NULL annehmen. Denkbar wäre z.B. dass f1 mit einem richtigen Wert initialisiert wird, während f2 leer bleibt.

1.2 Zugriff

1. Möglichkeit

Alle Objekte der Art e:

```
SELECT id, e1, e2 FROM e
UNION
SELECT id, e1, e2 FROM F
```

Alle Informationen über f-Objekte:

```
SELECT id, e1, e2, f1, f2 FROM f
```

2. Möglichkeit

Alle Objekte der Art e:

```
SELECT id, e1, e2 FROM e
```

Alle Informationen über f-Objekte:

```
SELECT id, e1, e2, f1, f2 FROM e, f
WHERE e.id = f.id
```

2 Direkte Vererbung in SQL

Seit SQL:1999 wird einfache Attributvererbung auch direkt unterstützt. Man muss sich nicht mehr darum kümmern welche Attribute in welche Tabelle gehören. Auch ist der Zugriff ohne zusätzliche JOIN- oder UNION-Operationen möglich.

Jedoch unterstützen einige Datenbank-Management-Systeme Vererbung gar nicht oder nicht standard-konform. Im Folgenden wird deshalb der SQL-Standard sowie PostgreSQL behandelt.

2.1 Erstellung

SQL-Standard:

```
CREATE TABLE e(...);
CREATE TABLE f UNDER e(...);
```

PostgreSQL:

```
CREATE TABLE e(...);
CREATE TABLE f(...) inherits (e)
```

Hierbei erbt f alle Attribute und Integritätsbedingungen von e, auch den Primärschlüssel. f kann, abgesehen vom Primärschlüssel, weitere Attribute und Integritätsbedingungen definieren.

2.2 Zugriff

```
SELECT * FROM e
```

Auf diese Art und Weise werden alle Tupel von e einschließlich der Tupel von f(reduziert auf die geerbten Attribute von e) selektiert.

```
SELECT * FROM ONLY(e)
```

Hierbei erhält man ausschließlich die Tupel von e, nicht aber Tupel von Subklassen wie f.

3 Quellen

Kowarschick, "Multimedia-Datenbanksysteme", Sommersemester 2009, Hochschule Augsburg]

Kategorien:
HowTo
SQL