Einsatz von Grid Files zur Sicherung der Referentiellen Integrität in SQL2

Theo Härder, Joachim Reinert

Universität Kaiserslautern

FB Informatik, Postfach 3049

6750 Kaiserslautern

Tel.: 0631/205-4030

e-mail: {haerder | jreinert} @informatik.uni-kl.de

ÜBERBLICK

Das Relationenmodell garantiert seinen Benutzern Integritätszusicherungen für Entitätsintegrität und Referentielle Integrität. Datenbanksysteme, die dem SQL2-Standard genügen wollen, müssen nicht nur die Einhaltung dieser fundamentalen Integritätsregeln, der sogenannten Relationalen Invarianten, sicherstellen, sie müssen auch eine explizite Systemkontrolle für erweiterte referentielle Beziehungen verfügbar machen. Solche Beziehungen ergeben sich durch Interpretation unvollständig definierter, zusammengesetzter Fremdschlüsselwerte; sie können durch die MATCH-Klausel mit den Optionen MATCH FULL und MATCH PARTIAL definiert werden. In diesem Aufsatz werden für zusammengesetzte Schlüssel Verfahren untersucht, um diese Integritätsprüfungen effizient abwickeln zu können. Dabei werden speziell die Möglichkeiten des Einsatzes von Grid Files genauer betrachtet. Eine Analyse des für die Integritätsüberprüfungen anfallenden Mehraufwands beschließt unsere Untersuchungen. Dabei zeigt sich, daß bei MATCH PARTIAL und der Nutzung von Mehrattributschlüsseln mit k einfachen Attributen mit einem Zusatzaufwand von $O(2^k)$, was die Anzahl von Bereichsfragen angeht, gerechnet werden muß.

1. Einführung

In den SQL2-Standardisierungsvorschlag [SQL2, Sh90] wurden eine Reihe von Aussagen aufgenommen, die die Definition und Einhaltung der Relationalen Invarianten in Relationalen Datenbanksystemen betreffen. Danach hat das Datenbanksystem automatisch für die Eindeutigkeit und Definiertheit der Primärschlüsselwerte (Entitätsintegrität) sowie für die Einhaltung der zwischen Primärschlüssel (Schlüsselkanditat) und Fremdschlüssel definierte Referentiellen Integrität zu sorgen [Da81]. Durch die UNIQUE-Option lassen sich ggf. mehrere Schlüsselkandidaten in einer Relation spezifizieren, für die das Datenbanksystem Eindeutigkeit der Werte zu gewährleisten hat. In diesem Zusammenhang muß auch die MATCH-Klausel genannt werden. Sie erlaubt die Definition und Überwachung von "erweiterten Beziehungen" zwischen Relationen über Schlüsselkandidaten und Fremdschlüssel, die sich durch partielle Undefiniertheit von Werten ergeben können.

Schlüsselkandidaten können im Relationalen Modell aus einem Attribut oder einer minimalen Gruppe von Attributen bestehen. Entsprechend sind die zugehörigen Fremdschlüssel aufgebaut und auf denselben Wertebereichen definiert. Für einfache Attribute wurden die Operationen zur Erhaltung bzw. Sicherung der Relationalen Invarianten im Detail in [HR91] betrachtet. Dabei wurde festgestellt, daß für alle Attribute, für die eine Schlüsseleigenschaft zu überprüfen oder zu wahren ist, spezielle Zugriffspfade zu spezifizieren sind, um zu akzeptablen Laufzeiten bei Modifikationsoperationen zu kommen.Ein Schlüsselkandidat ist dabei durch einen UNIQUE- und ein Fremdschlüssel durch einen NONUNIQUE-Index zu unterstützen. Eine genaue Kostenanalyse belegte die generelle Einschätzung der Praxistauglichkeit bestimmter Implementierungsformen von Zugriffspfaden: bei Einattributschlüsseln kommen nur Indexstrukturen auf der Basis von B*-Bäumen [Co79] oder von, als eine auf die Prüfaufgaben besonders zugeschnittene Variante, verallgemeinerten Zugriffspfadstrukturen [Hä78] in Frage.

Bei Mehrattributschlüsseln, mit denen für die zu speichernden Tupel Entitätsintegrität und Referentielle Integrität ausgedrückt werden, ist die Entscheidung für eine geeignete Zugriffspfad-Unterstützung nicht so eindeutig. Benötigt wird ein leistungsfähiger Mehrattributzugriff über die k Attribute eines zusammengesetzten Schlüssels auf die zugehörigen Tupeln einer Relation. Bei der Sicherung der Relationalen Invarianten ist die effiziente Abwicklung von Punktfragen ausreichend. Diese lassen sich mit Hilfe von konkatenierten Attributen als Schlüssel auf B*-Bäumen relativ einfach simulieren. Kommt jedoch die Kontrolle erweiterter Beziehungen (MATCH {FULL | PARTIAL}) hinzu, so müssen allgemeinere Suchbedingungen für die k Schlüsselattribute spezifiziert werden können. Dabei sind oft mehrdimensionale Schlüsselbereiche zu durchsuchen, um die betreffenden Tupel schnell lokalisieren zu können.

Als "echte" k-dimensionale Zugriffspfadstruktur wurde in [Hä92] das Grid File [NHS84] für den Datenbankeinsatz analysiert. Als Funktionen wurden dabei Speicherung, Verwaltung und räumlicher Zugriff auf "punktförmige Objekte" gefordert; die Speicherung und das Aufsuchen räumlicher Objekte (geometrische Objekte) dagegen wurden dagegen ausgeschlossen. Beim Grid File wurden die Abbildungseigenschaften seiner Speicherungsstruktur untersucht und Synchronisationsprotokolle für den Mehrbenutzerbetrieb entwickelt. Weitere Strukturen, die in der Literatur für den mehrdimensionalen Zugriff vorgeschlagen wurden [Be75, FB74, Ro81], konnten sich bei den durch

den Datenbankeinsatz vorgegebenen Auswahlkriterien nicht qualifizieren, so daß das Grid File einziger Kandidat blieb.

In diesem Aufsatz untersuchen wir Einsatzmöglichkeiten und -kosten eines Grid Files bei allen Fragen der Integritätssicherung von Mehrattributschlüsseln. Dazu beschreiben wir die bei der Überprüfung der Relationalen Invarianten und der erweiterten Beziehungen anfallenden Operationen, insbesondere auch die vom Transaktionsparadigma geforderten Synchronisationsmaßnahmen. Anschließend führen wir die wesentlichen Konzepte des Grid Files ein und versuchen anhand einfacher Kostenmodelle den Aufwand für die verschiedenen Aufgaben der Integritätsprüfung abzuschätzen. Diese Bewertung soll die praktische Tauglichkeit des Grid Files bei den komplexen Maßnahmen der Integritätssicherung in Relationalen Datenbanksystemen klären helfen.

2. Referentielle Integrität bei Mehrattributschlüsseln

2.1 Modellannahmen

In diesem Aufsatz nehmen wir an, daß Primärschlüssel und Fremdschlüssel durch k Attribute definiert sind. Formal läßt sich unser Beispiel folgendermaßen beschreiben:

```
CREATE   DOMAIN    A1_typ AS ...
. . .
CREATE   DOMAIN    Bm_typ AS ...
CREATE   TABLE V (
     A1     A1_typ,
     A2     A2_typ,
     ....
     Ak     Ak_typ,
     ...
     An     An_typ,
PRIMARY KEY (A1, A2, ..., Ak))

CREATE   TABLE S (
     B1     B1_typ     PRIMARY KEY,
     B2     B2_typ,
     ...
     A1     A1_typ,
     ...
     Ak     Ak_typ,
     ...
     Bm     Bm_typ,
 CONSTRAINT SFK FOREIGN KEY (A1, ..., Ak)
 REFERENCES V (A1, ..., Ak)
```

[MATCH {FULL | PARTIAL}]
[ON UPDATE {SET NULL | CASCADE | SET DEFAULT}]
[ON DELETE {SET NULL | CASCADE | SET DEFAULT}].

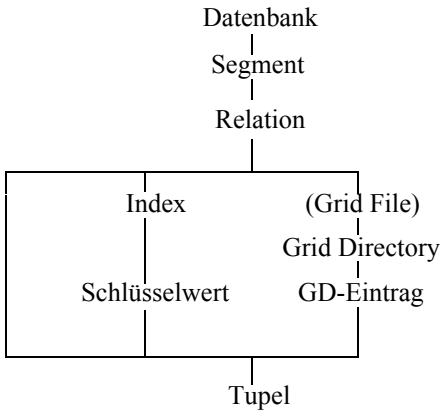
Die Referentielle Integrität zwischen den Relationen V (Vater) und S (Sohn) ist also über die Attribute A1, ..., Ak in beiden Relationen zu wahren. Dazu sei auf diesen k Attributen für jede der Relationen ein Grid File GF_V (A1, ..., Ak) bzw. GF_S (A1, ..., Ak) angelegt. Daneben sei eine Indexstruktur I_S (B1) auf dem Primärschlüssel von S definiert. Da im allgemeinen Fall mehrere Zugriffspfade auf eine Relation verweisen, werden die Tupeln einer Relation in einer separaten Speicherungsstruktur gehalten. Die Tupeln sind also nicht in eine Zugriffspfadstruktur eingebettet; jede Zugriffspfadstruktur enthält vielmehr Schlüsselwerte und TIDs als Verweise auf die Tupeln der Relation.

Die Relationalen Invarianten bezüglich der Relationen V und S können jeweils durch Einfügen, Ändern und Löschen eines Tupels in den betreffenden Relationen verletzt werden. Ihre Überprüfung soll zusammen mit der Durchführung der auslösenden Operation (IMMEDIATE) vorgenommen werden. Im Falle einer Integritätsverletzung ändert sich der DB-Zustand nicht, und es wird eine Fehlermeldung zurückgeliefert; sonst wird die Operation vollständig ausgeführt (statement atomicity).

2.2 Einhaltung des Transaktionsparadigmas

Alle DB-Operationen werden transaktionsgekapselt durchgeführt, d. h., die diese auslösenden Transaktionen verkörpern die ACID-Eigenschaften [HR83]. In unserem Zusammenhang ist dabei die Synchronisation von besonderer Wichtigkeit, da Integritätsprüfungen das Verkehrsaufkommen auf Zugriffspfaden und damit das Konflikt- und Blockierungspotential des Mehrbenutzerbetriebs verschärfen.

Das Transaktionskonzept fordert die Serialisierbarkeit aller am DB-Betrieb beteiligten Transaktionen, was die Isolation aller Änderungen bis Commit und die Wiederholbarkeit aller Lesevorgänge innerhalb einer Transaktion (Konsistenzebene 3) bedeutet. Alle effizienten Synchronisationsverfahren beruhen auf hierarchischen Sperrprotokollen, die sich unter den zahlreichen Literaturvorschlägen als einzige in der Praxis bewährt haben [EGLT76, Gr78]. Die Sperrhierarchie, die wir für unsere Überlegungen unterstellen, ist durch folgende Halbordnung gegeben:



Diese Darstellung beschreibt eine Typhierarchie; sie besagt, daß in unserem Modell ein Tupel zu einer Relation (in einem Segment) gehört. Zu dieser Relation können mehrere Indexstrkturen und Grid Files spezifiziert sein, so daß das Tupel über mehrere Zugriffspfade erreichbar ist. Ein Tupel der Relation S ist beispielsweise über einen Relationen-Scan, über einen Indexzugriff (I_S (B1)) oder über ein Grid File (GF_S (A1, ..., Ak)) zugreifbar. Da Grid File und Grid Directory stets in einer 1:1-Beziehung stehen, kann eines von beiden im Pfad weggelassen werden. Hierarchische Sperrprotokolle gehen nun gemäß der gezeigten Halbordnung vor und setzen auf den entsprechenden Knoten Sperren, deren Modi durch die folgende Verträglichkeitsmatrix definiert sind [GLPT76]

	IS	IX	S	SIX	X
IS	+	+	+	+	-
IX	+	+	-	-	-
S	+	-	+	-	-
SIX	+	-	-	-	-
X	-	-	-	-	-

Soll ein Tupel aktualisiert werden, so sind prinzipiell alle Pfade zum Tupel vorher zu sperren.; soll es nur gelesen werden, so genügt das Sperren eines hierarchischen Pfades. Die Sperren längs der hierarchischen Pfade müssen zur Sicherstellung der Serialisierbarkeit stets bis Commit der anfordernden Transaktion gehalten werden. Diese Forderungen implizieren nicht transaktionsgekapselte Strukturmodifikationen auf Zugriffspfaden oder gewisse Prüfoperationen im Rahmen der Sicherung der Relationalen Invarianten. Für solche Operationen sind Kurzzeitsperren ausreichend, so daß diese das Behinderungspotential auf diesen Strukturen, die oft eine hohe Verkehrsdichte aufweisen, nicht erhöhen.

2.3 Übersicht über die Definitionsmöglichkeiten für referentielle Beziehungen

Im neuen Standard zu SQL ist es möglich, verschiedene Arten der von Beziehungen durch referentielle Integrität zu erfassen. Die Unterschiede werden durch den Gebrauch von speziellen Optionen bei der Definition dieser Integritätsbedingungen realisiert. In SQL2 [SQL2] ist die Wahlmöglichkeit auf die MATCH-Klausel beschränkt. Diese beschreibt im wesentlichen die Interpretation von Null-Werten bei der Auswertung der referentiellen Integrität. In SQL3 [SQL3] sollen weitere Optionen möglich sein. Dabei scheint sicher, daß eine PENDANT-Option eingeführt wird, die eine Art symmetrische Beziehung definiert [Re92]. Auf die Betrachtung solcher symmetrischer Beziehungen und deren Erweiterungen zu Kardinalitätsrestriktionen wird im folgenden verzichtet. Es soll hier nur an einem einfachen Beispiel aufgezeigt werden, wie sich die Optionen, die von SQL2 zur Verfügung gestellt werden, auf die Operationen auswirken, die die Referentielle Integrität berühren.

In unserem Modellschema (Kap. 2.1) sind verschiedene Klauseln aufgenommen, die die Art und die Durchführung der Überprüfung der Referentiellen Integrität beschreiben. Abhängig von den Modifikationsoperationen auf den beteiligten Relationen sind unterschiedliche referentielle Aktionen wie SET NULL/SET DEFAULT, CASCADE

und RESTRICTED zu beachten. RESTRICTED ist in SQL2 nicht (aber in SQL3 wieder) definierbar; es entspricht hier dem Fall, daß keine referentielle Aktion spezifiziert ist.

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

Die Option MATCH FULL erlaubt nur vollständig definierte und vollständig undefinierte Werte im Fremdschlüssel; sie verbietet jedoch teilweise definierte zusammengesetzte Fremdschlüssel. Durch diese Option läßt sich erreichen, daß auch zusammengesetzte Fremdschlüssel als Einheit interpretiert werden.

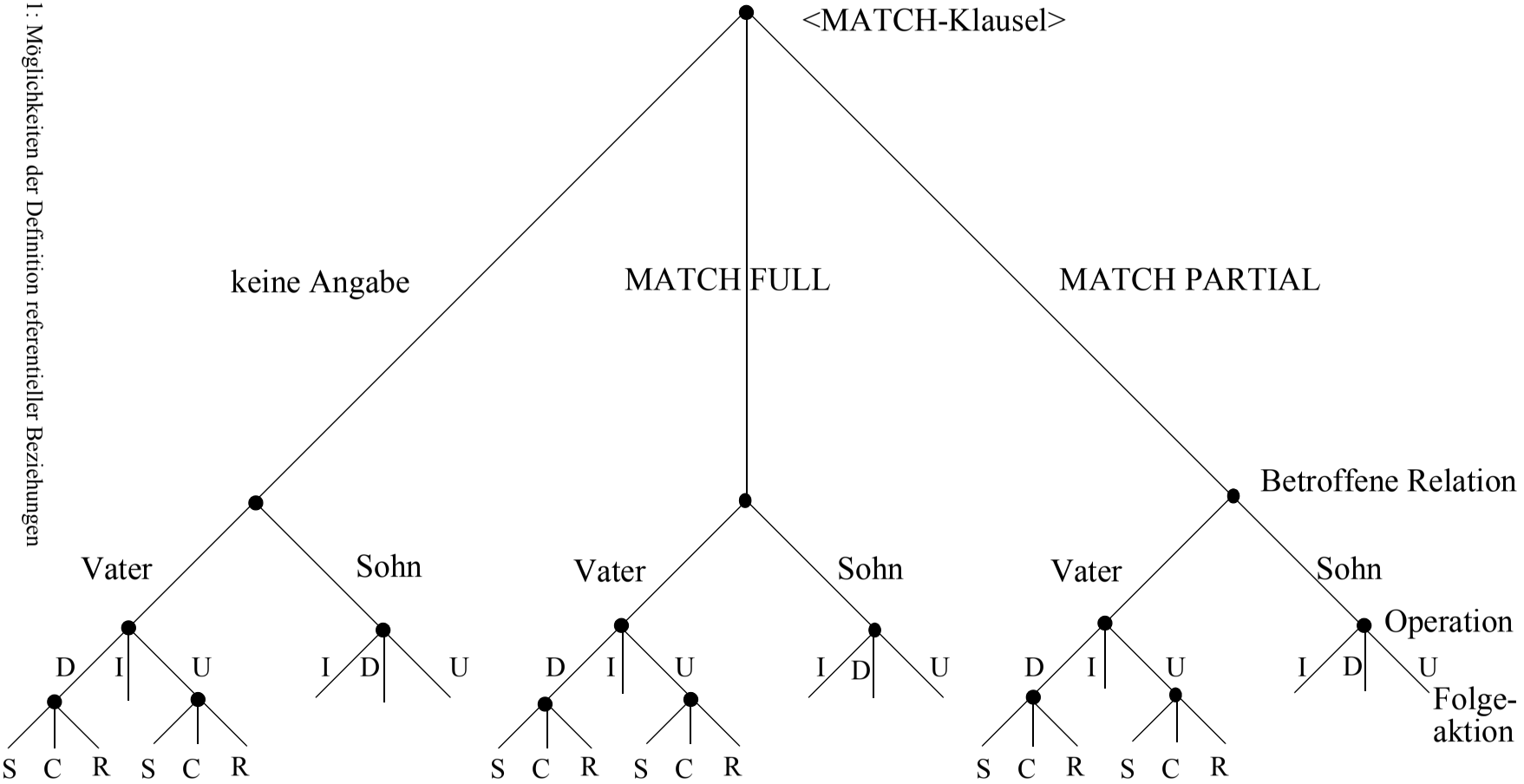
Im Gegensatz dazu läßt die Option MATCH PARTIAL teilweise definierte Fremdschlüssel zu und interpretiert diese auch noch. Auf diese Weise können partiell undefinierte Fremdschlüssel zur Darstellung von Beziehungen genutzt werden. Als Folge davon lassen sich die herkömmlichen 1:n-Beziehungstypen, die durch die Referentielle Integrität kontrolliert werden, zu speziellen n:m-Beziehungstypen erweitern. Diese auf die Verwendung von NULL-Werten aufgebauten Beziehungen werden ebenfalls systemkontrolliert verwaltet.

Eine Gliederung aller Möglichkeiten, referentielle Beziehungen zu definieren und zu überwachen, ist in Bild 1 veranschaulicht.

3. Wartung der Referentiellen Integrität

Bei der “klassischen” Referentiellen Integrität werden nur vollständig definierte Fremdschlüssel als Referenzwerte interpretiert. Dies wird erreicht entweder durch das Weglassen der MATCH-Klausel oder durch die Angabe MATCH FULL. Im folgenden werden alle Änderungsoperationen auf den an einer referentiellen Beziehung beteiligten Relationen betrachtet, wobei die anfallenden Kosten nur qualitativ beschrieben werden.Dabei wird immer nur der Ablauf der Operation im Erfolgsfall skizziert. Beim Sperren eines Objekts wird angenommen, daß seine Vorgängerobjekte auf dem hierarchischen Pfad die erforderlichen Anwartschaftssperren besitzen. Ferner werden auf einem Index oder einem Grid File spezielle Sperrprotokolle eingehalten [HR91, Hä92].

Bild 1: Möglichkeiten der Definition referentieller Beziehungen



S = Als Folgeoperation ist Delete/Update SET NULL oder SET DEFAULT definiert

C = Als Folgeoperation ist CASCADE definiert

R = Als Folgeoperation ist nichts definiert, entspricht etwa RESTRICTED in andernen Vorschlägen

D = Delete-Operation auf der Relation

U = Update-Operation auf der Relation

I = Insert-Operation auf der Relation

3.1 Operationen auf Tupeln in der Sohn-Relation S

S1: Einfügen eines Tupels in S

Es ist zu überprüfen, ob S.B1 einen eindeutigen Wert besitzt und ob das durch S.(A1,...,Ak) referenzierte Tupel in V existiert. In einer Standard-SQL- ähnlichen Anweisung kann man dies so formulieren:

```
IF    S.A1 = NULL OR S.A2 = NULL OR ... OR S.Ak = NULL OR
      EXISTS (SELECT * FROM V WHERE S.A1 = V.A1 AND ... AND S.Ak = V.Ak)
THEN INSERT INTO S ...
```

Der interne Ablauf dieser Operation läßt sich folgendermaßen beschreiben:

1. Setze für das S-Tupel eine lange X-Sperre und füge es in die S-Relation ein.
2. Füge Primärschlüssel S.B1 in den UNIQUE-Index I_S(B1) ein, wodurch automatisch die Primärschlüssel-Eigenschaft überprüft wird. Setze dabei eine lange X-Sperre für den Schlüsselwert von S.B1.
3. Füge den Fremdschlüssel S.(A1, ...,Ak) in den GF_S(A1, ...,Ak) ein. Setze dabei eine lange X-Sperre für Schlüsselwert von S.(A1, ...,Ak).
4. Überprüfe Existenz des Fremdschlüsselwerts S.(A1, ...,Ak), falls vollständig definiert, in V durch Zugriff auf GF_V(A1, ...,Ak) mit einer Punktfrage. Setze dabei auf den hierarchischen Pfad zum gesuchten Schlüsselwert nur kurze S-Sperren.

Bei der Überprüfung der Referentiellen Integrität auf GF_V(A1, ...,Ak) ist eine kurze S-Sperre ausreichend. Die Referentielle Integrität könnte nur durch eine zweite Transaktion T2, die gerade den betreffenden Schlüsselwert in GF_V(A1, ...,Ak) löscht, gefährdet werden. T2 muß nun aber ihrerseits die Referentielle Integrität durch Zugriff auf S überprüfen. Da alle Pfade zum neu eingefügten Tupel mit X-Sperren bis zum Commit (T1) gesperrt sind, müßte T2 so lange warten. Dabei kann entweder ein Deadlock mit dem Rücksetzen einer der beteiligten Transaktionen auftreten oder T2 wird nach T1 in die Serialisierungsreihenfolge eingeordnet.

Für jede Einfüge- oder Löschoperation in eine Relation R gilt, daß alle Pfade, über die das betreffende Tupel zu erreichen ist, gesperrt werden müssen. Da die Pfade nur nacheinander gesperrt werden können, ist zu überlegen, ob Leser von R Phantome sehen können. Nach Schritt 1 ist das neue Tupel bereits in R eingefügt. Da es noch in keiner Zugriffspfadstruktur aufgenommen wurde, könnte es also nur über einen Relationen-Scan aufgefunden werden. Dabei besitzt jedoch die Tupelsperre die erwartete Wirkung.. Greifen parallele Leser Ti nach Schritt 1 und vor Schritt 2 oder 3 über I_S(B1) oder über GF_S (A1, ...,Ak) auf R zu, so sehen sie das neue Tupel noch nicht. Sind die Schritte 2 und 3 erfolgreich, so müssen nachfolgende Leser bis Commit (T1) warten. Die Ti können also mit T1 in einen Deadlock geraten oder sie erscheinen vor T1 in der Serialisierungsreihenfolge.

S2: Löschen eines Tupels aus S

Diese Operation ist hinsichtlich der Überprüfung der Referentiellen Integrität unkritisch:

1. Lösche die Primärschlüssel-Referenz auf S aus I_S(B1) und setze lange X-Sperre.
2. Lösche die Fremdschlüssel-Referenz auf V aus GF_S(A1,...,Ak) und setze lange X-Sperre.
3. Setze lange X-Sperre auf das S-Tupel und lösche es.

S3: Ändern eines Tupels in S

Kosten zur Überprüfung der Referentiellen Integrität fallen nur an, wenn sich der Wert eines oder mehrerer Fremdschlüsselattribute ändert. In unserem einfachen Schema kann man die Änderung eines Tupels in S als die Nacheinanderausführung von Lösch- und Einfüge-Operation in GF_S und als Änderungsoperation auf dem S-Tupel betrachten.

Für die Untersuchung der Kosten wird angenommen, daß die Tupelsperre das feinste Granulat darstellt, d. h., es können keine speziellen Attributsperren gesetzt werden. Es ist zu beachten, daß alle Pfade zum Tupel vor seiner Änderung gesperrt sein müssen:

1. Sperre den Pfad zum Tupel über I_S(B1) und setze eine lange X-Sperre.
2. Aktualisiere GF_S(A1, ...,Ak) und sperre dabei alten und neuen Wert mit langer X-Sperre.
3. Sperre das betroffene S-Tupel mit einer langen X-Sperre und ändere die Attributwerte.
4. Überprüfe die Existenz des neuen Fremdschlüsselwertes durch Zugriff auf GF_V(A1, ...,Ak). Setze dabei auf den hierarchischen Pfad zum gesuchten Schlüsselwert nur kurze S-Sperren.

3.2 Operationen auf Tupeln in der Vater-Relation V

Für die Sohn-Relation werden nach SQL2 Maßnahmen spezifiziert (Kap. 2.1), die bei entsprechenden Modifikationsoperationen (ON DELETE ..., ON UPDATE ...) auf der Vater-Relation ausgelöst werden. Sie können als eine Art systemdefinierter Trigger aufgefaßt werden, mit dem eine Verletzung der Referentiellen Integrität korrigiert und die Datenbank auf einen zulässigen Zustand nachgeführt werden kann.

V1: Einfügen eines Tupels in V

Hinsichtlich der Überprüfung der Referentiellen Integrität ist diese Operation unkritisch:

1. Setze für das V-Tupel eine lange X-Sperre und füge es in die V-Relation ein.
2. Füge Primärschlüssel in GF_V(A1, ...,Ak) ein und überprüfe dabei seine UNIQUE-Eigenschaft. Setze lange X-Sperre für den Schlüsselwert V.(A1, ...,Ak).

V2: Löschen eines Tupels in V

Auch für dieses Operation kann in der Fremdschlüsselklausel von S eine Option angegeben werden:

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

Wird die Option ON DELETE ganz weggelassen, so entspricht das in etwa der RESTRICTED-Option in anderen Sprachentwürfen. Auf abstrakter Sprachebene lassen sich die Operationen zur Sicherung der Referentiellen Integrität wie folgt darstellen:

1. Für die Option RESTRICTED, d.h. eigentlich keine Angabe einer Option:

```
IF EXISTS (  SELECT FROM S
             WHERE    S.A1 <> NULL AND ... AND S.Ak <> NULL AND
                      S.A1 = V.A1 AND ... AND S.Ak = V.Ak
             )
THEN ROLLBACK WORK
```

2. Für die Option SET NULL oder SET DEFAULT:

```
UPDATE S
(S.A1, .., S.Ak) VALUES     (NULL, ..., NULL) beziehungsweise
                            (DEFAULT, .., DEFAULT)
WHERE       S.A1 <> NULL AND ... AND S.Ak <> NULL AND
            S.A1 = V.A1 AND ... AND S.Ak = V.Ak
```

3. Für die Option CASCADE ergibt sich:

```
DELETE FROM S
WHERE       S.A1 <> NULL AND ... AND S.Ak <> NULL AND
            S.A1 = V.A1 AND ... AND S.Ak = V.Ak
```

Für den internen Operationsablauf ergibt sich:

1. Suche das Tupel über GF_V(A1,...,Ak) und lokalisiere es. Sperre den Pfad zum Tupel über GF_V und das Tupel mit langen X-Sperren. Entferne Tupelreferenz aus GF_V und lösche Tupel.
2. Bei Option RESTRICTED: Überprüfe Existenz von Sohn-Tupeln in S durch Zugriff auf GF_S(A1,...,Ak); dabei sind nur kurze S-Sperren zu setzen.
3. Bei Option SET NULL und SET DEFAULT: Der Ablauf der Operation in S entspricht dem bei "Ändern eines Tupels in V", was einem Löschen der Beziehung gleichkommt.
4. Bei Option CASCADE: Die zugehörigen S-Tupel werden über GF_S(A1,...,Ak) aufgesucht. Es werden alle Pfade zu den Tupeln (I_S, GF_S) und die Tupel selbst mit langen X-Sperren belegt. Anschließend werden die Tupel und ihre Referenzen aus den Zugriffspfaden gelöscht.

V3: Ändern eines Tupels in V

In der Fremdschlüssel-Klausel der S-Relation kann die Option

```
[ON UPDATE {SET NULL | CASCADE | SET DEFAULT} ]
```

spezifiziert werden, was die Folgeänderungen auf der S-Relation festlegt. Weglassen bedeutet die Option RESTRICTED. Von der abstrakten Sprachebene läßt sich dieser Fall (ähnlich beim Ändern in S) als Folge einer Lösch- und einer Einfügeoperation darstellen, so daß wir auf die explizite Ausführung der Sprachkonstrukte hier verzichten.
Intern ergibt sich folgender Ablauf:

1. Lokalisiere V-Tupel über GF_V(A1, ...,Ak) und sperre alten und neuen Schlüsselwert in GF_V und V-Tupel mit langen X-Sperren. Ändere Schlüsselwerte und V-Tupel.
2. Bei Option RESTRICTED: Überprüfe Existenz von Sohn-Tupeln in S durch Zugriff auf GF_S(A1,...,Ak); dabei ist nur eine kurze S-Sperre zu setzen.
3. Bei Option SET NULL und SET DEFAULT: Alle Tupeln in S mit dem alten Wert in S.(A1,...,Ak) werden über GF_S(A1,...,ak) aufgesucht. Ihre Werte in werden auf NULL- oder Default-Werte gesetzt. Dabei ist GF_S(A1...,Ak) anzupassen. Dazu sind lange X-Sperren in GF_S, I_S(B1) und auf die betroffenen Tupel zu setzen.
4. Die Option CASCADE erzwingt einen Ablauf der Änderungsoperation in S ähnlich wie 3., wobei der neue Wert von V.(A1,...,Ak) in GF_S und in den qualifizierten S-Tupeln eingesetzt wird.

Zusammenfassend soll noch einmal festgehalten werden, daß zur Überprüfung der Referentiellen Integrität ausschließlich Punktfragen mit voll spezifizierten Suchschlüsseln V.(A1,...,Ak) und S.(A1,...,Ak) auf GF_V und GF_S anfallen. Solche Anforderungen benötigen als Zugriffsunterstützung nicht unbedingt Grid Files, die auf wesentlich allgemeinere Anfragen zugeschnitten sind. Die Auswertung solcher Anfragen ist jedoch Voraussetzung, soll die Integrität erweiterter Beziehungen effizient kontrollierbar bleiben.

4. Überprüfung erweiterter Beziehungen nach SQL2

In den folgenden Abschnitten werden die erweiterten Möglichkeiten des neuen SQL2-Standards im Bereich der Referentiellen Integrität untersucht. Für SQL2 ist dies die Option MATCH PARTIAL und die Möglichkeit, als Referenzziele nicht nur Primärschlüssel sondern auch Schlüsselkandidaten zu benutzen. Beide Erweiterungen beschäftigen sich mit der Zulassung von Null-Werten bei der Betrachtung von Referentieller Integrität.

4.1 MATCH PARTIAL

Die wesentliche Erweiterung zum klassischen Konzept der Referentiellen Integrität, das in etwa durch die Option MATCH FULL erfaßt wird, stellt die Angabe der Option MATCH PARTIAL dar. Während bisher diejenigen Fremdschlüssel, die teilweise mit Null-Werten belegt waren, entweder verboten waren (MATCH FULL) oder keinerlei Beziehungsinformation darstellten (keine MATCH-Klausel), werden diese nun auch als solche Beziehungsinformation interpretiert. Diese Interpretation verläuft so, daß vom Fremdschlüssel eines Tupels t_S der Sohn-Relation, in dem mindestens ein Attribut definiert ist, nur der definierte Anteil, d.h. diejenigen Attribute, die mit Werten

ungleich dem Null-Wert belegt sind, betrachtet werden. Väter im Sinne der Beziehung sind die Tupel t_{V_i} der Vater-Relation, die in den Attributen, die in t_S definiert sind, mit t_S übereinstimmen. Für diejenigen Attribute des Fremdschlüssels, die in t_S nicht definiert sind, ist der Wert in t_{V_i} unerheblich für diese Verbindung. Daraus folgt, daß bei einer solchen Interpretation keine funktionale Beziehung der Vater- und der Sohn-Relation mehr besteht, sondern eine allgemeine n:m-Beziehung. In unserem Modellschema drückt sich eine solche Verbindung wie folgt aus: Ist ein Tupel t_S = (B1, ..., A1, A2, ..., Ak) = (1, ..., 1, NULL, ..., NULL) in der Sohn-Relation S enthalten, so muß es in der Vater-Relation V ein oder mehrere Tupel geben, die im Attribut A1 den Wert 1 aufweisen, d.h., es gibt t_{V_i} = (A1, ...,Ak, ...) = (1, *, ..., *, ...) (* wird als wahlfreie Belegung inklusive dem Null-Wert interpretiert). Damit ist deutlich, daß ein Tupel in S beliebig viele Väter besitzen kann! Um bei Folgeaktionen mit dieser Tatsache umgehen zu können, wird folgende Definition der Verarbeitung in SQL2 [SQL2] zugrundegelegt:

1. Operationen auf den potentiellen Vätern t_{V_i} eines Tupel t_S der Sohn-Relation wirken sich dann und nur dann auf die Söhne aus, wenn entweder
   - der Vater schon vor Beginn der Operation eindeutig bezüglich t_S war, oder
   - einer der Väter während der Ausführung der Operation eindeutig bezüglich t_S wird (z.B. wenn alle Väter t_{V_i} geändert werden)
2. Wenn als Folgeaktion ON UPDATE CASCADE definiert wurde und während einer Änderungs-Operation ein Vater eindeutig wird (die Änderung für die Söhne zum tragen kommt), dann müssen sich alle t_{V_i}, die vor der Operation existiert haben, sich in gleicher Weise ändern. Sonst ist die Operation ungültig.

Diese Semantik der Verarbeitung spielt allerdings bei der Selektion der Tupel für die einzelnen Operationen keine Rolle. Im folgenden werden nun wieder die einzelnen Operationen untersucht, die bei der Wartung der Referentiellen Integrität notwendig werden. Dabei wird vom Modellschema ausgegangen.

V2: Löschen eines Tupels in V

Wenn ein Tupel t_V aus der Vater-Relation gelöscht werden soll, dann ist es, egal in welcher Definition, für die Wahrung der Referentiellen Integrität notwendig, daß die folgenden Tupelmengen ermittelt werden:

1. Alle potentiellen Söhne. Diese Bestimmung kann sehr teuer werden. Es muß dabei folgender Anfragetyp bearbeitet werden:
   In den potentiellen Söhnen sind Teile, mindestens jedoch ein Attribut, des Fremdschlüssels A1, ..., Ak mit Wert ungleich dem Null-Wert belegt. In diesen definierten Werten stimmen sie mit den Werten des Primärschlüssels des Vaters überein. Damit ergeben sich $2^k - 1$ verschiedene Belegungen von Fremdschlüsseln von zu t_V passenden Söhnen in S. Will man dies ohne das MATCH-Prädikat von SQL2 ausdrücken, so hat man die Wahl

von der entsprechenden Anzahl von Einzelanfragen, einer einzigen Anfrage mit einer OR-Verknüpfung der WHERE-Klauseln der Einzelanfragen oder einer UNION-Anfrage der Einzelanfragen. Konzeptionell unterscheiden sich diese Möglichkeiten nur wenig. Diese Gesamtanfrage wird im Rest dieser Untersuchtung als Typ-1-Anfrage bezeichnet.

2. Die potentiellen Väter. Für jede Konfiguration des Fremdschlüssels zur der in S auch tatsächliche Söhne vorhanden sind, muß (eventuell dynamisch) bestimmt werden, ob t_V der eindeutige Vater ist, denn nur dann hat eine Operation auf t_V eine Auswirkung auf die gefundenen Söhne. Dies erfordert im wesentlichen die Ermittlung aller Väter. In einem Tupel t_S, das durch das unter 1 beschriebene Verfahren als Sohn ermittelt wurde, sind nun $m_{def} \leq k$ Attribute des Fremdschlüssels definiert. Um mindestens die Anzahl aller damit in Verbindung stehenden Tupel t_{V_i} (unter anderem auch t_V) zu ermitteln, muß folgende Anfrage ausgewertet werden, die im folgenden als Typ-2-Anfrage bezeichnet:

   SELECT COUNT(*)
   FROM V
   WHERE $V|_{\text{<Definierte Attribute von sk>}} = t_S|_{\text{<Definierte Attribute von fk>}}$

Man erkennt hier deutlich, daß es sich um eine Partial-Match-Anfrage handelt (wie ja der Name der Option schon nahe legt).

Erst die komplette Beantwortung der aus diesen Anfragetypen bestehenden Anfragen gibt eine Antwort darauf, welche Tupel in S prinzipiell betroffen sein könnten (Typ-1-Anfrage) und ob diese Tupel auch tatsächlich betroffen sind (Typ-2-Anfrage). Die einzelnen Folgeaktionen werden dann für diejenigen Söhne durchgeführt, die nur (noch) Verbindung zu dem einen Vatertupel besitzen, das gerade betrachtet wird. Dabei können die Bereichsanfragen, ob eine Menge von Söhnen mit gleichem Fremdschlüssel noch einen weiteren Vater besitzt, unter Umständen durch verschiedene Auswertestrategien optimiert werden. Im ungünstigsten Fall werden jedoch 2^k-1 Punktanfragen in die Sohn-Relation und 2^k-2 Bereichsanfragen in die Vater-Relation benötigt.

V3: Ändern eines Tupels in V

Wie schon oben bemerkt wurde, kann die Änderung eines Schlüsselkandidaten näherungsweise als Lösch- und Wiedereinfüge-Operation betrachtet werden.

S1: Einfügen eines Tupels in S

Im wesentlichen ist hier der Test zu machen, ob ein entsprechender Vater existiert. Dies entspricht gerade dem Typ 2 von Anfragen, die für das Einfügen und Löschen des Vaters benötigt werden.
Damit gilt auch hier, daß Bereichsanfragen unterstützt werden müssen.

S3: Ändern eines Tupels in S

Analog zum Fall von INSERT INTO S.

4.2 Zulassung von Schlüsselkandidaten als Referenzziel

Durch die Zulassung von Schlüsselkandidaten als Referenzziel (Vater-Relation) ergibt sich grundsätzlich die Möglichkeit, Null-Werte im Referenzziel zu gestatten. Diese prinzipielle Zulassung von Null-Werten im Schlüsselkandidaten kann durch die bekannte Bedingung NOT NULL , die bei jedem Attribut vergeben werden kann, weiter verfeinert werden. Für ein System ist die Behandlung eines Schlüsselkandidaten, dessen Attribute alle mit NOT NULL definiert sind, identisch mit der eines Primärschlüssel.
Die oben ausführlich diskutierte MATCH-Klausel beeinflußt auch die Auswertung von Null-Werten im Referenzziel. Ist keine MATCH-Klausel angegeben oder wurde MATCH FULL definiert, so besteht von einem Tupel der Vater-Relation, das im Schlüsselkandidaten einen Null-Wert enthält, zu keinem Tupel der Sohn-Relation eine Beziehung, die von System gewartet wird.
Ist MATCH PARTIAL definiert, so sind potentielle Söhne eines Vater-Tupels t_V diejenigen, die mindestens in den Attributen, die in t_V den Null-Wert enthalten, ebenfalls den Null-Wert enthalten und für die gilt, daß die Attribute, die im jeweiligen Sohn-Tupel definiert sind, auch in t_V definiert sind und dort den gleichen Wert besitzen. Seien beispielsweise von k Attributen eines Schlüsselkandidaten n1 definiert, so gibt es 2^{n1}-1 mögliche Belegungen für den Fremdschlüssel.
Für einen Schlüsselkandidaten aus drei Attributen A1, A2 und A3, die in einem konkreten Tupel t_V mit (1, 1, NULL) belegt sind, ergeben sich (1, 1, NULL), (1, NULL, NULL) und (NULL, 1, NULL) als mögliche Belegungen für Fremdschlüssel in den Söhnen. (1, NULL, 1) ist dagegen keine Belegung für einen potentiellen Sohn, da die Null-Werte immer aus der Richtung des Sohnes interpretiert werden.
Wenn man nun die Parameter für die Behandlung der Null-Werte im Bezug auf die referentielle Integrität zusammenfaßt, so kommt man zu folgender Aufstellung:

Wahlmöglichkeiten in S

In einem einfachen Fremdschlüssel (F) können Null-Werte (N) zugelassen werden oder verboten (O) sein. Der Fremdschlüssel kann nun selbst ein Schlüsselkandidat (S) oder gar ein Primärschlüssel (P) sein. Es ergeben sich dabei die Kombinationen:

FO SO P	Bei einer solchen Definition unterscheidet sich MATCH PARTIAL nicht von MATCH FULL, was die zu betrachtenden Tupelmengen angeht
FN SN	Da Null-Werte in den Söhnen zugelassen sind, gibt es mehrere potentielle Väter für einen Sohn

Wahlmöglichkeiten in V

Bei der Vater-Relation kann man nur entscheiden, ob man einen Schlüsselkandidaten benutzen möchte, der in bestimmten Attributen Null-Werte zuläßt (ein Schlüsselkandidat ohne Null-Werte ist für das System einem Primär-

schlüssel äquivalent). Benutzt man einen solchen Schlüsselkandidaten als Referenzziel einer referentiellen Integritätsbedingung, so reduziert sich dadurch für konkrete Tupel der Vater-Relation (deren Schlüsselkandidat nicht vollständig definiert ist) die Anzahl der möglichen Belegungen der Fremdschlüssel in den Söhnen.

Der Standardfall in konkreten Anwendungen dürfte wahrscheinlich der sein, daß das Referenzziel ein Primärschlüssel ist. Die Eigenschaften des Fremdschlüssels lassen sich nicht so generell bestimmen. Entsteht eine referentielle Integritätsbeziehung aus der Normalisierung, so ist der Fremdschlüssel ein Teil des Primärschlüssels und erlaubt somit keine Null-Werte. Damit wird die Klausel MATCH PARTIAL in solchen Fällen überflüssig.

5. Überblick über das Grid File

In [Hä92] wurde das Grid File unter einer Reihe von mehrdimensionalen Zugriffsstrukturen für den DB-Einsatz vorgeschlagen, wenn raumbezogener Zugriff auf punktförmige Datenobjekte erforderlich wird. Bei der Speicherabbildung bewahrt es die Topologie der Objekte des Datenraumes D; sie werden Buckets[1] zugeordnet, wobei eine k-dimensionale Clusterbildung für die gespeicherten Datensätze erreicht wird. Eine schiefe und ungleichförmige Verteilung der Objekte des Datenraumes wird durch das Grid Directory GD adaptiert, so daß der Belegungsgrad der Buckets nicht beliebig klein werden kann. Die Zahl der Einträge im Grid Directory kann jedoch sehr groß werden, da jeder Grid Block GB einem Eintrag in GD entspricht. Ungünstige Verteilungen erzwingen das Einfügen neuer Rasterlinien (Splitt-Vorgänge), wobei immer eine ganze Dimension verfeinert wird, obwohl nur ein Grid Block, genauer gesagt das zugehörige Bucket, eine Neuaufteilung erzwingt. Bild 2 illustriert das Abbildungsprinzip beim Grid File. Über jede der k Dimensionen, die sich aus k ausgewählten Attributen (Schlüssel) des Datensatzes ergeben, bietet es symmetrischen und gleichförmigen Zugriff, wobei unabhängig von Schlüsselverteilungen sowie Einfüge- und Löschreihenfolgen eine balancierte Zugriffsstruktur garantiert wird.

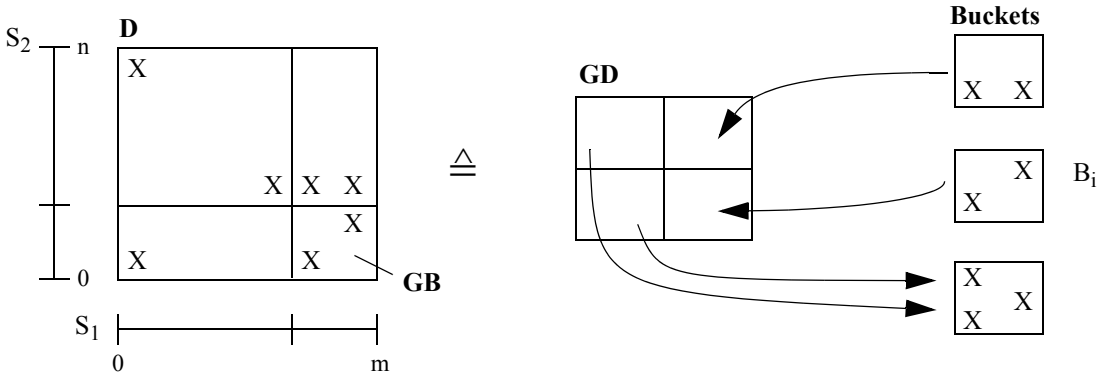


Bild 2: Abbildungsprinzip beim Grid File

Bild 3 soll eine mögliche Implementierung des Grid Files skizzieren. Dazu werden k Skalierungsvektoren S_i zur Darstellung des Rasters auf D und zu seiner Zuordnung auf GD, das Grid Directory selbst und der Bucketbereich

1. Der Begriff Bucket anstelle von Block oder Seite soll hier betonen, daß sein Inhalt völlig ungeordnet ist.

benötigt. Die Situation b in Bild 3 ist aus einem einzigen Grid Block durch zwei Verfeinerungen im Bereich a-z und durch eine Verfeinerungen im Bereich 0-9 entstanden. Dabei sind die GD-Einträge von 1 auf 9 gewachsen, d.h., die GD-Implementierung, durch die eine k-dimensionale dynamische Matrix auf einen linearen Vektor abzubilden ist, muß Verfeinerungen und Vergröberungen in D ohne allzu hohen Reorganisationsaufwand bewerkstelligen. Durch die Skalierungsvektoren werden die Rastereinteilungen von D dimensionsweise repräsentiert. Da die Rasterlinien geordnet sind und Wertebereichen entsprechen, müssen die Skalierungseinträge, die diese Wertebereiche kennzeichnen, auch der Rasterordnung entsprechen. Durch eine indirekte Zuweisung von S_i-Eintrag zu GD-Zeile oder -Spalte in Bild 3 ((k-1)-dimensionale Hyperebene im allgemeinen Fall) können die S_i an der erforderlichen Stelle verfeinert werden [Sa86]. GD aber kann durch Anlagerung eines Vektors am Rand erweitert werden. Bei Vergröberung würde der entsprechende Vektor in GD bleiben, aber mit NIL-Werten aufgefüllt werden. In beiden Fällen würde die Strukturmodifikation von GD die Namen der bestehenden GD-Einträge stabil lassen. Dies ist eine wichtige Eigenschaft, die beispielsweise die Entwicklung von Synchronisationsprotokollen erleichtert [Hä92].

Gewährte Sperren auf dem GD oder auf GD-Einträgen müssen bis Commit gehalten werden. Strukturänderungen an GD, die durch Kurzzeitsperren abgesichert werden, können so vorgenommen werden (Situation a→b), ohne die Namensstabilität in Frage zu stellen. In diesem Beispiel ändert sich nur GD(1,2), der auf das die Verfeinerung auslösende Bucket A verwies.

Wie in Abschnitt 2.1 dargestellt, sind die Tupel der Relation in einer separaten Speicherungsstruktur abgelegt. Die Buckets des zugehörigen Grid File enthalten Verweise, bestehend auf k Schlüsselwerten und einem TID, auf die Tupel der Relation.

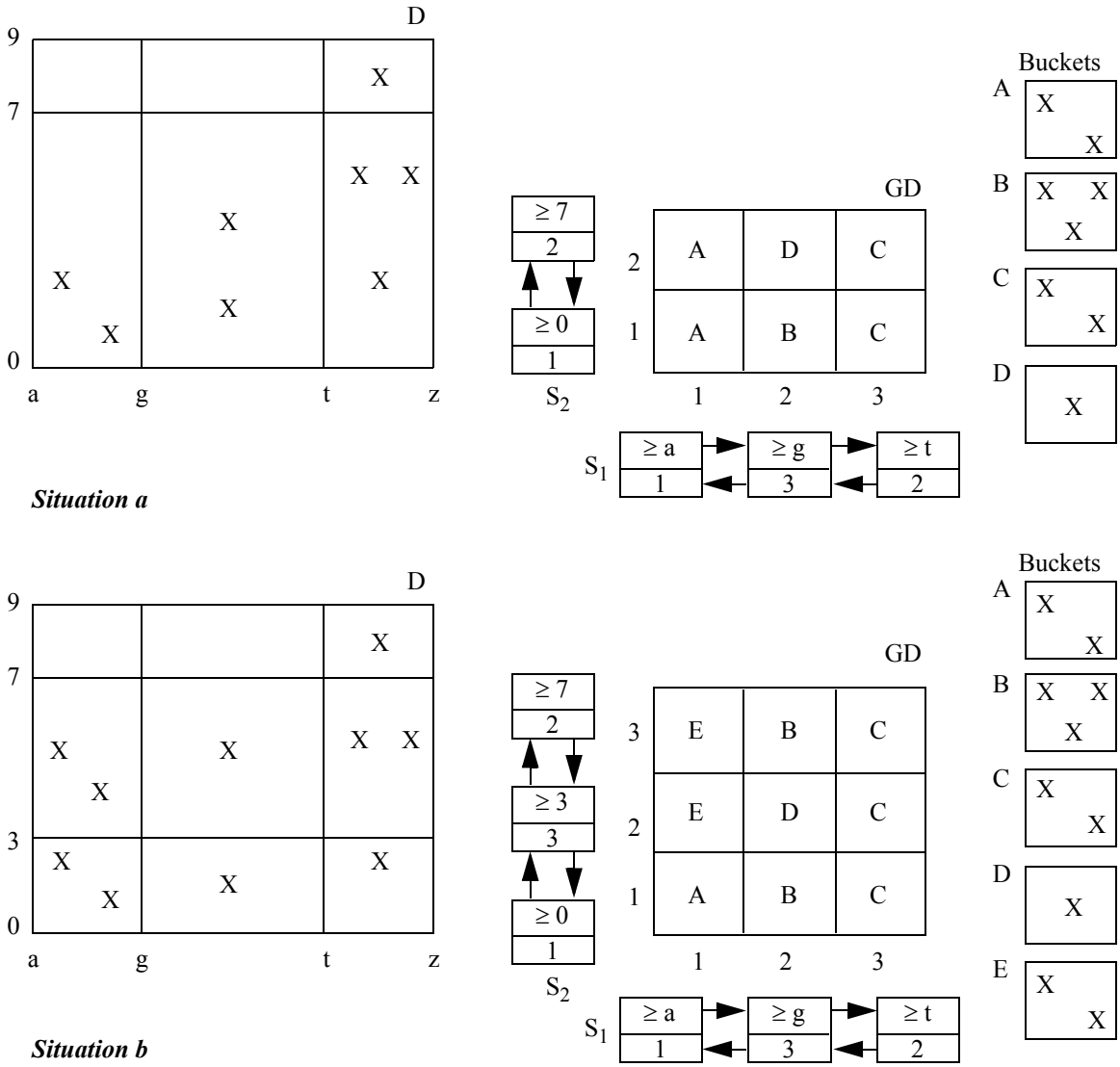


Bild 3: Grid-File Implementierung mit doppelt geketteten Listen als Skalierungsvektoren

6. Kostenabschätzungen

Zur Bewertung der für die Sicherung der Referentiellen Integrität anfallenden Kosten sollen einfache Modelle für den Einsatz von Grid Files entwickelt werden. Dabei unterstellen wir eine Situation gemäß dem Schema in Kap. 2.1, wo auf den beiden Ausgangsrelationen V und S

- ein Grid File $GF_V(A1, ...,Ak)$ für Primärschlüsselzugriff in V
- eine Indexstruktur $I_S(B1)$ für Primärschlüsselzugriff in S
- ein Grid File $GF_S(A1, ...,Ak)$ für Fremdschlüsselzugriff in S

angelegt sind.

Alle Kosten für die Wartung dieser Strukturen werden dem Konto der Integritätssicherung zugeschlagen. Natürlich werden im normalen DB-Betrieb diese Kosten (teilweise) amortisiert durch die zusätzliche Nutzung solcher Strukturen für die Anfrageverarbeitung. Diese ausgleichenden Aspekte sollen jedoch hier außer acht bleiben.

Weiterhin werden keine Annahmen über das erwartete Lastprofil getroffen, z.B. über Häufigkeiten und Lokalität von Operationen, so daß mit unseren Kostenabschätzungen keine Ermittlung von Betriebskosten oder von relativen Mehrkosten für die Integritätssicherung durchführbar ist. Wir können hier nur "statisch" einzelne Aktualisierungsoperationen auf Relationen unseres Modellschemas betrachten und den maximalen, der Integritätssicherung zurechenbaren Aufwand abschätzen, der dann beim Vergleich und bei der Bewertung mit anderen Strukturen herangezogen werden kann. Aspekte des Mehrbenutzerbetriebs können ebenfalls nur statisch und relativ grob berücksichtigt werden. Genaue quantitative Aussagen über den Einfluß der Maßnahmen zur Integritätssicherung auf das Systemverhalten würden umfangreiche Simulationen mit genauen Zugriffspfadmodellen und einer detaillierten Nachbildung der Synchronisationsprotokolle sowie einer hinreichend genauen Lastmodellierung verlangen. Als Ergebnis könnten dann Aussagen über Konflikthäufigkeiten und ihre Dauer sowie über das Auftreten von Deadlocks erwartet werden. Da diese Modellierungsgenauigkeit im Rahmen dieser Betrachtungen nicht zu erreichen ist, bescheiden wir uns mit der Angabe von Sperranforderungen.

Jede Operation zielt auf ein Tupel. Die Aktualisierung dieses Tupels fordert, wenn seine Adresse schon bekannt ist, das Sperren des Tupels gemäß der Sperrhierarchie "DB - Segment - Relation - Tupel" sowie das Lesen und das Schreiben einer Seite:

$$IO_{basis\text{-}read} = 1$$
$$IO_{basis\text{-}write} = 1$$
$$LOCK_{basis} = 4$$

Daneben ist auf weitere systeminterne Verwaltungsinformation zuzugreifen, z.B. auf Freispeicherverwaltungs- und Adressierungstabellen, die bei Einfüge- und Löschoperationen eines Tupels zu modifizieren sind. Die dabei anfallenden Kosten sind stark implementierungsabhängig und sollen deshalb nicht weiter quantifiziert werden. Zu diesem Basisaufwand bestimmen wir den Mehraufwand mittels zweier Kostenmaße:

1. Anzahl der zusätzlichen logischen Seitenzugriffe beim Lesen und beim Schreiben.
   Die Anzahl der logischen Referenzen lassen keinen direkten Rückschluß zu, wieviele physische E/A erforderlich sind. Zugriffslokalität tritt sicher verstärkt in Zugriffspfadstrukturen auf. Da wir jedoch keine Annahmen über Verarbeitungsreihenfolgen und -kontexte treffen, sind Aussagen über die Häufigkeit physischer E/As nicht zulässig. Der zusätzliche Instruktionsbedarf ist jedoch stark korreliert mit der Anzahl der logischen Seitenreferenzen.

2. Anzahl zusätzlicher Sperranforderungen

   Diese Größe ist ein Indikator für den zusätzlichen Instruktionsaufwand zur Synchronisation. Bei Aussagen über die Konfliktgefahr solcher Anforderungen müßten neben der Häufigkeit auch das Sperrgranulat usw. berücksichtigt werden. Kurzzeitsperren (Latches) auf Indexstrukturen oder Grid Files gehen nicht in die Betrachtung ein, da sie vergleichweise geringe Kosten und eine zu vernachlässigende Konfliktwahrscheinlichkeit verursachen.

Damit ergeben sich drei Kostenformeln für jede der in Kap. 2 diskutierten sechs Modifikationsoperationen, die Zusatzaufwand für die Sicherung der Relationalen Invarianten hervorrufen; sie werden mit $IO_{<op>\text{-read}}$, $IO_{<op>\text{-write}}$ und $LOCK_{<op>}$ bezeichnen, wobei die logischen Seitenreferenzen für Lesen und Schreiben sowie die Sperranforderungen ohne Kurzzeitsperren für Zugriffspfad- oder Systempufferoperationen gezählt werden.

Für die Bestimmung des Mehraufwandes für die Sicherung der Integrität werden folgende allgemeine Parameter verwendet. Bei den untersuchten Operationen haben beide Grid Files die gleiche DImension und die gleiche Schlüsselstruktur:

k	Dimension eines Grid Files
b	Bucketfaktor: maximale Anzahl von Einträgen in einem Bucket
h_S^*	Baumhöhe für $I_S(B1)$
$2k_I, 2k_I^*$	max. Anzahl von Indexeinträgen pro Seite für Index $I_S(B1)$ (innerer bzw. Blattknoten)
N_S, N_V	Anzahl der Tupel in Relation S bzw. V
n	durchschnittliche Anzahl von Söhnen pro Vater
β_S, β_V	Belegungsgrad für Buckets in S bzw. V

Die Werte für k_I und k_I^* lassen sich in einfacher Weise aus der Schlüssellänge L_{B1}, der Länge eines Zeigers L_Z oder eines TIDs, der Länge eines Tupels ohne Schlüssel L_D und der Seitengröße L_S ermitteln [HR91]:

$$2k_I = \left\lfloor \frac{L_S - 2L_Z}{L_{B1} + L_Z} \right\rfloor$$

$$2k_I^* = \left\lfloor \frac{L_S - 3L_Z}{L_{B1} + L_D} \right\rfloor$$

Werden im B*-Baum in den Blättern TIDs statt der Tupel gespeichert, so kann L_D durch L_Z ersetzt werden. Mit k_I, k_I^* sowie der Tupelanzahl N_S kann für einen Index die Anzahl der logischen Seitenzugriffe (die Baumhöhe h^*) beim Schlüsselzugriff wie folgt berechnet werden:

$$h_S^* \leq 2 + \log_{k_I+1}(N_S / (2k_I^*))$$

Die durchschnittliche Anzahl von Söhnen pro Vater ergibt sich zu:

$$n = \frac{N_S}{N_V}$$

6.1 Wartung der “klassischen “ Referentiellen Integrität

Beim Zugriff über GF_S und GF_V werden im Rahmen der Sicherung der Referentiellen Integrität nur Punktfragen angewendet. Bei GF_V wird man in der Regel, pathologische Verteilungen einmal ausgeschlossen, auf einen GD-Eintrag und folglich auf ein Bucket zugreifen, wo sich das TID des gesuchten Tupels befindet. Zur Ermittlung aller Söhne wird ebenfalls eine Punktfrage benutzt, die auf einen oder wenige GD-Einträge führt. In den Buckets sind $n=N_S/N_V$ Einträge aufzusuchen, die dann auf die n Datensätze führen. Für die n Einträge in den Buckets liegt eine Clusterbildung vor, so daß für $n \leq 100$ in der Regel $Z_{GFS}(=2)$ Seitenzugriffe auf dem GF_S und n Seitenzugriffe auf die Tupel anfallen. Bei Punktfragen ergeben sich normalerweise minimale GF-Zugriffskosten, nämlich zwei Seitenzugriffe; dominiert wird dieser Kostenanteil von den Zugriffskosten auf die Tupel.

Auch die Sperrkosten lassen sich in diesen Fällen leicht ermitteln. Dabei fallen für das Grid File eine GD-Sperre, eine Sperre auf den GD-Eintrag und eine bzw. n Sperren auf den Tupeln an (beim Schreibzugriff). Kurze Lesesperren auf GD-Einträgen sind dann ausreichend, wenn nur die Existenz eines Tupels (Vater) überprüft werden soll. Beim Sperren über einen Index werden die in [HR91] eingeführten Protokolle (“Sperren des nächsten Schlüsselwertes”) eingesetzt. Dabei sind bei Aktualisierungsoperationen jeweils 3 Sperren (für I_S und für zwei Schlüsselwerte) und bem Lesen jeweils 2 Sperren anzufordern. Zur Differenzierung der Sperrkosten unterscheiden wir die Anzahl der Sperraufrufe für Index, für Grid Files und für die Relationen durch L_{I_S}, L_{GF_S}, L_{GF_V} und L_S bzw. L_V.

Kosten für Strukturmodifikationen in I_S oder GF_S bzw. GF_V haben auf die mittleren Aktualisierungskosten keinen größeren Einfluß. Wir unterstellen vorwiegend ein Wachstum der Datenstrukturen und berücksichtigen nur beim Einfügen einen Kostenanteil für Strukturmodifikationen. Im Mittel soll nach 2k*(b) Einfügungen ein Splitt-Vorgang hervorgerufen werden, wobei 3 Seiten zu schreiben sind; Fortpflanzungen dieses Splitt-Vorganges werden nicht im Kostenmodell berücksichtigt.

S1: Einfügen eines Tupels in S

Zur Lokalisierung der Einfügepunkte in I_S und GF_S sowie zur Prüfung der Schlüsselexistenz in GF_V werden diese Strukturen einmal durchlaufen. I_S und GF_S sind dabei zu aktualisieren:

$$IO_{S1-read} = h_S^* + 2 + 2$$

$$IO_{S1-write} = \left(1 + \frac{3}{2k_i^*}\right) + \left(1 + \frac{3}{b}\right)$$

$$LOCK_{S1} = L_{I_S} + L_{GF_S} + L_{GF_V} + L_V = 3 + 2 + 2 + 3$$

S2: Löschen eines Tupels in S

Die Vater-Relation ist hierbei nicht betroffen.

$$IO_{S2-read} = h_S^* + 2$$

$$IO_{S2-write} = 1 + 1 = 2$$

$$LOCK_{S2} = L_{I_S} + L_{GF_S} = 3 + 2$$

Kosten zum Entfernen von leeren Blattseiten im Index oder von leeren Buckets wurden hier negiert.

S3: Ändern eines Tupels in S

Die Änderung soll Attribute des Fremdschlüssels betreffen; sonst fallen keine hier anrechenbare Kosten an.

$$IO_{S3-read} = h_S^* + 3 + 2$$

$$IO_{S3-write} = \left(1 + \frac{3}{b}\right) + 1$$

$$LOCK_{S3} = L_{I_S} + L_{GF_S} + L_{GF_V} + L_V = 3 + 3 + 2 + 3$$

V1: Einfügen eines Tupels in V

Der Zusatzaufwand beschränkt sich hier auf die Aktualisierung von GF_V.

$$IO_{V1-read} = 2$$

$$IO_{V1-write} = 1 + \frac{3}{b}$$

$$LOCK_{V1} = L_{GF_V} = 2$$

V2: Löschen eines Tupels in V

Die Kosten dieser Operation hängen stark von der gewählten Option für die Behandlung der abhängigen Tupel in S ab. Es sind Sperranforderungen für die Sperrhierarchie von S zu berücksichtigen (L_S):

a) RESTRICTED

$$IO_{V2-read} = 2 + 2$$

$$IO_{V2-write} = 1$$

$$LOCK_{V2} = L_{GF_V} + L_{GF_S} + L_S = 2 + 2 + 3$$

b) SET NULL/DEFAULT

Hierbei ist für jedes der n abhängigen Tupel in S der Fremdschlüssel auf den NULL- bzw. DEFAULT-Wert zu setzen. Dabei seien in GF_S zweimal $N_{BS}/N_V = \left\lceil \frac{n}{b \cdot \beta} \right\rceil$ (für jeden Vater fällt eine gleiche Anzahl an Buckets in GF_S an) Buckets zu modifizieren (aktuellen Wert löschen und NULL/DEFAULT-Wert eintragen). Außerdem sind die S-Tupel aufzusuchen (über GF_S) und zu ändern:

$$IO_{V2-read} = 2 + 2 \cdot \left(1 + \left\lceil \frac{n}{b \cdot \beta} \right\rceil\right) + n$$

$$IO_{V2-write} = 1 + \left\lceil \frac{n}{b \cdot \beta} \right\rceil \cdot \left(1 + 1 + \frac{3}{b}\right) + n$$

$$LOCK_{V2} = L_{GF_V} + L_{GF_S} + L_{I_S} + L_S = 2 + \left(1 + 2\left\lceil \frac{n}{b \cdot \beta} \right\rceil\right) + 3n + (3 + n)$$

In I_S werden n Werte gesperrt, was 3n Sperranforderungen verlangt. Außerdem sind n Tupel in S zu sperren (3+n).

c) CASCADE

Hierbei müssen n Sohn-Tupel gelöscht und ihre Einträge aus I_S und GF_S gelöscht werden:

$$IO_{V2-read} = 2 + \left(1 + \left\lceil \frac{n}{b \cdot \beta} \right\rceil\right) + h_S^* \cdot n + n$$

$$IO_{V2-write} = 1 + \left\lceil \frac{n}{b \cdot \beta} \right\rceil + n + n$$

$$LOCK_{V2} = L_{GF_V} + L_{GF_S} + L_{I_S} + L_S = 2 + \left(1 + \left\lceil \frac{n}{b \cdot \beta} \right\rceil\right) + 3n + (3 + n)$$

Es wird hier angenommen, daß die zu löschenden Schlüsselwerte in I_S gleichverteilt sind und daß die Löschungen jeweils verschiedene Seiten betreffen.

V3: Ändern eines Tupels in V

Die Berechnung der Kosten ist ähnlich der von V2.

a) RESTRICTED

$$IO_{V3-read} = (2 + 2) + 2$$

$$IO_{V3-write} = \left(1 + \frac{3}{b} + 1\right)$$

$$LOCK_{V3} = L_{GF_V} + L_{GF_S} + L_S = 2 + 2 + 3$$

Die zusätzlichen Schreibvorgänge betreffen hier ausschließlich GF_V.

b) SET NULL/DEFAULT

Wie in V2 ist für jedes der n abhängigen Tupel in S der Fremdschlüssel auf den NULL- bzw. DEFAULT-Wert zu setzen:

$$IO_{V3-read} = (2 + 2) + 2 \cdot \left(1 + \left\lceil \frac{n}{b \cdot \beta} \right\rceil\right) + n$$

$$IO_{V3-write} = \left(1 + 1 + \frac{3}{b}\right) + \left\lceil \frac{n}{b \cdot \beta} \right\rceil \cdot \left(1 + 1 + \frac{3}{b}\right) + n$$

$$LOCK_{V3} = L_{GF_V} + L_{GF_S} + L_{I_S} + L_S = 2 + \left(1 + 2\left\lceil \frac{n}{b \cdot \beta} \right\rceil\right) + 3n + (3 + n)$$

c) CASCADE

Hier muß der geänderte Wert des Primärschlüssels eines V-Tupels in den Fremdschlüsseln von n S-Tupeln nachgetragen werden:

$$IO_{V3-read} = (2+2) + \left(1 + \left\lceil \frac{n}{b \cdot \beta} \right\rceil\right) + h_S^* \cdot n + n$$

$$IO_{V3-write} = \left(1 + 1 + \frac{3}{b}\right) + \left\lceil \frac{n}{b \cdot \beta} \right\rceil \cdot \left(1 + 1 + \frac{3}{b}\right) + n$$

$$LOCK_{V3} = L_{GF_V} + L_{GF_S} + L_{I_S} + L_S = 3 + \left(1 + 2\left\lceil \frac{n}{b \cdot \beta} \right\rceil\right) + 3n + (3+n)$$

Alle n S-Tupel müssen über alle Pfade, auf denen sie erreicht werden können, gesperrt werden. In I_S wird nicht geändert, jedoch müssen alle Primärschlüsselwerte der betroffenen S-Tupel vor Modifikation der Tupel gesperrt sein.

Diskussion

In Tabelle 1 sind die Zusatzkosten für die Integritätssicherung in unserem Anwendungsbeispiel übersichtlich zusammengefaßt. Dabei wurden für die folgenden Parameter der Speicherungsstrukturen bereits typische Werte eingesetzt, um die Lesbarkeit zu erhöhen.

$k_I^* = 150$

$b = 100$ (etwa für L_S = 2 KBytes und 20 Bytes pro Eintrag (k=2 oder 3 Werte und TID))

$\beta = 0.8$

$h_S^* = 3$

	Einfügen	Löschen	Ändern
Vater	2	R: 4 S: $4 + 2\left\lceil \frac{n}{80} \right\rceil + n$ C: $3 + \left\lceil \frac{n}{80} \right\rceil + 4n$	R: 6 S: $6 + 2\left\lceil \frac{n}{80} \right\rceil + n$ C: $5 + \left\lceil \frac{n}{80} \right\rceil + 4n$
Sohn	7	5	6

a) zusätzliche lesende Seitenzugriffe

	Einfügen	Löschen	Ändern
Vater	1.03	R: 1 S: $1 + 2{,}03\left\lceil \frac{n}{80} \right\rceil + n$ C: $1 + \left\lceil \frac{n}{80} \right\rceil + 2n$	R: 2,03 S: $2{,}03\left(1 + \left\lceil \frac{n}{80} \right\rceil\right) + n$ C: $2{,}03\left(1 + \left\lceil \frac{n}{80} \right\rceil\right) + n$
Sohn	2.04	2	2.03

b) zusätzliche schreibende Seitenzugriffe

	Einfügen	Löschen	Ändern
Vater	2	R: 7 S: $6 + 2\left\lceil \frac{n}{80} \right\rceil + 4n$ C: $6 + \left\lceil \frac{n}{80} \right\rceil + 4n$	R: 7 S: $6 + 2\left\lceil \frac{n}{80} \right\rceil + 4n$ C: $7 + 2\left\lceil \frac{n}{80} \right\rceil + 4n$
Sohn	10	5	6

c) zusätzliche Sperranforderungen

Tabelle 1: Zusatzkosten für die Integritätssicherung

Als einziger Parameter wurde $n = N_S/N_V$ als die mittlere Anzahl von Söhnen pro Vater in der Tabelle belassen. Es wird hier wiederum deutlich, daß die beiden unkritischen Operationen V1 (Einfügen eines Vaters) und S2 (Löschen eines Sohnes) die geringsten Zusatzkosten verursachen. S1 (Einfügen eines Sohnes), S3 (Ändern eines Sohnes), V2 und V3 (Löschen, Ändern eines Vaters) mit der Option RESTRICTED rufen moderate und konstante Kosten hervor, da nur ein einfacher Existenztest zur Überprüfung der Referentiellen Integrität erforderlich ist. Erheblicher Aufwand ist dagegen bei V2 und V3 (Löschen , Ändern eines Vaters) mit den Optionen SET NULL/

DEFAULT und CASCADE zu erwarten, da alle Kostenindikatoren ein mit n lineares Kostenwachstum anzeigen, das den Wert von 4n erreichen kann. In Zahlenwerten ist diese Abhängigkeit der Kosten in Tabelle 2 illustriert.

	Löschen eines V-Tupels			Ändern eines V-Tupels		
$n = N_S/N_V$	1	10	100	1	10	100
RESTRICTED	4	4	4	6	6	6
SET NULL/DEF.	7	16	108	9	18	110
CASCADE	8	44	405	10	46	407

a) Anzahl zusätzlicher Lesezugriffe

	Löschen eines V-Tupels			Ändern eines V-Tupels		
$n = N_S/N_V$	1	10	100	1	10	100
RESTRICTED	1	1	1	2.03	2.03	2.03
SET NULL/DEF.	4.03	13.03	105.06	5.06	14.06	106.09
CASCADE	4	22	203	5.06	14.06	106.09

b) Anzahl zusätzlicher Schreibzugriffe

	Löschen eines V-Tupels			Ändern eines V-Tupels		
$n = N_S/N_V$	1	10	100	1	10	100
RESTRICTED	7	7	7	7	7	7
SET NULL/DEF.	12	48	410	12	48	410
CASCADE	11	47	408	13	49	411

c) Anzahl der zusätzlichen Sperranforderungen

Tabelle 2: Zahlenbeispiel für die zusätzlichen Mehrkosten

Durch Tabelle 2 wurde noch einmal herausgestellt, daß die Optionen SET NULL/DEFAULT und CASCADE bei Lösch- oder Änderungsoperationen des Vaters erhebliche Kosten insbesondere bei größerem n verursachen können. Nimmt man die Anzahl der Sperranforderungen als ersten Indikator für das Konflikt- oder Blockierungspotential solcher Operationen, so wird dadurch klar, daß ihrer Synchronisation besonderes Augenmerk zu schenken ist.

6.2 Wartung der erweiterten referentiellen Beziehungen

Für die Berechnung der erweiterten referentiellen Beziehungen müssen wir unsere Kostenmodelle verfeinern. Dazu benötigen wir zunächst einige weitere Parameter:

N_{BS}, N_{BV}	Anzahl der Buckets in GF_S bzw. GF_V
S_{Si}, S_{Vi}	Anzahl der Skalierungswerte (Wertintervalle) für das i-te Attribut der Relation S bzw. V
α_S, α_V	durchschnittliche Anzahl der GD-Einträge pro Bucket

Mit diesen Größen lassen sich weiterhin folgende Zusammenhänge analytisch festhalten:

$$N_{BV} = \frac{N_V}{b \cdot \beta_V}$$

$$N_{BS} = \frac{N_S}{b \cdot \beta_S}$$

Wegen der n:1-Abbildung zwischen GD-Einträgen und Buckets ist $\alpha \geq 1$. Als Anzahl der GD-Einträge erhält man:

$$N_{GDV} = \prod_{i=1}^{k} S_{Vi} = S_V^k \quad \text{mit } S_{Vi} = S_V,\ i = 1...k$$

$$N_{GDV} = \prod_{i=1}^{k} S_{Si} = S_S^k \quad \text{mit } S_{Si} = S_S,\ i = 1...k$$

$$\alpha_S = \frac{N_{GDS}}{N_{BS}} \quad \text{oder} \quad N_{GDS} \geq N_{BS}$$

$$\alpha_V = \frac{N_{GDV}}{N_{BV}} \quad \text{oder} \quad N_{GDV} \geq N_{BV}$$

Um die Komplexität der Modellbetrachtungen zu beschränken, sind weitere Modellparameter festzulegen. Relativ einfach läßt sich noch der Fall k=2 behandeln, jedoch verbirgt er wesentliche Aspekte bei der Überprüfung erweiterter Beziehungen. Deshalb beziehen wir uns bei den weiteren Betrachtungen auf den Fall k=3.

In unserem Modell ist der Bezugsschlüssel in der Vater-Relation Primärschlüssel, d.h., alle seine Attributwerte sind definiert, was wir durch die Konfiguration (W, W, W) bei k=3 ausdrücken wollen. Als Konflguration bezeichnen wir eine im Rahmen der erweiterten Beziehung erlaubte Schlüsselzusammensetzung, wobei W für einen gültigen Wert des Wertebereichs steht und N für den undefinierten Wert NULL. Für die Vater-Relation ist nur obige Konfiguration zulässig. In der Sohn-Relation werden im Fremdschlüssel alle Kombinationen von W und N für eine erweiterte referentielle Beziehung, außer der vollständig undefinierten Kombination (N, ..., N), ausgewertet.

Ein Fremdschlüssel, der nach einer der (2^k-1) relevanten Konfigurationen aufgebaut ist, unterliegt den Überprüfungsregeln erweiterter Beziehungen. Dabei können verschiedene Konfigurationen unterschiedliche Kosten hervorrufen, so daß hinsichtlich der anfallenden Kosten eine Fallunterscheidung angezeigt ist. Bei k=3 gehören Fremdschlüssel einer der folgenden Konfigurationen an:

1. W, W, W
2. W, W, N
3. W, N, W
4. W, N, N
5. N, W, W
6. N, W, N
7. N, N, W

Wird beispielsweise ein Sohn mit Fremdschlüssel (1, 5, N) eingefügt, so entspricht das der Konfiguration (W, W, N). Das Sohn-Tupel wird im GF_S eingetragen, wobei hier N als spezieller Wert des entsprechenden Attributs angesehen wird. Jeder Skalierungsvektor berücksichtigt die Abbildung von N, so daß diese NULL-Werte zum Aufsuchen herangezogen werden können. Beim Existenztest des Vaters, muß in der V-Relation nach mindestens ei-

nem Tupel gesucht werden, dessen Primärschlüssel eine (teilweise) Übereinstimmung mit (1, 5, N) liefert. In diesem Fall ist eine Bereichsfrage mit (1, 5, N) auf GF_V erforderlich, wobei bei N alle Werte der betreffenden Dimension ausgewählt werden.

Für die Zugriffskosten beim Grid File ist unter der Annahme der Gleichverteilung der Attributwerte für jedes Attribut und der Unabhängigkeit der Attribute die Anzahl der GD-Einträge, die sich für eine Anfrage qualifizieren, wichtig. Bei einer Punktfrage mit (W, W, W) erhält man einen GD-Eintrag. Bei Fragen mit einem unbeschränkten Attribut A_i (Konfigurationen mit einem N) erhält man S_{Si} bzw. S_{Vi} GD-Einträge. Schließlich liefern Fragen mit zwei unbeschränkten Attributen A_i und A_j (Konfigurationen mit zwei N) $S_{Si} \cdot S_{Sj}$ bzw. $S_{Vi} \cdot S_{Vj}$ GD-Einträgen.

Nach diesen Vorbemerkungen können konkrete Kostenmodelle erstellt werden.

S1: Einfügen eines Tupels in S

Fällt das S-Tupel mit seinem Fremdschlüssel in Konfiguration 1, so bleiben die Kostenmodelle aus Kap. 6.1 gültig (E_{GDV}=1).

Es errechnen sich beim Existenztest vom V-Tupel als Anzahl der qualifizierten GD-Einträge bei den Konfigurationen:

2: $E_{GDV} = S_{V3}$
3: $E_{GDV} = S_{V2}$
5: $E_{GDV} = S_{V1}$

und bei den Konfigurationen

4: $E_{GDV} = S_{V2} \cdot S_{V3}$
6: $E_{GDV} = S_{V1} \cdot S_{V3}$
7: $E_{GDV} = S_{V1} \cdot S_{V2}$.

Damit erhält man folgende Kostenmodelle, wobei E_{GDV} die maximale Anzahl der auf GD_V-Einträgen anfallender Zugriffe und Sperranforderungen beschreibt. Die minimale Anzahl ist dabei 1.

$$IO_{S1-read} = h_S^* + 2 + (1 + \lceil E_{GDV} / \alpha_V \rceil)$$

$$IO_{S1-write} = (1 + 3/(2 \cdot k_I^*)) + (1 + 3/b)$$

$$LOCK_{S1} = L_{I_S} + L_{GF_S} + L_{GF_V} + L_V = 3 + 2 + (1 + E_{GDV}) + 3$$

S2: Löschen eines Tupels in S

Im Vergleich zur Überprüfung der klassischen Referentiellen Integrität fallen keine Mehrkosten an.

S3: Ändern eines Tupels in S

Die Änderung soll nur Attribute des Fremdschlüssels betreffen. Die Änderungskosten in S entsprechen dem herkömmlichen Fall: in GF_S wird ein Schlüsselwert-TID-Eintrag aus- und ein neuer eingetragen. Der Test auf Existenz eines Vaters mit dem neuen Schlüsselwert könnte wie unter S1 durchgeführt werden, so daß sich folgende Kostenmodelle ergeben:

$$IO_{S3-read} = h_S^* + 3 + (1 + \lceil E_{GDV} / \alpha_V \rceil)$$

$$IO_{S3-write} = \left(1 + \frac{b}{3}\right) + 1$$

$$LOCK_{S3(1)} = L_{I_S} + L_{GF_S} + L_{GF_V} + L_V = 3 + 3(2) + (1 + E_{GDV}) + 3$$

Da Bereichsfragen teuer sind, lohnt es sich, den Existenztest des Vaters zu optimieren. Ändert sich der Fremdschlüssel von beispielsweise (1, 5, N) nach (1, N, N) oder (N, 1, N), so ist kein expliziter Existenztest des Vaters erforderlich, weil ja der bisherige Vater unter der neuen Beziehung Vater bleibt. Generell kann man sagen, daß ein Existenztest unterbleiben kann, wenn bei der Änderung des Fremdschlüssels Einzelwerte einschließlich der NULL-Werte erhalten bleiben oder auf NULL gesetzt werden. Sonst werden Anfragen entsprechend der Konfiguration, in die der neue Fremdschlüsselwert fällt, erforderlich.

V1: Einfügen eines Tupels in V

Diese Operation ist auch bezüglich der erweiterten referentiellen Beziehungen unkritisch. Die Kosten entsprechen denen in Kap. 6.1.

V2: Löschen eines Tupels in V

Die beiden folgenden Operationen werden sehr komplex, da hierbei die erweiterten referentiellen Beziehungen zu S automatisch zu warten sind. Als erster Schritt sind die Söhne des zu löschenden V-Tupels t_V unter der erweiterten Beziehung zu bestimmen. Aus den aktuellen Werten des Primärschlüssels lassen sich alle zulässigen Fremdschlüsselwerte berechnen, indem die aktuellen Werte auf die W-Platzhalter der 2^k-1 Konfigurationen vererbt werden. Bei k=3 und einem Primärschlüssel (1, 2, 3) ergeben sich folgende Fremdschlüssel in S, die hinsichtlich der erweiterten Beziehung überprüft werden müssen:

1. 1, 2, 3
2. 1, 2, N
3. 1, N, 3
4. 1, N, N
5. N, 2, 3
6. N, 2, N
7. N, N, 3

Im allgemeinen Fall sind also 2^k-1 Punktfragen auf GF_S für diesen Teil der Überprüfung erforderlich. Für jede dieser sieben Konfigurationsbelegungen können dabei n_i S-Tupel gefunden werden.

RESTRICTED

Es ist zu überprüfen, ob nach Löschung von t_V die V-Relation für die Menge T_S aller gefundenen S-Tupel Väter im Sinne der erweiterten Beziehung besitzt. Wird für ein $t_S \in T_S$ kein weiterer Vater gefunden, so wirkt die RESTRICTED-Option und V2 wird zurückgesetzt.

Bei der Vater-Prüfung überdecken bestimmte Konfigurationsbelegungen andere, wenn sie in einer oder mehr Positionen einen Wert besitzen, wo die anderen N haben, und in den restlichen Positionen übereinstimmen. Ihre Überprüfung macht dann andere überflüssig (z.B. entfällt bei Überprüfung von K2 oder K3 die von K4).

Gibt es Tupel in Konfigurationsbelegung K1 (1, 2, 3), so kann die Zurückweisung automatisch erfolgen. Deshalb wird man zuerst diese Teilprüfung mit einer Punktfrage auf GF_S durchführen. Verläuft diese Teilprüfung negativ, wird mit den übrigen Konfigurationsbelegungen fortgefahren.

Die Überprüfung von K2 erfordert eine Bereichsfrage an GF_V. Ist kein Vater vorhanden, wird die Operation zurückgesetzt. Wird für K2 ein Vater gefunden, ist die Vater-Bedingung auch für K4 und K6 erfüllt und die Prüfung kann mit der nächsten unerledigten Konfigurationsbelegung fortgesetzt werden. Der gleiche Prüfschritt wird für K3 angewendet. Falls kein Vater dafür existiert, wird die Operation zurückgesetzt, sonst hat die Prüfung auch K7 implizit erledigt, und es wird K5 in Angriff genommen. Wenn ein Vater festgestellt wird, war die Operation erfolgreich, ansonsten muß sie abgebrochen werden.

Als grober Prüfungsaufwand ergeben sich folglich maximal

- 2^k-1 Punktfragen an GF_S zur Ermittlung der Menge T_S der an der erweiterten Beziehung beteiligten Söhne (Typ-1-Anfragen)
- k Bereichsfragen an GF_V zur Prüfung, ob weitere Väter für die Söhne in T_S existieren (Typ-2-Anfragen).

SET NULL/SET DEFAULT

Bei dieser Option werden nur die Söhne in einer erweiterten Beziehung verändert, die keinen weiteren Vater mehr besitzen. Hierbei ist kein vorzeitiger Abbruch der Operation (wie bei RESTRICTED) vorgesehen. Die Konfigurationsbelegungen können der Reihe nach bestimmt und verarbeitet werden.

Bei den Söhnen nach K1 erfolgt eine unbedingte Änderung. Die nachfolgenden Konfigurationsbelegungen werden nach folgendem Prinzip der Reihe nach behandelt: Wenn sich für die Söhne von K_i, i=-2, ..., 2^k-1, noch mindestens ein Vater findet, dann tue nichts, sonst wende Option SET NULL/DEFAULT an.

Als Prüfaufwand fallen hier an:

- 2^k-1 Punktfragen an GF_S (Typ-1-Anfragen)
- 2^k-2 Bereichsfragen an GF_V (Typ-2-Anfragen).

Der Änderungsaufwand ist hier hochgradig vom Ergebnis dieser Bereichsfragen abhängig.

CASCADE

Alle Söhne, die in einer erweiterten Beziehung keinen weiteren Vater mehr besitzen, werden gelöscht. Die Löschung der Söhne nach K1 wird direkt vorgenommen. Danach werden die Konfigurationsbelegungen der Reihe nach bearbeitet: Wenn sich für die Söhne von K_i, i=2, ..., 2^k-1, noch mindestens ein Vater findet, dann tue nichts, sonst lösche die betreffenden Söhne. Auch hier ist der Wartungsaufwand exponentiell:

- 2^k-1 Punktfragen an GF_S (Typ-1-Anfragen)
- 2^k-2 Bereichsfragen an GF_V (Typ-2-Anfragen).

V3: Ändern eines Tupels in V

Diese Operation ist im wesentlichen als Kombination von V1 und V2 zu verstehen. Ihre Komplexität ist mit der von V2 zu vergleichen, für die wir exponentiellen Aufwand festgestellt haben.

Wir wollen hier beide Operationen und ihre Kosten nicht weiter detaillieren. Die Entwicklung von hinreichend genauen Kostenmodellen würde die Einführung zusätzlicher Parameter und Annahmen erfordern. Die zu erwartenden Ergebnisse dagegen würden uns unserem Untersuchungsziel nicht wesentlich näher bringen.

7. Zusammenfassung

SQL2 erlaubt eine Reihe verschiedener Möglichkeiten, systemkontrollierte referentielle Beziehungen zwischen zwei Relationen zu spezifizieren. Schon die Überwachung der klassischen Referentiellen Integrität erzeugt einen nicht zu vernachlässigenden Zusatzaufwand, selbst wenn die Primär-/Fremdschlüssel-Beziehung nur durch einfache Attribute ausgedrückt wird. Bei zusammengesetzten Attributen bleibt der Zusatzaufwand prinzipiell gleich, die Zugriffspfadunterstützung und die erforderlichen Synchronisationsmaßnahmen werden jedoch komplexer. Die neuen Modellierungsmöglichkeiten, die sich durch die Nutzung der Option MATCH PARTIAL ergeben, gestatten die Darstellung und Kontrolle erweiterter referentieller Beziehungen und spezieller n:m-Beziehungen. Unsere Untersuchung hat gezeigt, daß diese Modellierungsmöglichkeiten und ihre Semantik nicht nur schwer zu verstehen sind, sondern auch exponentiellen Überwachungsaufwand, was die Anzahl von Bereichsfragen angeht, bei bestimmten Aktualisierungsoperationen eines Vater-Tupels erfordern.

B^*-Bäume und ihre Varianten stellen die beste Zugriffspfad-Unterstützung bei der Sicherung der klassischen Referentiellen Integrität dar; das gilt auch bei zusammengesetzten Schlüsseln, wenn dabei nur auf Punktfragen zurückgegriffen werden muß.

Das Grid File ist vom Konzept her prinzipiell geeignet für die Überwachung aller referentiellen Beziehungen über Mehrattributschlüssel. Die konkreten Anforderungen im Mehrbenutzerbetrieb, die beim Grid File durch komplexe Sperrprotokolle zu synchronisieren sind, und die mit k exponentiell wachsende Anzahl von Bereichsfragen für Kontollaufgaben ($O(2^k)$) lassen jedoch ein äußerst ungünstiges Leistungsverhalten erwarten. Häufig werden Algorithmen, die mit Hilfe von Relationen-Scans sequentiell suchen, keine schlechtere Lösung verkörpern.

Schließlich bleibt die Frage nach der praktischen Relevanz der erweiterten referentiellen Beziehungen zu stellen und danach, ob der Gewinn an Modellierungsmächtigkeit den unangemessen hohen Zusatzaufwand für die Sicherung dieser Beziehungen ausgleicht.

8. Literaturverzeichnis

Be75 Bentley, J.L.: Multi-Dimensional Search Trees Used for Associative Searching, in: Comm. ACM, Vol. 18, No. 9, 1975, pp. 509-517.

Co79 Comer, D. The Ubiquitous B-Tree, in: ACM Surveys, Vol. 12, No. 2, 1979, pp. 121-137.

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

EGLT76 Eswaran, K.P., Gray, J.N., Lorie, R.A., Traiger, I.L.: The Notions of Consistency and Predicate Locks in a Database System, in: Comm. ACM, Vol. 19, No. 11, Nov. 1976, pp. 624-633.

FB74 Finkel, R.A., Bentley, J.L.: Quad Trees: A Data Structure for Retrieval on Composite Keys, in: Acta Informatica, Vol. 4, No. 1, 1974, pp. 1-9.

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.

HR83 Härder, T., Reuter, A.: Principles of Transaction-Oriented Database Recovery, in: ACM Com-puting Surveys, Vol. 15, No. 4, 1983, pp. 287-317.

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

Hä92 Härder, T.: Mehrdimensionale Zugriffspfade in Relationalen Datenbanksystemen, Interner Bericht, Universität Kaiserslautern, Jan. 1992.

NHS84 Nievergelt, J., Hinterberger, H., Sevcik, K.C.: The Grid File: An Adaptable, Symmetric Multikey File Structure, in: ACM TODS, Vol. 9, No. 1, 1984, pp. 38-71.

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

Ro81 Robinson, J. T.: The k-d-B-Tree: a Search Structure for Large Multidimensional Dynamic Indexes, in: Proc. SIGMOD Conference 1981, ACM, New York, pp. 10-18.

Sa86 Salzberg, B.: Grid File Concurrency, in: Information Systems, Vol. 11, No. 3, 1986, pp. 235-244.

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.

SRF87 Sellis, T., Roussopoulos, N. Faloutsos, C.: The R+-Tree: a Dynamic Index for Multi-dimensional Objects, in: Proc. 13th VLDB Conference, Brighton, U.K., August 1987, pp. 507-518..

.