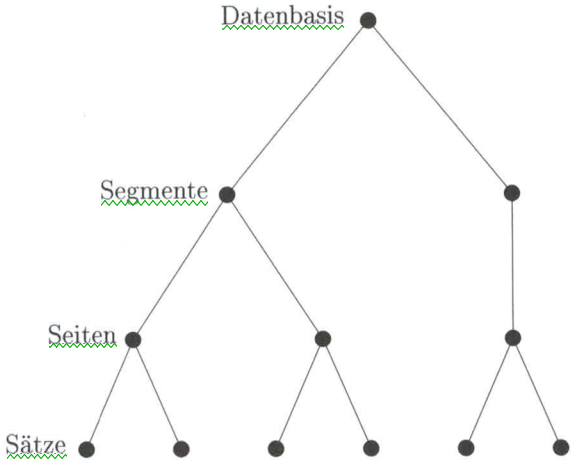

Abb. 11.20: Hierarchische Anordnung der möglichen Sperrgranulate

11.8 Hierarchische Sperrgranulate

Bislang haben wir nur die zwei Sperrmodi S und X betrachtet. Weiterhin waren wir bislang davon ausgegangen, dass alle Sperren auf derselben „Granularität" erworben werden. Mögliche Sperrgranulate sind:

- *Datensätze*: Ein Datensatz (Tupel) ist i.A. die kleinste Sperreinheit, die in einem Datenbanksystem angeboten wird. Transaktionen, die auf sehr viele Datensätze zugreifen, müssen bei dieser Sperrgranularität einen hohen Sperraufwand in Kauf nehmen.

- *Seiten*: In diesem Fall werden durch Vergabe einer Sperre implizit alle auf der Seite gespeicherten Datensätze gesperrt.

- *Segmente*: Ein Segment ist eine logische Einheit von mehreren (i.A. vielen) Seiten. Werden Segmente als Sperreinheit gewählt, wird natürlich die Parallelität drastisch eingeschränkt, da bei einer X-Sperrung implizit alle Seiten innerhalb des betreffenden Segments exklusiv gesperrt werden.

- *Datenbasis*: Dies ist der Extremfall, da dadurch die serielle Abarbeitung aller Änderungstransaktionen erzwungen wird.

In Abbildung 11.20 sind diese hierarchisch miteinander in Beziehung stehenden Sperrgranulate grafisch dargestellt. Bezogen auf Abbildung 11.20 waren wir bislang davon ausgegangen, dass alle Transaktionen ihre Sperren auf derselben Hierarchiestufe – also auf der Ebene der Sätze, Seiten, Segmente oder Datenbasis – anfordern. Überlegen wir uns, welche Auswirkungen die Vermischung der Sperrgranulate hätte: Nehmen wir an, Transaktion T_1 will das „linke" Segment exklusiv sperren. Dann müsste man alle Sperren auf Seitenebene durchsuchen, um zu überprüfen ob nicht irgendeine andere Transaktion eine in dem Segment enthaltene Seite gesperrt hat. Gleichfalls muss man alle Sperren auf der Satz-Ebene durchsuchen, ob nicht ein

Satz, der in einer Seite des Segments steht, von einer anderen Transaktion gesperrt ist. Dieser Suchaufwand ist so immens, dass sich diese einfache Vermischung von Sperrgranulaten verbietet. Andererseits hat die Beschränkung auf nur eine Sperrgranularität für alle Transaktionen auch entscheidende Nachteile:

- Bei zu kleiner Granularität werden Transaktionen mit hohem Datenzugriff stark belastet, da sie viele Sperren anfordern müssen.
- Bei zu großer Granularität wird der Parallelitätsgrad des Systems unnötig eingeschränkt, da implizit zu viele Datenobjekte unnötigerweise gesperrt werden – es werden also Datenobjekte implizit gesperrt, die gar nicht benötigt werden.

Die Lösung des Problems besteht in der Einführung zusätzlicher Sperrmodi, wodurch die flexible Auswahl eines bestimmten Sperrgranulats pro Transaktion ermöglicht wird. Dieses Verfahren wird wegen der flexiblen Wahl der Sperrgranularität in der englischsprachigen Literatur als *multiple-granularity locking* (MGL) bezeichnet.

Die zusätzlichen Sperrmodi bezeichnet man als *Intentionssperren*, da dadurch auf höheren Ebenen der Sperrgranulatshierarchie die Absicht einer weiter unten in der Hierarchie gesetzten Sperre angezeigt wird. Die Sperrmodi sind:

- *NL*: keine Sperrung (no lock),
- *S*: Sperrung durch Leser,
- *X*: Sperrung durch Schreiber,
- *IS* (intention share): Weiter unten in der Hierarchie ist eine Lesesperre (S) beabsichtigt,
- *IX* (intention exclusive): Weiter unten in der Hierarchie ist eine Schreibsperre (X) beabsichtigt.

Die Kompatibilität dieser Sperrmodi zueinander ist in der folgenden Kompatibilitätsmatrix aufgeführt (in der Horizontalen ist die derzeitige Sperre eines Objekts angegeben, in der Vertikalen die – von einer anderen Transaktion – angeforderte Sperre):

	NL	*S*	*X*	*IS*	IX
S	√	√	–	√	–
X	√	–	–	–	–
IS	√	√	–	√	√
IX	√	–	–	√	√

Die Sperrung eines Datenobjekts muss dann so durchgeführt werden, dass erst geeignete Sperren in allen übergeordneten Knoten in der Hierarchie erworben werden. D.h. die Sperrung verläuft „top-down“ und die Freigabe „bottom-up“ nach folgenden Regeln:

1. Bevor ein Knoten mit *S* oder *IS* gesperrt wird, müssen alle Vorgänger in der Hierarchie vom Sperrer (also der Transaktion, die die Sperre anfordert) im *IX*- oder *IS*- Modus gehalten werden.

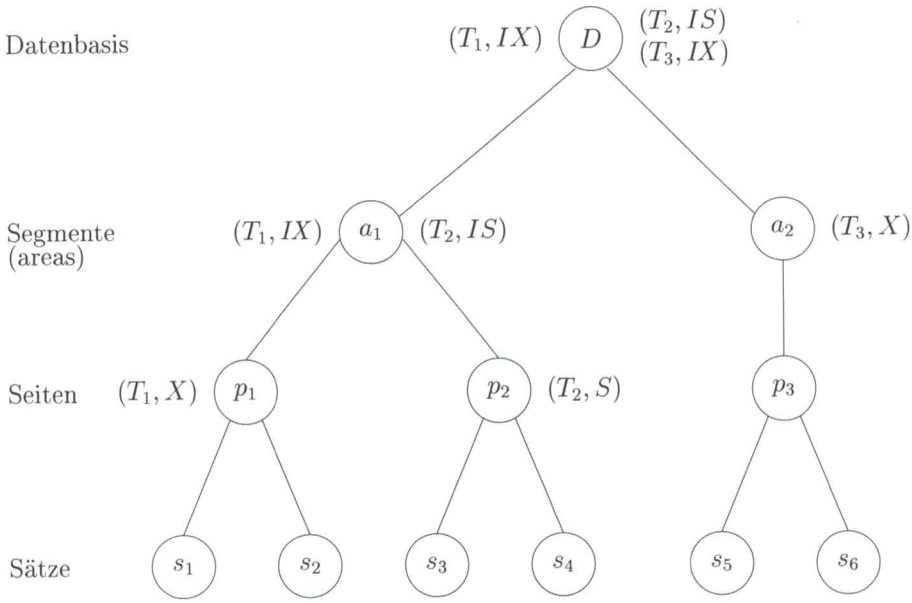

Abb. 11.21: Datenbasis-Hierarchie mit Sperren

2. Bevor ein Knoten mit X oder IX gesperrt wird, müssen alle Vorgänger vom Sperrer im IX-Modus gehalten werden.

3. Die Sperren werden von unten nach oben (bottom up) freigegeben, so dass bei keinem Knoten die Sperre freigegeben wird, wenn die betreffende Transaktion noch Nachfolger dieses Knotens gesperrt hat.

Wenn das strenge 2-Phasen-Sperrprotokoll befolgt wird, werden Sperren natürlich erst am Ende der Transaktion freigegeben. Anhand von Abbildung 11.21 wollen wir das Sperrprotokoll illustrieren. Sperren sind hier mit (T_i,M) bezeichnet, wobei T_i die Transaktion und M den Sperrmodus darstellt. Dazu betrachten wir drei Transaktionen:

- T_1 will die Seite p_1 exklusiv sperren und muss dazu zunächst IX-Sperren auf der Datenbasis D und auf a_1 (den beiden Vorgängern von p_1) besitzen.

- T_2 will die Seite p_2 mit einer S-Sperre belegen, wozu T_2 erst IS-Sperren oder IX-Sperren auf den beiden Vorgänger-Knoten D und a_1 anfordert. Da IS mit den an T_1 vergebenen IX-Sperren kompatibel ist, können diese Sperren gewährt werden.

- T_3 will das Segment a_2 mit X sperren und fordert IX für D an, um danach die X-Sperre auf a_2 zu bekommen. Damit hat T_3 dann alle Objekte unterhalb von a_2 – hier die Seite p_3 mit den Datensätzen s_5 und s_6 – implizit mit X gesperrt.

Die Abbildung 11.21 zeigt den Zustand zu diesem Zeitpunkt – nachdem alle Sperranforderungen der drei Transaktionen erfüllt wurden.

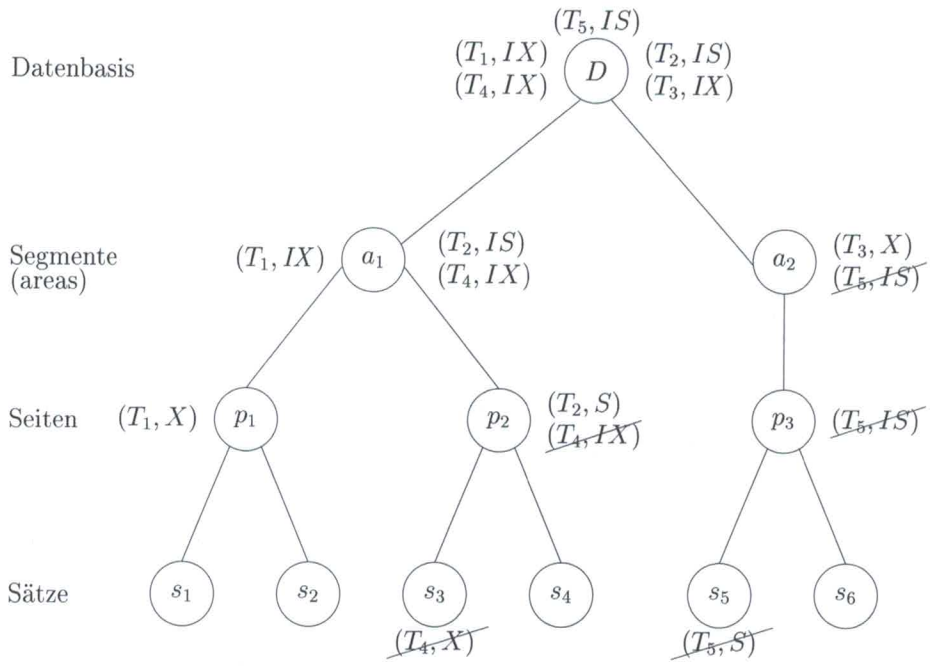

Abb. 11.22: Datenbasis-Hierarchie mit zwei blockierten Transaktionen T_4 und T_5.

Wir wollen nun noch zwei weitere Transaktionen T_4 (Schreiber) und T_5 (Leser) betrachten, deren Sperranforderungen in dem aktuell herrschenden Zustand nicht gewährt werden können.

- T_4 will den Datensatz s_3 exklusiv sperren. Dazu wird T_4 zunächst IX-Sperren für D, a_1 und p_2 – in dieser Reihenfolge – anfordern. Die IX-Sperren für D und a_1 können gewährt werden, da sie mit den dort existierenden Sperren IX und IS kompatibel sind – laut Kompatibilitätsmatrix. Aber die IX-Sperre auf p_2 kann nicht gewährt werden, da IX nicht mit S verträglich ist.
- T_5 will eine S-Sperre auf s_5 erwerben. Dazu wird T_5 IS-Sperren auf D, a_2 und p_3 erwerben müssen. Nur die IS-Sperre auf D ist mit den existierenden Sperren verträglich, wohingegen die auf a_2 benötigte IS-Sperre nicht mit der von T_3 gesetzten X-Sperre kompatibel ist.

Die Abbildung 11.22 zeigt den Zustand nach den oben beschriebenen erfüllten Sperranforderungen. Die noch ausstehenden Sperren sind durch die Durchstreichung gekennzeichnet. Die Transaktionen T_4 und T_5 sind blockiert aber nicht verklemmt und müssen auf die Freigabe der Sperren (T_2, S) auf p_2 bzw. (T_3, X) auf a_2 warten. Erst danach können die beiden Transaktionen T_4 und T_5 mit ihren Sperranforderungen von oben nach unten fortfahren und sukzessive die „durchgestrichenen" Sperren erwerben.

Beim MGL-Sperrverfahren können – obwohl das in diesem Beispiel nicht der Fall ist – durchaus Verklemmungen auftreten (siehe Übungsaufgabe 11.16).

Aus den Beispielen sollte deutlich geworden sein, dass man zu einem gegebenen Knoten in der Datenbasis-Hierarchie alle Sperren verwalten muss. Wenn z.B. in Abbildung 11.21 T_1 die IX-Sperre auf a_1 freigibt, muss der Knoten weiterhin im IS-Modus für T_2 gesperrt bleiben. Deshalb muss man bei einer Sperranforderung im Prinzip die angeforderte Sperre mit allen am Knoten gesetzten Sperren hinsichtlich Kompatibilität überprüfen. Man kann dies aber beschleunigen, indem jedem Knoten ein Gruppenmodus zugewiesen wird. Dazu werden die zueinander kompatiblen Sperren geordnet. Es gilt:

$$\begin{aligned} S &> IS \\ IX &> IS \end{aligned}$$

Alle anderen Sperrmodi können laut Kompatibilitätsmatrix nicht gleichzeitig an demselben Knoten gehalten werden – und brauchen demnach auch nicht geordnet zu werden. Der Gruppenmodus stellt dann die größte (d.h. schärfste) am Knoten gehaltene Sperre dar, und neu eintreffende Sperranforderungen brauchen nur gegen diesen Gruppenmodus auf Verträglichkeit überprüft zu werden.

In der Literatur wurde für das MGL-Sperrverfahren noch ein zusätzlicher Sperrmodus SIX vorgeschlagen, der einen Knoten im S-Modus und gleichzeitig im IX-Modus sperrt. Dieser Modus ist vorteilhaft für Transaktionen die einen Unterbaum der Hierarchie vollständig (oder zumindest zu großen Teilen) lesen, aber nur wenige Daten in diesem Unterbaum modifizieren. Der Sperrmodus SIX erlaubt parallel arbeitenden Transaktionen den Sperrmodus IS, so dass diese Transaktionen gleichzeitig die Daten lesen können, die von der „SIX-Transaktion" nicht modifiziert werden. Die Erweiterung des MGL-Sperrverfahren um diesen Sperrmodus ist Gegenstand der Übungsaufgabe 11.17.

Zusammenfassend erlaubt das MGL-Sperrverfahren den Transaktionen mit geringem Datenaufkommen auf niedriger Hierarchieebene – also in kleiner Granularität – zu sperren, um dadurch die Parallelität zu erhöhen. Transaktionen mit großem Datenvolumen erwerben ihre Sperren auf entsprechend höherer Hierarchieebene – also in größerer Granularität –, um dadurch den Sperraufwand zu reduzieren. Bei einigen Systemen wird automatisch von einer niedrigen Granularität auf die nächst-höhere Granularität umgeschaltet, sobald eine bestimmte Anzahl von Sperren in der kleineren Granularität erworben wurde. Diesen Vorgang nennt man im Englischen „lock escalation".

11.9 Einfüge- und Löschoperationen, Phantome

Es ist klar, dass man auch Einfüge- und Löschoperationen in die Mehrbenutzersynchronisation einbeziehen muss. Die naheliegende Methode besteht in folgendem Vorgehen:

- Vor dem Löschen eines Objekts muss die Transaktion eine X-Sperre für dieses Objekt erwerben. Man beachte aber, dass eine andere TA, die für dieses Objekt ebenfalls eine Sperre erwerben will, diese nicht mehr erhalten kann, falls die Löschtransaktion erfolgreich (mit **commit**) abschließt.