Referentielle Integrität – Ansätze zur Symmetrie in SQL3

Theo Härder
Universität Kaiserslautern
FB Informatik, Postfach 3049
6750 Kaiserslautern
Tel.: 0631/205-4030
e-mail: haerder @informatik.uni-kl.de

ÜBERBLICK

Das Relationenmodell garantiert seinen Benutzern die Einhaltung der Relationalen Invarianten - also der Entitätsintegrität und Referentiellen Integrität. Nach dem SQL2-Standard werden durch das Datenbanksystem eine Reihe von referentiellen Aktionen unterstützt, die bei Aktualisierungsoperationen, die die Referentielle Integrität gefährden, diese automatisch wiederherstellen. Weiterhin erlaubt die MATCH-Klausel die Kontrolle erweiterter referentieller Beziehungen, wobei auch unvollständig definierte Fremdschlüsselwerte systemseitig interpretiert werden können. Nach dem SQL3-Vorschlag ist nun noch beabsichtigt, eine Art symmetrischer Referentieller Integrität durch die PENDANT-Option vorzusehen. Dieser Aufsatz beschreibt die sich dabei ergebenden Möglichkeiten und diskutiert die Komplexität ihres Einsatzes. Er untersucht den Wartungsaufwand bei Aktualisierungsoperationen und schlägt einige Methoden vor, wie die Einhaltung der PENDANT-Bedingung dabei überprüft werden kann. Schließlich wird der Zugewinn an Modellierungsmächtigkeit, der sich durch die PENDANT-Option ergibt, kritisch beleuchtet.

1. Einführung

In der SQL-Standardisierung spielt die Integritätssicherung in Relationalen Datenbanksystemen eine wichtige Rolle. In SQL89 wurden die Referentielle Integrität und die Entitätsintegrität [Da81] verbindlich festgelegt, wodurch diese beiden Grundkonzepte - auch Relationale Invarianten genannt - erstmalig in einem Sprachstandard des Relationenmodells aufgenommen wurden. Für einen Primärschlüssel (Entitätsintegrität), der aus ein oder mehreren Attributwerten bestehen kann, hat das Datenbanksystem Eindeutigkeit und Definiertheit zu garantieren; für Schlüsselkandidaten, durch die UNIQUE-Option spezifiziert, ist ebenso die Eindeutigkeit der Werte zu gewährleisten. Zur Einhaltung der Referentiellen Integrität, die zwischen Primärschlüssel (Schlüsselkandidat) und Fremdschlüssel definiert ist, muß dafür gesorgt werden, daß für jeden Fremschlüsselwert stets ein gleicher Wert im zugehörigen Primärschlüssel (oder Schlüsselkandidat) existiert.
Im SQL-Standardisierungsvorschlag [SQL2, Sh90] wurden Maßnahmen zur Wahrung der Referentiellen Integrität bei Änderungsoperationen (ON DELETE, ON UPDATE usw.) eingeführt, die als Referentielle Aktionen (SET NULL, CASCADE usw.) im DB-Schema spezifiziert werden können. Die Überwachung dieser Integritätsbedingungen und die automatische Nachführung konsistenter DB-Zustände - insbesondere bei Löschungen oder Änderungen des Primärschlüssels - erzeugen einen nicht unerheblichen Überprüfungsaufwand durch das Datenbanksystem [HR91]. Zusätzliche und teilweise sehr spezielle Aspekte der Referentiellen Integrität lassen sich durch die MATCH-Klausel beschreiben. Sie erlaubt die Definition und Überwachung von “erweiterten Beziehungen” zwischen Relationen über Schlüsselkandidaten und Fremdschlüssel, die sich durch partielle Undefiniertheit von Werten ergeben können. Bei zusammengesetzten Schlüsseln erlaubt die Option MATCH PARTIAL teilweise definierte Fremdschlüssel, die von Datenbanksystemen zu interpretieren sind. In [HR92] wurde gezeigt, daß die Wahrung solcher Integritätsbedingungen prohibitiven Aufwand annehmen kann.

Die bisher diskutierten Konzepte der Referentiellen Integrität waren unsymmetrisch in dem Sinne, daß sie Einschränkungen nur für die Richtung Fremdschlüssel-Schlüsselkandidat festlegten und diese überwachten. Mit der PENDANT-Option wird in SQL3 der Versuch gemacht, eine symmetrische Behandlung der Referentiellen Integrität zu spezifizieren und systemseitig kontrollieren zu lassen. In diesem Aufsatz untersuchen wir die Einsatzmöglichkeiten der PENDANT-Option und versuchen, ihre Semantik zu ergründen. Zur Ermittlung der Kosten zur Überprüfung der PENDANT-Bedingung diskutieren wir die einzelnen Schritte zur Abwicklung der Aktualisierungsoperationen auf einer der Sohn- oder Vater-Relationen. Die PENDANT-Bedingung verlangt dabei die Feststellung, daß noch mindestens ein Sohn-Tupel (aus einer der verschiedenen Sohn-Relationen) für das betreffende Vater-Tupel vorhanden ist. Da sich hier der Prüfbereich auf mehrere Sohn-Relationen erstrecken kann, kommt neben den Zugriffskosten auch der Sperrproblematik erhebliche Bedeutung zu.

Die PENDANT-Option sollte die Modellierungsmächtigkeit von SQL erhöhen. Diese Fragestellung wird anhand wichtiger Modellierungsaufgaben und typischer Beispiele diskutiert. Ebenso wird ein Vergleich mit dem Konzept der Kardinalitätsrestriktion vorgenommen.

Die verschiedenen Untersuchungen und Erörterungen sollen helfen, die Semantik der PENDANT-Option genau zu verstehen und ihre praktische Tauglichkeit besser zu bewerten. Außerdem sollen sie es ermöglichen, die erfor-

derlichen Implementierungsmaßnahmen abzuschätzen und den zu erwartenden Aufwand zur Überprüfung der PENDANT-Bedingung in groben Zügen zu klären.

2. Symmetrische Referentielle Integrität

In diesem Abschnitt soll zu klären versucht werden, was im SQL3-Standardisierungsvorschlag unter symmetrischer Referentieller Integrität verstanden wird bzw. was mit der PENDANT-Option bewirkt werden kann. Für diese Fragestellung nehmen wir zunächst an, daß Primär- und Fremdschlüssel der beteiligten Relationen durch einfache Attribute definiert sind. Unser Referenzbeispiel läßt sich dafür formal wie folgt beschreiben.

```
CREATE DOMAIN V1_typ AS ...
. . .
CREATE DOMAIN Snk_typ AS ...
CREATE   TABLE V (
     V1     V1_typ PRIMARY KEY,
     SK1    SK1_typ UNIQUE,
     ...
     SKm    SKm_typ UNIQUE,
     ...
     Vp     Vp_typ)

CREATE   TABLE S1 (
     S11    S11_typ, PRIMARY KEY,
     V1     V1_typ,
     ...
     S1m    S1m_typ,
     CONSTRAINT    S1FK FOREIGN KEY (V1)
     REFERENCES PENDANT V (V1)
     ON UPDATE
     ON DELETE ...   )
     ...
CREATE TABLE Sn(
     Sn1    Sn1_typ    PRIMARY KEY,
     SKi    SKi_typ,
     ...
     Snk    Snk_typ
     CONSTRAINT SnFK FOREIGN KEY(SKi)
     REFERENCES PENDANT V (SKi)
     ON UPDATE...
     ON DELETE      )
```

Die Relationen V (Vater) und S1 bis Sn (Sohn-Relationen) haben jeweils einen einfachen Primärschlüssel. V beinhaltet m Schlüsselkandidaten SKi (i=1, ...,m). Die Sohn-Relationen besitzen jeweils einen einfachen Fremdschlüssel, der sich auf V1 oder ein SKi bezieht.

2.1 Wirkungsweise der PENDANT-Option

An diesem Basisbeispiel soll die Wirkung der PENDANT-Option und ihre Auswirkungen auf die Überprüfung der Referentiellen Integrität bei Aktualisierungsoptionen diskutiert werden. Die "einfache" Referentielle Integrität ist unsymmetrisch; für jeden Fremdschlüsselwert wird das Vorhandensein eines entsprechenden Primärschlüsselwertes (oder Schlüsselkandidatenwertes) gefordert. Durch Referentielle Aktionen kann diese Bedingung, die bei Akualisierungsoperationen auf Vater-Tupeln gefährdet ist, automatisch durch das DBVS eingehalten werden.

Die PENDANT-Option dagegen führt eine gewisse Symmetrie bei der Referentiellen Integrität ein. Diese ist jedoch nicht zwischen zwei Relationen (Vater und Sohn) definiert, sondern zwischen der Vater-Relation und allen zugehörigen Sohn-Relationen, für die die PENDANT-Option spezifiziert wurde. Die PENDANT-Option verlangt nun, daß zu jeder Zeit zu einem Vater-Tupel mit gegebenem Primärschlüssel mindestens ein Sohn-Tupel mit passendem Fremdschlüssel aus irgendeiner dieser Sohn-Relationen existiert.

Dieser Sachverhalt soll durch ein Beispiel in Bild 1 verdeutlicht werden, bei dem sich alle Fremdschlüssel auf den Primärschlüssel V1 beziehen:

V (V1, ...)
9
73
45

S1 (S11,	V1, ...)
1,	73
5,	73
9,	99

S2 (S21,	V1, ...)
2,	45
3,	73

S3 (S31,	V1, ...)
1,	99
4,	73
5,	-

Bild 1: PENDANT-Option mit 3 Sohn-Relationen

Ein (voll definierter) Fremdschlüsselwert in einem Sohn-Tupel bildet eine PENDANT-Referenz. In [SQL3] wird die Anzahl der PENDANT-Referenzen aus allen beteiligten Sohn-Relationen auf ein Vater-Tupel als Anzahl der PENDANT-Pfade (#PP) dieses Tupels bezeichnet. Damit läßt sich dann die PENDANT-Bedingung wie folgt definieren:

Für jedes Tupel in V muß die Anzahl der PENDANT-Pfade mindestens 1 (#PP≥1) sein.

In Bild 1 ergeben sich für

- V1= 45 #PP = 1
- V1= 73 #PP = 4
- V1 = 99 #PP = 2

2.2 Weitere Aspekte der PENDANT-Option

Die PENDANT-Option verlangt nicht, daß alle Fremdschlüssel der beteiligten Sohn-Relationen den Primärschlüssel der Vater-Relation referenzieren. Es sind auch Referenzen auf Schlüsselkandidaten (SKi) erlaubt (siehe Schema in Abschnitt 2.1). Im Prinzip könnte jede Sohn-Relation einen anderen Schlüsselkandidaten referenzieren. Die PENDANT-Bedingung besagt nur, daß die Anzahl der PENDANT-Referenzen auf das betreffende Vater-Tupel nicht Null werden darf.

Weiterhin ist der Zusammenhang zwischen Schlüssselinterpretation und PENDANT-Klausel bei zusammengesetzten Schlüsseln zu klären. Das wird im SQL-Standard ganz allgemein durch Spezifikation oder Weglassen der MATCH-Klausel geregelt.

Wird die MATCH-Klausel weggelassen, so wird zwischen Tupeln von V und Si nur dann eine referentielle Beziehung gewartet, wenn die zugehörigen Primär- und Fremdschlüsselwerte vollständig definiert sind und übereinstimmen. Sie bilden jeweils einen PENDANT-Pfad. Unvollständig definierte Fremdschlüsselwerte werden zwar in Si akzeptiert, jedoch nicht hinsichtlich der PENDANT-Option und der Referentiellen Integrität beachtet.

Die Option MATCH FULL erlaubt keine unvollständig definierten Schlüsselwerte; alle Werte müssen vollständig definiert oder vollständig undefiniert sein. Auch hier nehmen nur vollständig definierte Fremdschlüsselwerte an der referentiellen Beziehung teil und bilden einen PENDANT-Pfad.

Die Option MATCH PARTIAL läßt teilweise definierte Fremdschlüssel zu und interpretiert diese auch noch. Auf diese Weise können partiell undefinierte Fremdschlüssel zur Darstellung von Beziehungen herangezogen werden (spezielle n:m-Beziehungen). Da MATCH PARTIAL im Zusammenhang mit der PENDANT-Option nicht erlaubt ist, können partiell spezifizierte Fremdschlüsselwerte nicht für die PENDANT-Beziehung berücksichtigt werden (was sicherlich eine enorme Auswertungskomplexität eingeführt hätte). Folglich lassen sich zusammengesetzte Schlüssel hinsichtlich der Behandlung der PENDANT-Option auf einfache Schlüssel zurückführen.

2.3 Referentielle Integrität und PENDANT-Option

Im allgemeinen Fall können, wie bisher angenommen, für eine Vater-Relation V n Sohn-Relationen Si, für die jeweils die PENDANT-Option spezifiziert wurde, existieren. Diese n Relationen müssen für jedes V-Tupel die PENDANT-Bedingung erfüllen. Da diese Bedingung nicht an einer Relation festgemacht werden kann (außer bei n=1), wird eine gewisse Unübersichtlichkeit eingeführt. Weiterhin sind auf der V-Seite bis zu m+1 Schlüsselwerte an der Bildung der PENDANT-Pfade beteiligt, was die Komplexität des Verständnisses (vor allem bei Ausführung der Referentiellen Aktionen (siehe Abschnitt 3.3)) wesentlich steigert.

Neben den n Si-Relationen können weitere k Sohn-Relationen spezifiziert werden, für die zur V-Relation die Referentielle Integrität, aber nicht die PENDANT-Bedingung eingehalten wird. Für diese k Relationen kann die Klausel MATCH PARTIAL herangezogen werden. Außerdem ist bei Ändern und Löschen die gesamte Menge der Referentiellen Aktionen [{CASCADE | SET NULL | SET DEFAULT | RESTRICT}] anwendbar, während für die Si-Relationen auf die Option SET DEFAULT verzichtet werden muß. Die durch MATCH PARTIAL eingeführte Auswertungs- und Wartungskomplexität bleibt auf die betreffenden referentiellen Beziehungen beschränkt.

Die Verwaltung der PENDANT-Beziehung kann ausschließlich durch Bezug auf die V- und Si-Relationen bewerkstelligt werden und ist unabhängig von den Operationen auf den weiteren k Sohn-Relationen.

3. Wartung der PENDANT-Beziehung

In unserem Modellbeispiel (Bild 2) sind neben den Primärschlüssselbedingungen aller beteiligten Relationen die Referentielle Integrität zwischen S1,..., Sn und V und die PENDANT-Option als zusätzliche Integritätsbedingung zwischen V und S1,..., Sn zu wahren. Dazu seien eine Reihe von Indexstrukturen I_V(V1), I_V(SK1), ... , I_V(SKm), I_{S1}(S11), ... , I_{Sn}(Sn1) (UNIQUE) für alle Primärschlüssel und Schlüsselkandidaten sowie I_{S1} (V1), ... , I_{Sn}(SKm) für alle Fremdschlüssel definiert. Diese Indexstrukturen werden separat von den zugehörigen Relationen angelegt und enthalten neben den Schlüsselwerten TIDs als Verweise auf die Tupeln der Relation.

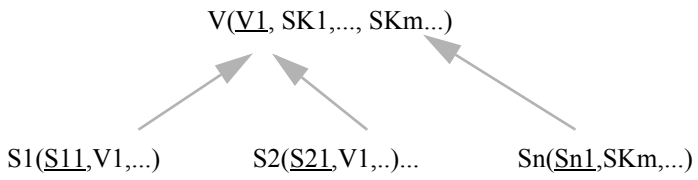


Bild 2: n Sohn-Relationen mit PENDANT-Referenzen auf Primärschlüssel und m Schlüsselkandidaten

Durch Aktualisierungsoperationen in den betreffenden Relationen können die Relationalen Invarianten und die PENDANT-Bedingung verletzt werden. Bei den Relationalen Invarianten sollte ihre Überprüfung zusammen mit der auslösenden Operation vorgenommen werden (IMMEDIATE). Die PENDANT-Option erzwingt in manchen Fällen eine Verzögerung der Überprüfung (DEFERRED). Wird bei der Überprüfung eine Integritätsverletzung erkannt, so wird die auslösende Operation zurückgesetzt, sonst wird sie vollständig ausgeführt.

Im folgenden diskutieren wir den algorithmischen Aufwand für die Überwachung und Einhaltung der PENDANT-Option neben der Kontrolle der Relationalen Invarianten. Dabei wird immer nur der Erfolgsfall der Operation skizziert. Beim Sperren eines Objektes wird angenommen, daß seine Vorgängerobjekte auf dem hierarchischen Pfad die erforderlichen Anwartschaftssperren [GLPT76, Gr78] besitzen. Bei Indexstrukturen werden spezielle Sperrprotokolle [HR91] eingesetzt.

3.1 Operationen auf Tupeln in den Sohn-Relationen Si

Wir gehen vom allgemeinen Fall aus, daß n Sohn-Relationen über verschiedene Fremdschlüssel mit der Vater-Relation verknüpft sind. Dabei seien in der Vater-Relation m+1 Schlüsselkandidaten (UNIQUE inkl. Primärschlüssel) an den referentiellen Beziehungen, für die die PENDANT-Option zu kontrollieren ist, beteiligt (Bild 2). Der Sonderfall m=0 (Primärschlüssel) entspricht unserem anfänglichen Modellfall (Bild 1).
In den Relationen Si seien zur Vereinfachung der Diskussion keine UNIQUE-Attribute spezifiziert. Fremdschlüssel dürfen Null-Werte annehmen.

S1: Einfügen eines Tupels in Si

Die Einfügeoperation erfordert die Überprüfung der Primärschlüsselbedingung in Si und der referentiellen Beziehung zu V mit V1=v. Die PENDANT-Option führt hier auf keinen oder geringfügigen Zusatzaufwand.

Der interne Verlauf dieser Operation läßt sich wie folgt beschreiben:

1. Setze für das Si-Tupel eine lange X-Sperre und füge es in die Si-Relation ein.
2. Füge Primärschlüssel in den UNIQUE-Index I_{Si}(Si1) ein und setze eine lange X-Sperre für den Primärschlüsselwert. Dabei wird automatisch die Eindeutigkeit des Wertes überprüft.
3. Füge den Fremdschlüssel V1=v in I_{Si} (V1) (oder I_{Si} (SKj)) ein und setze dabei für den Schlüsselwert eine lange X-Sperre.
4. Überprüfe die Referentielle Integrität des eingefügten Fremdschlüssels durch Zugriff auf I_V(V1) (oder I_V(Skj)). Der zu überprüfende Schlüsselwert V1=v wird nur mit einer kurzen Sperre belegt:
   - Ist kein Vater vorhanden, so kann die Operation rückgängig gemacht werden oder
   - die Überprüfung zurückgestellt werden (DEFERRED), um ggf. ein entsprechendes Vater-Tupel einzufügen.

Bei der Überprüfung der Referentiellen Integrität auf der Vater-Seite ist eine kurze S-Sperre ausreichend für die Serialisierbarkeit der Transaktionen. Eine zweite Transaktion T2, die eine Aktualisierung auf dem betreffenden Vater-Tupel ausführt, könnte die Referentielle Integrität nicht gefährden, da dabei T2 seinerseits auf die Si-Relation zugreifen muß. Da dort jedoch alle Pfade zum neu eingefügten Tupel bis COMMIT gesperrt sind, müßte T2 solange die Sperre beachten [HR91].

In der weiteren Diskussion unterscheiden wir bei den Fremdschlüssel-Referenzen nicht mehr zwischen Primärschlüssel und Schlüsselkandidat und stellen den Index einheitlich mit I_{Si}(V1) dar.

S2: Löschen eines Tupels aus Si

Hinsichtlich der Überprüfung der Referentiellen Integrität ist diese Operation vollkommen unkritisch, da die Beziehungsrichtung Vater-Sohn nicht überprüft zu werden braucht. Bei Vorliegen der PENDANT-Option kann jedoch abhängig von der Implementierungstechnik erheblicher Aufwand hinzukommen.

1. Lösche Primärschlüssel-Referenz auf Si aus I_{Si}(Si1) und setze lange X-Sperre.
2. Lösche Fremdschlüssel-Referenz V1=v auf V aus I_{Si}(V1) und setze lange X-Sperre.
3. Setze lange X-Sperre auf das Si-Tupel und lösche es.
4. Überprüfe die PENDANT-Bedingung #PP≥1 (nach Löschen des Si-Tupels). Falls sie erfüllt ist, endet die Operation. Andernfalls gibt es mehrere Möglichkeiten:
   - V war nur mit Sohn-Relationen mit der PENDANT-Option verknüpft. In diesem Fall wird das zugehörige V-Tupel gelöscht; es gibt keine Folge-Operationen auf den Si-Relationen.
   - V war auch mit Sohn-Relationen verknüpft, die nicht die PENDANT-Option spezifiziert hatten. Abhängig von deren spezifizierten Referentiellen Aktionen sind Folgeaktionen auf diesen Sohn-Relationen erforderlich (z.B. ON DELETE CASCADE), die vom Löschen des Vater-Tupels ausgelöst werden [HR91].
   - Ist dabei für eine Sohn-Relation RESTRICT spezifiziert, so kann das Vater-Tupel nicht automatisch gelöscht werden. Die gesamte Operation - das Löschen des Si-Tupels - wird also rückgängig gemacht. Nach explizitem Löschen der RESTRICT-Tupel durch den Programmierer kann die Lö-schung des Si-Tupels wiederholt werden.

S3: Ändern eines Tupels in Si

Kosten für die Referentielle Integrität/PENDANT-Option fallen nur an, wenn an der Änderung Fremdschlüsselwerte beteiligt sind. Deshalb soll sich hier die Änderung auf den Fremdschlüsselwert beziehen. Die Kosten setzen sich dabei im wesentlichen aus Anteilen der beiden vorher skizzierten Operationen zusammen.

Es ist zu beachten, daß alle Pfade zum Tupel vor seiner Änderung gesperrt sein müssen:

1. Setze eine lange X-Sperre für den Primärschlüsselwert in I_{Si}(Si1).
2. Aktualisiere I_{Si}(V1) und sperre dabei den alten und neuen Wert mit einer langen X-Sperre.
3. Sperre das betroffene Si-Tupel mit einer langen X-Sperre und ändere die Attributwerte.
4. Überprüfe die PENDANT-Bedingung #PP≥1 für den gelöschten Fremdschlüsselwert. Falls sie nicht erfüllt ist, ist eine der beschriebenen Reaktionsmöglichkeiten anzuwenden.
5. Überprüfe die Referentielle Integrität des neuen Fremdschlüsselwertes durch Zugriff auf I_V(V1). Setze dabei auf dem hierarchischen Pfad zum gesuchten Schlüsselwert nur kurze S-Sperren.

3.2 Operationen auf Tupeln in der Vater-Relation V

Für jede Sohn-Relation können nach SQL2 Maßnahmen spezifiziert werden, die durch Aktualisierungsoperationen auf der Vater-Relation ausgelöst werden (z. B. ON DELETE). Diese Maßnahmen lassen sich als systemdefinierte Trigger auffassen, die etwaige Verletzungen der Referentiellen Integrität automatisch korrigieren und den DB-Zustand entsprechend nachführen. In unseren Kontext ist hierbei vorrangig die Wirkung der PENDANT-Option zu untersuchen.

V1: Einfügen eines Tupels in V

Diese Operation verlangt hinsichtlich der normalen Referentiellen Integrität keine Prüfungen. Die PENDANT-Option jedoch verkompliziert das Einfügen in die V-Relation, da für jedes V-Tupel mindestens ein PENDANT-Pfad vorliegen muß. Da andererseits eine PENDANT-Referenz nicht ohne Partner-Tupel in V bestehen kann (Referentielle Integrität), kann das Einfügen in V nur durch Zusammenfassung mehrerer Operationen und einer verzögerten Integritätsprüfung erfolgen, um das "Henne-Ei-Problem" zu umgehen.

1. Setze für das V-Tupel eine lange X-Sperre und füge es in die V-Relation ein.
2. Füge Primärschlüssel und Schlüsselkandidaten in die betreffenden I_V(V1) oder I_V(SKi) ein und überprüfe dabei ihre UNIQUE-Eigenschaft. Setze lange X-Sperren für die zugehörigen Schlüsselwerte in den Indexstrukturen.
3. Setze für ein Si-Tupel eine lange X-Sperre und füge es in die Si-Relation ein (oder ändere für ein existierendes Tupel den Fremdschlüsselwert, so daß eine gültige PENDANT-Referenz zum neuen V-Tupel entsteht).
4. Aktualisiere die zum Si-Tupel gehörenden Indexstrukturen und setze für die Schlüsselwerte lange X-Sperren.

Damit ist die PENDANT-Bedingung für das neu eingefügte V-Tupel sichergestellt.

V2: Löschen eines Tupels in V

Für diese Operation kann in der Fremdschlüssel-Klausel der Si-Relationen die Option

[ON DELETE {SET NULL | CASCADE | RESTRICT}]

angegeben werden. In Zusammenhang mit der PENDANT-Option ist im SQL3-Entwurf die Option SET DEFAULT nicht erlaubt. Neben den n Si-Relationen können weitere k Sohn-Relationen von V existieren, die aber nicht die PENDANT-Bedingung spezifiziert haben. Sie können als Referentielle Aktionen alle Löschoptionen spezifizieren. Damit kann also das Löschen eines V-Tupels mengenorientierte Folgeoperationen in bis zu k+n Sohn-Relationen auslösen. Der interne Operationsverlauf läßt sich wie folgt skizzieren:

1. Suche das V-Tupel über eine eine Indexstruktur I_V und lokalisiere es. Sperre die Pfade zum Tupel über alle zugehörigen I_V und das V-Tupel mit langen X-Sperren. Entferne die Tupelreferenzen und lösche das Tupel.

Für jede der k+n Sohn-Relationen sind die Schritte 2-4 auszuführen:

2. Bei Option RESTRICT: Zur erfolgreichen Durchführung dieser Operation müssen die zugehörigen Si-Tupeln vorher gelöscht werden. Die Integritätsprüfung ist dabei ggf. zu verzögern.
   Überprüfe weiterhin die Existenz von S-Tupeln (ohne PENDANT-Option) durch Zugriff auf die entsprechenden Sohn-Relationen; dabei sind nur kurze Sperren zu setzen. Wird ein S-Tupel gefunden, muß die gesamte Operation zurückgesetzt werden.

3. Bei Option SET NULL und SET DEFAULT: Alle Tupeln in Si und den weiteren Sohn-Relationen mit dem betreffenden Fremdschlüsselwert werden über die zugehörigen Indexstrukturen aufgesucht. Ihre Werte werden auf NULL- oder Default-Werte gesetzt. Dabei sind die Indexstrukturen anzupassen. Lange X-Sperren sind auf die geänderten Tupel und auf alle zugehörigen Indexstrukturen (Primärschlüssel- und Fremdschlüsselwerte) zu setzen.

4. Bei Option CASCADE: Die zugehörigen Tupeln in den Si-Relationen und in den weiteren Sohn-Relationen werden über den Fremdschlüssel-Index lokalisiert. Es müssen alle Pfade zu diesem Tupeln und die Tupel selbst mit langen X-Sperren belegt werden. Anschließend sind die Tupeln und ihre Referenzen aus den Indexstrukturen zu löschen.

Die Folgeoperationen auf den k Sohn-Relationen ohne PENDANT weisen lediglich bei der Operation RESTRICT Abhängigkeiten mit den Si-Relationen auf, die der PENDANT-Bedingung entstammen. Bei allen anderen Optionen sind diese Operationen unabhängig voneinander.

V3: Ändern eines Tupels in V

In der Fremdschlüsselklausel der Si-Relationen kann die Option

[ON UPDATE{SET NULL | CASCADE | RESTRICT}]

spezifiziert werden, was die Referentiellen Aktionen auf den Si-Relationen festlegt. Die weiteren Bedingungen entsprechen denen von “Löschen eines Tupels in V”.
Es wird hier angenommen, daß sich die Änderung auf den Primärschlüsselwert oder den Wert eines Schlüsselkandidaten bezieht, über den Fremdschlüsselreferenzen definiert sind. Die PENDANT-Referenzen können sich aus Referenzen auf Primärschlüssel oder Schlüsselkandidat ergeben.
Für den internen Verlauf ergibt sich:

1. Lokalisiere V-Tupel über eine Indexstruktur I_V und sperre alten und neuen Schlüsselwert sowie V-Tupel mit langen X-Sperren. Ändere Schlüsselwerte und V-Tupel.

Für jede der k+n Sohn-Relationen sind die Schritte 2-4 auszuführen:

2. Bei Option RESTRICT: Zur erfolgreichen Durchführung dieser Änderungsoperation müssen die zugehörigen Si-Tupeln vorher gelöscht werden. Wegen der PENDANT-Bedingung ist die Integritäts-prüfung dabei ggf. zu

verzögern. Für den neuen Wert ist ein neues Si-Tupel oder eine neue PENDANT-Referenz einzufügen. Für Tupel und Indexstrukturen sind lange X-Sperren zu setzen. Die Überprüfung der Referentiellen Integrität ist entsprechend zu verzögern. Falls weitere Sohn-Relationen existieren, ist zu überprüfen, ob Si-Tupeln (ohne PENDANT-Option) vorhanden sind: dabei sind nur kurze S-Sperren zu setzen. Die Existenz eines S-Tupels erzwingt das Rücksetzen der gesamten Operation.

3. Bei Option SET NULL oder SET DEFAULT: Der Ablauf der Operation in den Si-Relationen und in den weiteren Sohn-Relationen entspricht dem bei "Löschen eines Tupels in V", was einem Löschen der Beziehungen gleichkommt. Wegen der PENDANT-Option ist zu überprüfen, ob nach Löschen des alten Wertes im V-Tupel #PP≥1 erfüllt ist (über Werte anderer Schlüsselkandidaten oder über Fremdschlüssel mit anderen Referentiellen Aktionen, die auf demselben Schlüsselkandidaten hin definiert sind). Falls #PP=0 festgestellt wird, ist für den neuen Schlüsselwert ein neues Si-Tupel oder eine neue PENDANT-Referenz (in ein vorhandenes Tupel) einzufügen. Entsprechende lange X-Sperren sind in diesem Fall zu setzen.

4. Bei Option CASCADE: Diese Option erzwingt einen Ablauf der Änderungsoperationen in den Si-Relationen und den weiteren Sohn-Relationen ähnlich wie bei "Löschen eines Tupels in V, 3.", wobei der neue Schlüsselwert von V in den qualifizierten S-Relationen und deren Indexstrukturen einge-setzt wird. Auch wenn nur einer von m Schlüsselkandidaten geändert wird, bleibt die Anzahl der PENDANT-Pfade nach der Änderung gleich, da die PENDANT-Referenzen auf dem alten Wert auf den neuen Wert umgesetzt werden

3.3 Überprüfung der PENDANT-Bedingung

Die zentrale Frage, wie die PENDANT-Bedingung effizient zu überprüfen ist, wurde bisher ausgeklammert. Da ihre praktische Realisierbarkeit jedoch wesentlich von einem kostengünstigen Verfahren abhängt, kommt dieser Frage ein hoher Stellenwert zu. Im folgenden werden deshalb verschiedene Lösungsansätze genauer untersucht.

1. Dynamische Evaluierung

Mit unserem Zugriffspfadmodell ist etwa folgende dynamische Evaluierung möglich. Unter Ausnutzung der Indexstrukturen wird neben der zu löschenden PENDANT-Referenz noch mindestens eine gesucht, die die PENDANT-Bedingung #PP≥1 erfüllen würde. Wir nehmen an, daß alle PENDANT-Referenzen über die Schlüsselbedingung V1=v definiert sind und daß in Si ein Tupel mit V1=v gelöscht wird. Der Prüfalgorithmus hat dann im Prinzip folgende Schritte auszuführen. Dabei sind alle Schlüsselwerte in allen Indexstrukturen, die aufgesucht werden, mit langen S-Sperren zu versehen (slehe 4. Synchronisationsaspekte):

1. Suche lokal in I_{Si} (V1):

   Bleibt für V1=v noch mindestens eine Referenz im Index, ist #PP≥1,

   sonst:

2. Suche global in allen $I_{Sj}(V1)$:

   For j:=1 T0 n D0

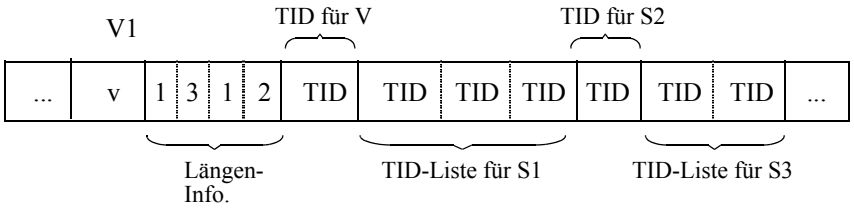
   IF (j<>i) AND #Ref (I_{Sj} (V1=v)) > 0 THEN EXIT; #PP≥1

3. sonst: #PP=0

Falls nicht alle PENDANT-Referenzen über denselben Schlüssel in der Vater-Relation definiert sind, ist nur für Schritt 1 die Suchbedingung (V1=v) sicher bekannt. Um in Schritt 2 die betreffenden Referenzen in den Indexstrukturen auffinden zu können, muß vorher das Vater-Tupel über den bekannten Referenzwert (V1=v) in $I_V(V1)$ lokalisiert und aufgesucht werden, um die anderen Schlüsselwerte (Skj=wj) zu bestimmen.
Ganz offensichtlich ist die dynamische Evaluierung besonders geeignet, wenn mehrere PENDANT-Referenzen pro Sohn-Relation existieren, d. h., wenn in den meisten Fällen die lokale Prüfung erfolgreich ist. Im worst case müssen jedoch n Indexstrukturen aufgesucht werden, bis entschieden werden kann, ob die PENDANT-Bedingung erfüllt ist oder nicht. Die Bedingung #PP=0, die ein Löschen des Vater-Tupels auslöst, kann immer erst nach Durchsuchen aller n Indexstrukturen festgestellt werden. Auch wenn mehrere PENDANT-Referenzen vorhanden sind, kann der Suchaufwand beträchtlich sein, da er von der Verteilung der Referenzen auf die Sohn-Relationen und von der Suchreihenfolge abhängt.

2. Zugeschnittene Zugriffspfadunterstützung

Unter gewissen Bedingungen läßt sich dieser Suchaufwand drastisch reduzieren. Sind alle Fremdschlüssel auf einen Schlüssel in der Vater-Relation hin definiert (ein gemeinsamer Wertebereich), so läßt sich zur Abbildung aller referentieller Beziehungen eine verallgemeinerte Zugriffspfadstruktur (VZPS) heranziehen [Hä78]. Sie basiert auf einem B*-Baum, wobei in den Blattknoten für einen Schlüsselwert alle variabel langen TID-Listen zusammengefaßt werden. Diese gehören zu Attributen verschiedener Relationen, die auf demselben Wertebereich definiert sind. In der folgenden Darstellung ist ein Eintrag für V1=v für die Relationen V und S1, S2 und S3 skizziert:



Mit Hilfe dieser Struktur fällt für die Kontrolle der PENDANT-Bedingung überhaupt kein Zusatzaufwand an, da sie bei der Löschung der Fremdschlüsselreferenz (V1=v) ohne weitere Seitenzugriffe mit überprüft werden kann.

Auch wenn s Schlüssel (s>1) der Vater-Relation referenziert werden, wäre die Nutzung vorteilhaft. Mit s solcher Zugriffspfade kann der anfallende Prüfaufwand im Mittel reduziert werden, solange s<n ist.

3. Referenzzähler-Methode

Eine Referenzzähler-Methode müßte die aktuelle #PP an einer Stelle verwalten, so daß der Suchaufwand möglichst gering bleibt (wie bei einem VZPS). Aktualisierungskosten würden dann bei jedem Einfügen, Löschen oder Ändern eines Tupels in einer Sohn-Relation fällig. Die zu prüfende Frage ist, ob sich durch eine solche zentrale Methode Kosten im Vergleich zur dynamischen Evaluierung einsparen lassen.
Im einfachsten Fall sind alle PENDANT-Referenzen über ein Attribut (genauer einen Wertebereich) definiert. Wir unterstellen hier das Primärschlüsselattribut V1. Da I_V(V1), wie ausgeführt, beim Einfügen und folglich auch beim Ändern (neuer Wert) eines Sohn-Tupels zur Überprüfung der Referentiellen Integrität aufgesucht werden muß, könnte man im Index für jeden Wert einen Referenzzähler einrichten, der während der Prüfoperation inkrementiert wird. Beim Löschen und beim Austragen des alten Wertes während des Änderns ist ein zusätzlicher Zugriff auf I_V(V1) erforderlich, der ausschließlich der Kontrolle der PENDANT-Bedingung anzulasten ist. Dabei wird der betreffende Referenzzähler dekrementiert. In diesem Fall enthält der Referenzzähler dann unmittelbar den aktuellen Wert für #PP.

Die Referenzzähler-Methode liefert zuviel Information, da immer nur nach dem Erfülltsein von #PP≥1 gefragt wird. Sie ist offensichtlich nur dann besonders vorteilhaft, wenn es mehrere Sohn-Relationen mit jeweils einer sehr geringen durchschnittlichen Anzahl von PENDANT-Referenzen gibt, so daß eine dynamische Evaluierung einen größeren Prüfaufwand verursachen würde.

Die Referenzzähler-Methode verliert jedoch ihre konzeptuelle Einfachheit, sobald PENDANT-Referenzen über mehrere Attribute (Wertebereiche) spezifiziert sind. In den Indexstrukturen der Vater-Relation würden pro Vater-Tupel mehrere Zähler auftreten, die die beabsichtigte Verwendung stark verkomplizieren würden. Deshalb müßte in solchen Fällen der Referenzzähler im jeweiligen Vater-Tupel selbst angelegt werden, was bei jeder Inkrement- und Dekrementoperation (INC/DEC) neben dem Indexzugriff ein Lokalisieren und Aktualisieren des Vater-Tupels impliziert. Diese Tatsache beeinträchtigt die Leistungsfähigkeit der Referenzzähler-Methode erheblich.

Bei der Nutzung einer VZPS stehen bei jedem Eintrag (V1=v) die Menge der Referenzzähler und Referenzen der beteiligten Relationen, so daß man das auch als Führen eines Referenzzählers ohne Zusatzaufwand interpretieren kann. Bei Einsatz von s VZPS müßte der Zählerwert aus den verteilten Teilbeiträgen ermittelt werden, was aber für die Überprüfung der PENDANT-Bedingung gar nicht erforderlich ist (siehe obiger Prüfalgorithmus).

4. Synchronisationsaspekte

Zur Verhütung von Phantomproblemen und von Abhängigkeiten von nicht freigegebenen Änderungen sind bei der Überprüfung der PENDANT-Bedingung alle Schlüsselwerte und Referenzen, auf die zugegriffen wird, bis Commit vor Änderungen paralleler Transaktionen zu isolieren.

Das Lesen des Referenzzählers verlangt bei normaler Implementierung das Setzen einer langen S-Sperre. Zusätzlich erfordert die Wartung bei der Referenzzähler-Methode für jede Aktualisierungsoperation in einer Sohn-Relation das Setzen von langen X-Sperren auf einem Schlüsselwert im Index $I_V(V1)$ oder gar im kostspieligeren Fall auf einem Vater-Tupel. Eine Spezialimplementierung dieser Sperre im Sinne der Escrow-Idee [ON86] könnte ihre Behinderungswirkung erträglich gestalten, da nur der Referenzzähler zu schützen ist. INC und DEC sind außerdem kommutativ, was prinzipiell parallele Aktualisierungen von nicht beendeten Transaktionen erlaubt. Jedoch ist hier stets zu kontrollieren, daß die PENDANT-Bedingung #PP≥1 nicht durch Mehrbenutzereffekte verletzt werden kann [ON86, Hä88]. In jedem Fall kostet die Wartung des Referenzzählers zusätzliche Schreib- und Log-Operationen.

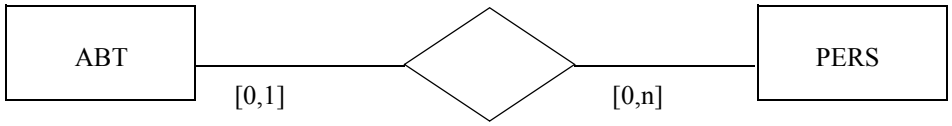
Der Prüfalgorithmus bei der dynamischen Evaluierung erfordert keine zusätzlichen Schreib- oder Log-Operationen. Jedoch sind potentielle Schlüsselwerte in n Indexstrukturen auf Sohn-Relationen (und ggf. das Vater-Tupel zusammen mit dem Schlüssel in $I_V(V1)$) zu sperren. Dafür genügen keine kurzen S-Sperren (siehe Abschnitt 3.1), da es sonst zu Race-Bedingungen zwischen Einfügungen und Löschungen von PENDANT-Referenzen kommen könnte, d.h., es sind lange S-Sperren zur Sicherung der Serialisierbarkeit anzuwenden. Die Anzahl dieser Sperren hängt offensichtlich stark von der Menge und der Verteilung der PENDANT-Referenzen ab. Genügt eine lokale Überprüfung (Schritt 1), so sind keine zusätzlichen Sperren erforderlich. Bei globaler Suche dagegen kann es zu langen Wartezeiten kommen, wenn ein betreffender Indexwert mit einer langen X-Sperre belegt ist (Einfügen oder Löschen einer PENDANT-Referenz).
Eine weitere Lösungsmöglichkeit bei der dynamischen Evaluierung bietet sich durch exklusives Sperren des Vaters (lange X-Sperre), was kurze S-Sperren auf den referenzierten Schlüsseln der Sohn-Relationen erlaubt (siehe Abschnitt 3.1). Jedoch könnte eine solche Lösung anfällig für Deadlocks sein.

Die beiden Methoden - dynamische Evaluierung und Referenzzähler-Methode - verlangen erheblichen Zusatzaufwand, der entweder bei der Überprüfung oder bei den Aktualisierungsoperationen aufzubringen ist. Sie haben beide ihre Stärken und Schwächen, die wesentlich von den Problemparametern abhängen: Anzahl der Sohn-Relationen in der PENDANT-Beziehung, Anzahl der verschiedenen Schlüsselwerte für die PENDANT-Pfade eines Vater-Tupels, Verteilung der PENDANT-Referenzen auf die Sohn-Relationen, Lokalität der Überprüfung. Deshalb ist auf dieser abstrakten Betrachtungsebene keine konkrete Empfehlung für die eine oder andere Methode angezeigt. In jedem Fall reduziert die Beschränkung auf einen Schlüsselwert für alle PENDANT-Pfade das Problem. Die zusätzliche Nutzung einer verallgemeinerten Zugriffspfadstruktur zur Darstellung aller PENDANT-Beziehungen bringt alle weiteren Schwierigkeiten zum Verschwinden.

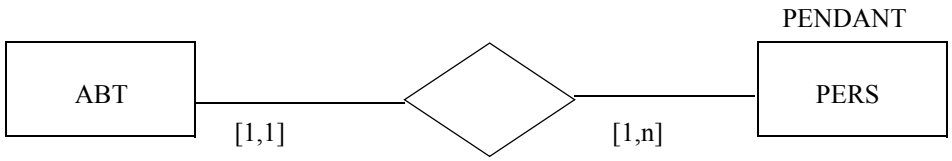
4. Modellierung mit der PENDANT-Option

Das Relationenmodell erlaubt die Spezifikation von benutzerdefinierten Beziehungen mit Hilfe der Relationalen Invarianten. Solche Beziehungen werden über die Gleichheit von Attributwerten ausgedrückt und durch das Konzept der Referentiellen Integrität überwacht. Die Beziehungen selbst sind vom Typ 1:n, wobei die Anzahl der Tu-

pel, die an einer bestimmten Beziehung teilnehmen, nicht genau festgelegt werden kann. Deshalb bedeutet die Beziehung ABTEILUNGSZUGEHÖRIGKEIT zwischen ABT und PERS gewöhnlich, daß ein Angestellter höchstens einer Abteilung angehört und eine Abteilung zwischen 0 und n Angestellte hat. Als ER-Diagramm läßt sich dies folgendermaßen ausdrücken:



Will man durch die Definition der Beziehung erreichen, daß jeder Angestellte einer Abteilung angehören muß ([1,1]), so muß man die Option NOT NULL für das Fremdschlüsselattribut in PERS heranziehen. Die Zuordnung der PERS-Tupel zu einem ABT-Tupel gewährleistet dann das Konzept der Referentiellen Integrität. Bei der Einschränkung der anderen Seite der Beziehung hilft das Konzept der Referentiellen Integrität nicht.
Mit Hilfe der PENDANT-Option kann gewährleistet werden, daß eine Abteilung immer mindestens einen Angestellten hat, wie im folgenden ER-Diagramm dargestellt.



Allerdings ist hier unterstellt, daß PERS die einzige Sohn-Relation von ABT mit der PENDANT-Option ist. Eine Einschränkung von n als maximale Anzahl der PERS-Tupel, die zu einer Abteilung gespeichert sind, gelingt auf diesem Wege nicht. Diese Wirkung der PENDANT-Option ließe sich auch mit der CHECK-Klausel nachbilden:

```
CREATE ASSERTION A1
        CHECK (ABT.ANR IN
                (SELECT PERS.ANR
                 FROM PERS))
```

Eine allgemeine Lösung für die Einschränkung von Beziehungstypen erreicht man mit Hilfe des Konzeptes der Kardinalitätsrestriktion, wodurch für jede Seite einer Beziehung eine Unter- und Obergrenze (at least, at most) für die Anzahl der Tupel, die an einer Beziehung dieses Typs teilnehmen, festgelegt wird. Beispielsweise könnte man so durch [1,1] und [5,10] festlegen, daß jeder Angestellte immer einer Abteilung zugeordnet sein muß und daß eine Abteilung immer zwischen 5 und 10 Angestellte haben muß. Es ist offensichtlich, daß zur Einhaltung solcher Beziehungen ein Transaktionskonzept erforderlich ist und die Überprüfung von Aktualisierungen im Modus 'DEFERRED' erfolgen muß. Eine Simulation von Kardinalitätsrestriktionen durch CHECK-Bedingungen scheint möglich (siehe oben).

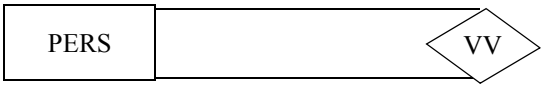
Als Nebenbemerkung: Soll eine dauerhafte Beziehung spezifiziert werden (definierter Fremdschlüsselwert darf sich nicht ändern), so war das im Relationenmodell überhaupt nicht möglich. Was die Spezifikation solcher struktureller Beziehungen angeht, so bietet beispielsweise schon das Netzwerkmodell durch die Set-Mitgliedschaft aus-

gefeiltere Beschreibungsmöglichkeiten (AUTOMATIC, MANUAL, FIXED [CODA78]). In SQL3 könnten solche Bedingungen jedoch durch geeignete Formulierungen mit Hilfe von Triggern und Assertions nachgebildet werden.

Das obige Beispiel hat gezeigt, daß die PENDANT-Option zur Modellierung ohnehin nur dann gezielt eingesetzt werden kann, wenn sich PENDANT auf eine Sohn-Relation beschränkt. In solchen Fällen kann man bei einer Beziehung etwas genauer modellieren. Dieser Zugewinn an Modellierungsmächtigkeit soll nun an weiteren Beispielen diskutiert werden.

Hierarchie mit PENDANT-Option

Die Vorgesetzten-Hierarchie



kann man mit MNR als Fremdschlüssel und PNR als Primärschlüssel als Relation PERS1 darstellen:

PERS1 (PNR, MNR,...)

1	-
2	1
3	1
4	2
5	2

Will man durch die PENDANT-Option erreichen, daß es keine isolierten Untergebenen/Vorgesetzten gibt, so hilft sie in der obigen Lösung nicht weiter. Es ergibt sich #PP=0 für die Angestellten, die keine Untergebene haben (Tupel mit PNR 3,4,5). Wenn diese gelöscht werden, dann muß auch PNR=2 und schließlich PNR=1 gelöscht werden.

Durch separate Darstellung der VV-Beziehung gelangt man über einen Umweg ans Ziel:

PERS2 (PNR,...)

1
2
3
4
5

VV (UPNR, MNR)

2	1
3	1
4	2
5	2

UPNR und MNR seien als Fremdschlüssel in bezug auf PNR definiert und beide referentiellen Beziehungen seien mit der PENDANT-Option belegt. Die Anzahl der PENDANT-Pfade ist jetzt für alle PERS-Tupel mindestens 1, so daß diese Darstellung akzeptiert wird. Andererseits kann kein PERS2-Tupel ohne VV-Beziehung existieren, weil dann für dieses Tupel #PP=0 wird.

Um die angestrebte Lösung (Hierarchiebeziehung in einer Relation) zu erhalten, müssen also die PENDANT-Pfade beibehalten werden. Dies kann man aber auch mit einer Relation bewerkstelligen, wenn man die beiden Relationen zu einer zusammenmischt:

PERS3 (PNR, UPNR, MNR

1	-	-

2	2	1
3	3	1
4	4	2
5	5	2

Hier tritt UPNR als Rollenname für die VV-Beziehung auf und ist offensichtlich redundant. Bei einer Vereinigung mit PNR müßte die Bedeutung undefinierter Werte geregelt werden. Weiterhin müßte PNR dann Primär- und Fremdschlüssel zugleich sein, um die PENDANT-Pfade aufrechtzuerhalten.

Stückliste mit PENDANT-Option

Die Stückliste als Beispiel für eine n:m-Beziehung auf einem Objekttyp (TEIL) wird im Relationenmodell gewöhnlich folgendermaßen definiert:

TEIL (TNR,...)
1
2
3
4

STRUKTUR (OTNR, UTNR,...)	
1	3
1	4
2	3
2	4

Soll durch Modellierung erreicht werden, daß keine Einzelteile für sich existieren können, so läßt sich das mit der PENDANT-Option bewerkstelligen. Dazu ist die Spezifikation der PENDANT-Option für beide Fremdschlüsselbeziehungen erforderlich. Dadurch können alle TEIL-Tupel, die in einer STRUKTUR-Beziehung sind, dargestellt werden. Sobald eine Beziehung gelöscht wird und eines der beteiligten TEIL-Tupel in keiner weiteren Beziehung auftritt, wird es automatisch gelöscht (#PP=0). Als Folge davon sind bei dieser Modellierung keine Teile isoliert darstellbar. Einige Beispiele dafür sind im Bild 3 skizziert.

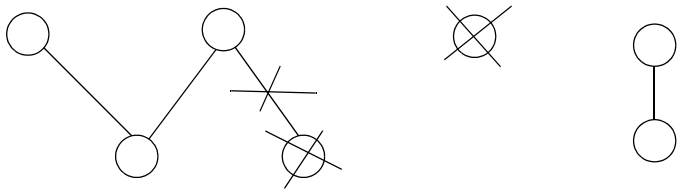


Bild 3: Ergebnis der PENDANT-Modellierung bei der Stückliste

Die Einschränkung der PENDANT-Option auf eine der beiden referentiellen Beziehungen ist wenig nützlich. Zusammenhängende Aggregate und Einzelteile (wie bisher gezeigt) könnten nicht dargestellt werden, da entweder Oberteile (PENDANT auf UTNR) oder Einzelteile (PENDANT auf OINR) ohne PENDANT-Pfade wären. Erst die Einführung von Zyklen macht gewisse Aggregate lebensfähig, wie folgendes Beispiel zeigt:

TEIL (TNR,...
1
2
3

STRUKTUR (OTNR, UTNR (PENDANT),...	
1	2
1	3
3	1

Erst die Einfügung des STRUKTUR-Tupels (3,1) gewährleistet die Darstellung des Aggregats. Da bei dieser Modellierung immer Zyklen beteiligt sein müssen, ist ihr allgemeiner Nutzen fragwürdig.

Generell haben die diskutierten Beispiele gezeigt, daß wohl in der Einführung der PENDANT-Option keine ausgeprägte Absicht der Verbesserung der Modellierungsmächtigkeit von SQL steckt. Es ist zum einen schwer, das Modellierungskonzept von PENDANT zu identifizieren, und zum anderen, es zielgerichtet einzusetzen. Das liegt vor allem daran, daß sich seine Wirkung auf n beteiligte Relationen bezieht, was außerdem noch einen hohen Prüfaufwand generiert.

Nur in Sonderfällen dürfte es deshalb eine Erleichterung und Verbesserung der Modellierung bringen. Das ist sicherlich nur dann der Fall, wenn die PENDANT-Option auf eine Sohn-Relation beschränkt bleibt. Bei dieser Einschränkung ist die Semantik "berechenbar" (spezielle Kardinalitätsrestriktion für eine Seite der Beziehung) und dem Programmierer verständlich. Für die Modellierung der allgemeinen Kardinalitätsrestriktionen von Beziehungen bringt sie jedenfalls nur wenig Hilfe. Wie die Beispiele gezeigt haben, scheint hierbei die CHECK-Klausel für die Modellierung ausreichend zu sein.

5. Zusammenfassung

Eine PENDANT-Beziehung kann über einer Vater-Relation und n Sohn-Relationen definiert sein. Zugleich kann diese Vater-Relation mit k weiteren Sohn-Relationen in jeweils einer Beziehung stehen, für die nur die Referentielle Integrität kontrolliert wird. Da die Fremdschlüsselverweise sich auch auf Schlüsselkandidaten beziehen können, ist es möglich, daß die zu einer PENDANT-Beziehung gehörenden PENDANT-Referenzen durch Werte von bis zu m+1 Attributen (Wertebereichen) gebildet werden. Diese Komplexität führt zu einer unklaren Semantik der PENDANT-Beziehung, erschwert ihr Verständnis und trägt zu ihren hohen Wartungskosten bei ($n>1$, $m>0$).
Im Vergleich zur Wartung der Referentiellen Integrität erfordert die PENDANT-Beziehung zusätzliche Kosten, die sich durch die Symmetrie der Beziehung ergeben. Der wesentliche Zusatzaufwand ist dabei die Feststellung der PENDANT-Bedingung ($\#PP \geq 1$), für die dynamische Evaluierung, Referenzzähler-Methode und Einsatz einer Verallgemeinerten Zugriffspfadstruktur diskutiert wurden. Dabei kann die Verallgemeinerte Zugriffspfadstruktur ihre großen Stärken zeigen, daß sie alle Referenzen von Attributen, die zu einem Wertebereich gehören, zusammenhängend speichert.
Generell läßt sich sagen, daß die Überprüfung der PENDANT-Bedingung vereinfacht wird, wenn keine Schlüsselkandidaten ($m=0$) oder nur ein Wertebereich an der Bildung der PENDANT-Beziehung beteiligt sind. Gibt es nur eine Sohn-Relation ($n=1$) in der PENDANT-Beziehung, so wird eine Überprüfung sehr einfach.
Der Fall $n=1$ erlaubt auch keine klarere Festlegung der Semantik der PENDANT-Option. In diesem Fall kann sie gezielt zur Modellierung einer symmetrischen Beziehung herangezogen werden, wobei sie jedoch nicht die Modellierungsmächtigkeit des Konzeptes der Kardinalitätsrestriktion erreicht. Insgesamt gesehen bleibt der Zugewinn an Modellierungsmächtigkeit, der durch die PENDANT-Option für referentielle Beziehungen im Relationenmodell eingeführt wird, recht bescheiden.

Danksagung:

Die Kommentare von E. Rahm, S. Meybrink und J. Reinert haben dazu beigetragen, eine frühere Version dieses Aufsatzes zu verbessern.

6. Literaturverzeichnis

CODA78 CODASYL Data Description Language, Journal of Development, Ottawa, 1978.

Da81 Date, C.J.: Referential integrity, in: Proc. 7th Int. Conf. on VLDB, 1981, pp. 2-12.

GLPT76 Gray, J.N., et al.: Granularity of Locks and Degrees of Consistency in a Large Shared Data Base, in: Modelling in Data Base Management Systems, North-Holland, 1976, pp. 365-394.

Gr78 Gray, J.N.: Notes on Data Base Operating Systems, in: Lecture Notes Computer Science, 60, Operating systems: An advanced course, Springer-Verlag, 1978, pp. 393-481.

Hä78 Härder, T.: Implementing a Generalized Access Path Structure for a Relational Data Base System, in: ACM Transactions on Database Systems, Vol. 3, No. 3, 1978, pp. 285-298.

Hä88 Härder, T.: Handling Hot Spot Data in DB-Sharing Systems, IBM Research Report RJ 5223, Almaden Research Center, San Jose, Calif., March 1987, in: Information Systems, Vol. 13, No. 2, 1988, pp. 155-166.

HR91 Härder, T., Rahm, E.: Zugriffspfad-Unterstützung zur Sicherung der Relationalen Invarianten, ZRI-Bericht 2/92, Universität Kaiserslautern, Dez. 1991.

HR92 Härder, T., Reinert, J.: Einsatz von Grid Files zur Sicherung der Referentiellen Integrität in SQL2, ZRI-Bericht 4/92, FB Informatik, Universität Kaiserslautern, Feb. 1992.

ON86 O'Neil, P.: The Escrow Transactional Method, in: ACM Transactions on Database Systems, Vol. 11, No. 4, 1986, pp. 405-430.

Re92 Reinert, J.: Sicherung der Referentiellen Integrität - Optimierung durch Schemaanalyse, Interner Bericht, Universität Kaiserslautern, Jan. 1992.

Sh90 Shaw, P.: Database Language Standards: Past, Present, Future, in: Database Systems of the 90s, A. Blaser (ed.), LNCS 466, Springer-Verlag, 1990, pp. 50-88.

SQL2 ISO/IEC JTC1/SC21 Information Retrieval, Transfer and Management for OSI: International Standard IOS/IEC 9075:1992, Revised Text of CD 9075.2, Information Technology - Database Languages - SQL2, for DIS registration and letter ballot, 10.04.1991, 522 pp.

SQL3 X3H2-91-183 DBL-KAW-003 ISO/IEC JTC1/SC21/WG3 N1223: ISO/ANSI Working Draft - Database Language SQL3, 07.1991, 772 pp.