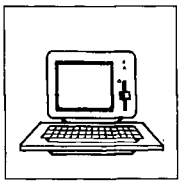
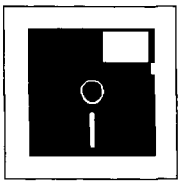




Die „inneren Werte" netzwerkfähiger (Rechtsanwalt-)Software

Dieter Schedel

„Oberfläch(lichkeit)en"

In einer Zeit, in der „Oberfläch(lichkeit)en" wie der Betriebssystemaufsatz Windows 3.1 immer mehr (kommerzielle) Bedeutung gewinnen, scheinen die „inneren Werte" einer Software immer weniger hinterfragt zu werden. Indessen sind es die „motorischen" Strukturen eines Programmes, die hinter einer grafischen oder zeichenorientierten Oberfläche die Qualität der Verarbeitung eingegebener Daten bestimmen. Während die Qualitätsmerkmale „Programmgeschwindigkeit" bzw. „Programmkompaktheit" – sie werden freilich bereits bei Verwendung grafischer Oberflächen in Mitleidenschaft gezogen – durch den Einsatz einer leistungsfähige(re)n Hardware weitgehend ausgeglichen werden können, hat der Programmierer für eine gute Logistik des Programmablaufs und eine angemessene Datensicherheit zu sorgen. Ausreichende Datensicherheit fordert insbesondere in einer Mehrplatz- bzw. Netzwerkumgebung „unsichtbare" Schutzmechanismen, die ein ungestörtes Arbeiten erlauben und verhängnisvolle, „unerklärliche" Programmabstürze nach Möglichkeit unterbinden. Diverse Testläufe, die im übrigen von dem Verfasser mit recht bekannten Programmen durchgeführt worden sind, haben ergeben, daß Hersteller dazu neigen, an dem unsichtbaren Anwenderschutz zu Gunsten sichtbarer Verkaufskriterien zu sparen; relativ einfach erzeugbare, jedoch für den Anwender nur schwer oder überhaupt nicht nachvollziehbare Programmabstürze im Netz, legen hierüber Zeugnis ab. Warum der Einsatz eines flexiblen Netzwerkmanagers – mithin eine Investition in einen „unsichtbaren" Programmteil – für seriöse, netzwerkfähige Software selbstverständlich sein sollte, wird im folgenden am Beispiel eines dezentral arbeitenden Netzwerkes – wie z. B. NOVELL – dargelegt.

Sperrfunktionen als Schutzmechanismen im Netz ...

Netzwerksysteme: sensiblere Strukturen als Einplatzsysteme

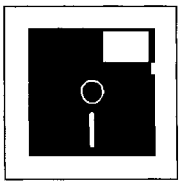
Netzwerksysteme verkörpern weitaus sensiblere Strukturen der Datenverarbeitung als Einplatzsysteme; bei Ausnutzung der heutigen, programmiertechnischen Möglichkeiten vermögen sie jedoch sehr robust und zuverlässig zu arbeiten. Im Gegensatz zur Einplatzsoftware müssen sich netzwerkfähige Programme – und hiermit sind im Gegensatz zu lediglich netzwerktauglicher Software in erster Linie „echte" Netzwerkprogramme gemeint – selbst um die Koordination des gemeinschaftlichen Datenzugriffs mehrerer Arbeitsstationen auf den Datenbestand des Zentralcomputers („Server") kümmern. Zwar bietet ein Netzwerkbetriebssystem wie NOVELL diverse, problemlos extern installierbare Schutzmechanismen zur Vermeidung von Zugriffskollisionen an, jedoch handelt es sich hier nur um einen „Grobschutz", der lediglich bis auf Datei-Ebene herunterreicht. Anwender können hier z. B. Verzeichnisse oder Dateien zu ihrer ausschließlichen Verwendung – sei es für Lese- und/oder Schreibzugriffe – reservieren lassen („directory-/file-locking"). Differenzierte Schutzmechanismen sind indessen nur innerhalb eines Programmes – nämlich durch den sachgerechten Aufruf betriebssystemspezifischer Sperr- bzw. „Lock(ing)"-Funktionen – vorzusehen; der feinstrukturierte, nach dem programminternen Umfeld ausgerichtete Einsatz dieser Funktionen entscheidet über die Qualität „echter" netzwerkfähiger Programme.

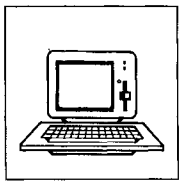
Netzwerktaugliche und „echte" netzwerkfähige Software ...

„file-locking" und „record locking"

Während netzwerktaugliche Programme in der Regel durch Schaffung der einfachen Voraussetzung einer Dateisperre („file-locking") sich selbst, andere aufgerufene Programme oder Daten für die gleichzeitige Benutzung durch andere Arbeitsstationen vorübergehend sperren, arbeiten die Sperrfunktionen netzwerkfähiger Programme sachgerecht auch auf Datensatzebene („record-locking"). Dies bedeutet, daß z. B. eine netzwerkfähige Stammdatenverwaltung auch einzelne gerade bearbeitete Datensätze von einem störenden Zugriff anderer Arbeitsstationen abschotten kann, ohne den gesamten Datenbestand per „file-locking" für andere Arbeitsstationen zu blockieren. Zudem unterstützen netzwerktaugliche Programme in der Regel keine Varianten der Sperrfunktion, während netzwerkfähige Soft-

RA Dr. Dieter Schedel ist Mitglied des RAMANDATA-Entwicklungsteams in Würzburg.





ware ihre Sperren sachgerecht nach dem Programmumfeld einsetzt. Der Programmierer netzwerkfähiger Software muß sich stets Gedanken gemacht haben, ob beim Öffnen einer Datei und – eine Ebene tiefer – beim Zugriff auf einen Datensatz exklusive Schreib-, Lese- oder Mischrechte („read-/write-locks") im Hinblick auf Folgezugriffe anderer Arbeitsstationen vergeben werden müssen. Netzwerktaugliche Software dagegen sperrt häufig unerbittlich eine ganze Datei exklusiv gegen Schreib- und Lesezugriffe, so daß andere Arbeitsstationen an jeglicher Arbeit mit den „gemeinsamen" Daten gehindert werden. Ein Bildschirmhinweis auf die Zugriffssperre bei aufrufbemühten Arbeitsstationen soll für manche programmgestreßte Sekretärin schon als Ankündigung einer ausgedehnten Kaffeepause verstanden werden. Während netzwerktaugliche Software im Zuge einer „voll blockierenden Radikalsperre" ungehemmt Schreib- und Lesezugriffe ausführen kann, muß sich netzwerkfähige Software zur Erhaltung bestmöglicher Netzwerkflexibilität um feinere Ablaufmechanismen bemühen; schließlich soll hier nur ein unbedingt erforderliches Minimum des zu bearbeitenden Datenbestandes gesperrt und nicht – wie bei der nur „tauglichen" Software – zu den Daten Zugang suchende Programme gestoppt werden. Zwar sind Datensatzsperren für (gemeinsame) Lesezugriffe – hier werden in der Regel nur Schreibzugriffe anderer Arbeitsstationen zurückgewiesen – relativ problemlos zu handhaben; Schreibzugriffe in schnellen Dateiverwaltungssystemen, die indexsequentiell arbeiten („BTree-Systeme/Bayer-Bäume"), erfordern zumeist jedoch weitere Vorkehrungen. Hier muß nicht nur der Zugriff auf Index- und Datendateien harmonisiert werden, sondern es sind auch Dateizustandsmeldungen an zentraler Stelle (i. d. R. im sog. Null-Record der Datendatei) zum allseitigen Zugriff für die Arbeitsstationen zu verwalten. Mithin ergibt sich bei solchen Systemen eine zwingende Notwendigkeit, für Schreibzugriffe (Veränderung/Hinzufügung/Löschung von Datensätzen) nicht nur eine Sperre auf die zu verändernden Datenbereiche zu legen, sondern über den kurzen Zeitraum der Veränderung die gesamte Datei zu sperren, damit keine Fremdzugriffe eine aktualisierte Zustandseintragung (z. B. Datensatzzahl) verfälschen.

Grundsätze für den Einsatz der Sperrfunktionen ...

Prinzip der geschlossenen Sperrzyklen

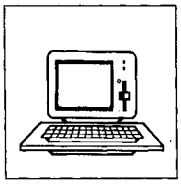
Ein wichtiger Grundsatz bei der Programmierung netzwerkfähiger Programme ist das Prinzip der geschlossenen Sperrzyklen, d. h., daß während der möglichst kurzen Aktivierung einer Sperrfunktion keine Tastendruckabfrage erfolgen sollte, damit keine Behinderung anderer Arbeitsstationen eintritt. Es ergibt sich ansonsten ein ähnlicher Effekt wie beim Einsatz netzwerktauglicher Programme: Eine andere Arbeitsstation kommt – während das aufrufende System im Sperrzustand auf eine Tastatureingabe wartet – an Dateien oder Dateibereiche nicht heran, was besonders bei den zuvor beschriebenen Schreibzugriffen auf indexsequentielle Dateien sehr störend wirkt; wie bereits ausgeführt wird hier kurzzeitig eine Sperre der gesamten Datei erforderlich, die ihrerseits beim Vorhandensein einer jedweden Vorsperre (Datei- oder Datensatzsperre) vom System nicht zugelassen werden darf. In der Programmierpraxis wird indessen gerade die heikle Problematik der Absperrung von Schreibzugriffen häufig unterschätzt. Nachdem Schreibzugriffe grundsätzlich nur zuzulassen sind, wenn keinerlei Datei-Vorsperren existieren und das „Einrasten" der Sperre für einen Schreibzugriff weder das Lesen noch Schreiben anderer Arbeitsstationen in die betroffene Datei zulassen darf, muß ein Programm Sperren für Schreibzugriffe so kurz wie irgendmöglich halten. Tastaturabfragen innerhalb solcher Sperrzyklen kann man – wegen der vom Anwender abhängigen Reaktionszeit – nur als groben Programmierfehler ansehen.

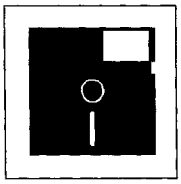
Programmierkonsequenzen

Werden die vorbezeichneten Anforderungen von einem Programmierer netzwerkfähiger Programme verinnerlicht, so hat er bei einfachen Dateiverwaltungssystemen sicherzustellen, daß etwa die häufig vorkommenden Editiervorgänge, d. h. die Eingabe und Veränderung von Daten in Eingabemasken nicht von einer Datei- oder Datensatzsperre begleitet werden. Vielmehr ist bei der Datenveränderung der zu verändernde Satz nach dem Setzen einer Schreibsperre (externes Schreibverbot/externe Lesezulassung) einzulesen, die Sperre aufzuheben und alsdann der Satz zu editieren bzw. die Maske auszufüllen; wird der Datensatz (zurück-)geschrieben, so ist für den kurzen Zeitraum der (Rück-)Speicherung – zumindest in zeitgemäßen „BTree"-Systemen – eine Schreib- und Lesesperre (externes Schreib- und Leseverbot) auf den gesamten Datenbestand (Datei) zu setzen, damit die zentrale Zustandseintragung konsistent bleibt.

Kompliziert: Verwaltung „verzahnter" Datensatzbestände

Komplizierter wird die ganze Angelegenheit, wenn ein indexsequentielles, integriertes Dateiverwaltungssystem (Tabellenkonzept) eine Mehrheit ineinander verzahnter Datensatzbestände zu verwalten hat. Man stelle sich nur vor, es soll eine Mandantenkontobuchung





durchgeführt werden; ein integriertes System wird hieran vernünftigerweise die Stammdaten-, Mandanten- und Sachkontenverwaltung sowie – gegebenenfalls optional – die Forderungskontenverwaltung – beteiligen. Eine zeitgemäße Eingabemaske für die Mandantenkontobuchungen wird während des Maskendurchlaufs etwa den aktuellen Stand angewählter Mandantenkonten, aber auch eines anzusprechenden Finanzkontos vor und nach („preview") der beabsichtigten Buchung anzeigen; der Anwender vermag den „Vorschauzustand" im Zuge der Schlußabfrage („So buchen J/N?") schließlich in eine definitive Buchung zu verwandeln.

Prinzip der Totalblockade?

Programme mit „Sicherheitsbewußtsein" und dem vorbezeichneten, zeitgemäßen Komfort, jedoch ohne besonderes Netzwerkmanagement arbeiten hier nach dem Prinzip der Totalblockade, d. h. daß ein die Maske bearbeitender Anwender durch eine Sperrung aller beteiligter Datenbestände völlig abgeschottet wird. Dies scheint aus Gründen der logischen Datenkonsistenz bei Masken unverzichtbar zu sein, die mit Zwischenberechnungen oder -anzeigen arbeiten, deren Voraussetzungen durch verschiedene Datenbestände geschaffen werden. Würde sich nämlich die Basis der Verrechnung (Vor-Kontostand) vor der definitiven Kontobuchung durch den Einfluß einer anderen Arbeitsstation ändern, so ergäbe sich ein fehlerhafter Kontostand auf dem betreffenden Sammelkonto. Mit einer sachgerechten Schreib-Sperrung der Einlesedaten (Daten können von anderen Arbeitsstationen nur noch gelesen werden) kann für bestimmte Situationen bereits ein gewisses Maß an Flexibilität erreicht werden. Noch effektiver ist allerdings ein Netzwerkmanager, der die Beibehaltung des Grundsatzes ermöglicht, daß Sperren während der Bearbeitung von Eingabemasken (= Tastaturabfragen) nicht gesetzt sein sollten.

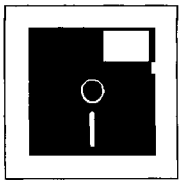
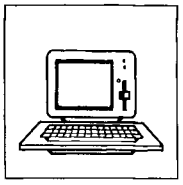
Flexible Netzwerkmanager

Wie kann dies nun ein Netzwerkmanager innerhalb eines netzwerkfähigen Programmes bewirken? Flexible Systeme arbeiten hier mit sog. Parallelsätzen („parallel-records"). Werden Einlese-Datenbestände verarbeitet, so ist der Einlesezustand der jeweiligen Datensätze „parallel" zum tatsächlichen Zustand – er kann durch andere Arbeitsstationen verändert werden – festzuhalten. Wird der über die Eingabemaske erarbeitete Datensatz abgespeichert, so sind nur für den Zeitpunkt der Abspeicherung – ohne Berührung einer Tastendruckabfrage – Sperren zu setzen, die sich auf die tatsächlichen Einlesedaten – z. B. Kontostände – und den schließlich abzuspeichernden Datenbestand – z. B. die Kontobuchung – beziehen. Dabei müssen die tatsächlichen Einlesedaten mit den Vergleichsdaten („parallel-records") auf Übereinstimmung überprüft werden; hierbei genügt ein Schreibschutz gegenüber anderen Arbeitsstationen. Nur bei festgestellter Übereinstimmung der verglichenen Datensätze kann nunmehr eine logisch konsistente Buchung zugelassen werden; sie ergeht freilich – wie bereits ausgeführt – als Schreibzugriff unter Aktivierung eines Lese- und Schreibschutzes gegenüber Alternativstationen. Ein wesentlicher Vorteil dieser Methode ist die Tatsache, daß Anwender, die in Eingabemasken – aus welchen Gründen auch immer – „hängenbleiben" oder probeweise Eingaben vornehmen, keine Systembehinderung – insbesondere für fremde Schreibzugriffe – auslösen. Nachteilig kann wohl im Einzelfall der Umstand wirken, daß der definitive Abspeichervorgang bei logischer Inkonsistenz der Voraussetzungsdaten unterbunden wird; dies läßt sich jedoch als „kleineres Übel" hinnehmen bzw. durch Auffangmaßnahmen wie z. B. einer automatischen, internen Behandlung der „Fehlerstelle" mildern.

Kollisionsprobleme lösen ...

Verträgliche oder unverträgliche Zugriffe?

Ergeben sich im Netz gemeinsame Datenzugriffe zwischen zwei Arbeitsstationen, so stellt sich die Frage, ob es sich um verträgliche oder unverträgliche Zugriffe handelt. Während verträgliche Zugriffe bei gemeinsam zum Lesen berechtigten Stationen problemlos gemeinsame Dateneinlesungen ermöglichen (jeweils Schreibsperre!), muß das System bei gemischten Zugriffen der zweiten (aufrufenden) Station prinzipiell eine unverträgliche Zugriffskollision melden und den Sperrvermerk nebst Datenzugriff unterbinden. Leistungsfähige, netzwerkfähige Software hat Folgezugriffe im Zuge vergeblicher Bemühungen in bestimmten – am besten voreinstellbaren – Abständen selbständig zu wiederholen, bis die Vorsperre aufgehoben und der Zugriff möglich oder ein Wiederholungsabbruchkriterium erreicht ist. Letzteres sollte regelmäßig ein Tastendruck – z. B. auf <ESC> – sein. Häufig erscheint bei Zugriffskollisionen selbst in recht verbreiteter Netzwerksoftware lediglich noch ein Fenster, das die Zugriffskollision und den bereits eingeleiteten Abbruch – gegebenenfalls nach einigen Zugriffsversuchen – mitteilt, worauf eine Taste zum Fortfahren gedrückt werden soll; dies engt die Flexibilität des Systems enorm ein, da nicht nur die Möglichkeit fehlt, im Bedarfsfalle der Sperrenursache nachzugehen und so z. B. laufende Listendrucke bewußt

abzuwarten, sondern es gehen auch bewältigte Eingabearbeiten durch den Abbruch verloren. Besonders die Bildschirmausgabe numerischer Sammellisten (z. B. der Einnahme-Überschußrechnung) kann eine Ausnahme von der Regel rechtfertigen, daß Sperren während einer Tastendruckabfrage nicht gesetzt sein dürfen; hier ist in der Regel zur Ansicht durch den Anwender eine für andere Stationen schreibgesperrte, (bildschirm-)seitenweise Listenausgabe vertretbar, soweit die Ausgabe an eine temporäre Datei – sie wird bei der sofort anschließenden Einlesung exklusiv gesperrt – zu speicheraufwendig erscheint. Numerische Sammellisten versorgen während der Ausgabe schließlich Saldo-Variablen, deren Inhalt(e) am Listenende ausgegeben werden; schon hieraus erklärt sich die Unmöglichkeit, daß die Listenausgabe von Schreibzugriffen „durcheinander gebracht" werden darf.

Sinnvoll abbrechen ...

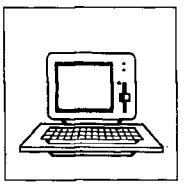
Ein Abbruchkriterium für kollisionäre Zugriffsversuche

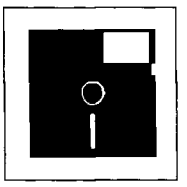
Wie bereits ausgeführt, muß ein Abbruchkriterium für kollisionäre Zugriffsversuche – sei es per Zeitablauf oder Tastendruck – stets vorgesehen sein, zumal ansonsten die Gefahr „endloser" Zugriffsbemühungen besteht („dead-locks"), wenn eine Station ihre Sperre im Störungsfalle nicht mehr verlassen kann, während sich die andere fortwährend um Zugriff bemüht. Dabei darf nicht verkannt werden, daß ein brauchbarer Netzwerkmanager im Zuge von Einlese- und Abspeichervorgängen – letztere können aus Konsistenzgründen auch im Verbund erforderlich werden – nicht nur Kollisionen abzufragen, sondern innerhalb der gesetzten Sperren auch anderweitige Fehlerhaftigkeiten des Systems zu prüfen hat; hierzu zählen bei Schreibzugriffen z. B. die Überwachung der Speicherplatzkapazität auf dem Massenspeicher, die Einhaltung einer maximal zulässigen Datensatzzahl sowie das Auftreten von (anderweitigen) Betriebssystemfehlern. Für ein „sauber" programmiertes System sollten die zuletzt genannten Fehlerhaftigkeiten freilich nur im höchst seltenen Störungsfalle Abbruchursache sein. Wie ein Abbruch schließlich behandelt wird, unterscheidet gleichwohl die Qualität netzwerkfähiger Software. Einfach ausgeführte Programme schließen hier nach Möglichkeit offene Datenbestände und Fenster und steuern unverzüglich den „Programmausgang" an. Anspruchsvollere Lösungen unterscheiden nach der Fehlerqualität. Während die vorbezeichneten „echten Störungen" in der Regel nur den sachgerechten Programmaustritt nach bestmöglicher Erledigung interner Aufräumarbeiten zulassen, ist von dem Programm bei „einfachen" Zugriffskollisionen stets die nächsterreichbare Programmebene anzusteuern; dies wird bei Kollisionen im Zuge der Bearbeitung einer Eingabemaske üblicherweise das Vormenü mit der Bearbeitungsoption sein. Nicht zu vergessen ist schließlich, daß ein guter Netzwerkmanager in der Lage sein muß, sachgerecht auch interne Reduktionen etwa bei Mehrfachschreibzugriffen durchzuführen. So wird z. B. eine integrierte Forderungskontobuchung aus Konsistenzgründen Schreibvorgänge in die Mandanten- und Finanzkontenbuchhaltung innerhalb eines Sperrzyklus erfordern. Wird hier ein Schreibvorgang aus Störungs- oder Kollisionsgründen zurückgewiesen, so hat der Netzwerkmanager diesem Vorgang vorausgegangene Schreibvorgänge zurückzuführen; eine Wiederherstellung des Vorzustandes – im Wege einer Datenlöschung oder -veränderung – muß freilich innerhalb der aktuellen Sperre erfolgen, was die Protokollierung jeweils erfolgter Schreibvorgänge durch den Netzwerkmanager voraussetzt.

Browserprobleme lösen ...

„Statische" Browseranzeige vs. dynamische Zustandsveränderungen

Eine besondere Herausforderung für Programmierer netzwerkfähiger Programme ist der zeitgemäße Einsatz netzwerkfähiger Datensatzbrowser. Ein solcher Browser ermöglicht dem Anwender die Datenselektion über eine Sichtbalkensteuerung innerhalb einer Liste mehrerer (zeilenweise) eingeblendeter Datensätze. In einer Netzwerkumgebung stellt sich das Problem, daß eine Arbeitsstation, die gerade eine Browseransicht – etwa zum Blättern bzw. „Rollen" per Pfeiltastensteuerung – anbietet, den Ausschnitt einer Datensatzliste „statisch" anzeigt. Ändert oder löscht eine andere Arbeitsstation einen Datensatz dieses Ausschnittes auf dem Zentralcomputer („Server"), so zeigt das System einen z.T. tatsächlich nicht mehr existenten Datenbestand; hierbei erscheint neben der fehlerhaften „Optik" auch die Tatsache störend, daß die Arbeitsstation versuchen könnte, auf einen tatsächlich nicht mehr existenten Datensatz zuzugreifen. Schließlich ordnet eine Datensatzbrowser-Liste regelmäßig die angezeigten Datensätze nach einem vorgegebenen Schlüssel; fügt eine fremde Arbeitsstation einen „in den Bildschirmausschnitt der anzeigenden Station passenden" Datensatz hinzu, so erfolgt – ohne besondere Vorkehrungen – keine Anzeige des aktuellen





Zustandes innerhalb dieses Bildschirmausschnittes. Was kann ein Netzwerkmanager hier tun? Während unflexible Systeme einen (bildschirm-)aktualisierenden Zugriff über den zentralen Computer auf den Vorgang der Listen- bzw. Balkenbewegung beschränkt, arbeitet ein Netzwerkmanager sachgerecht in einer Browserumgebung mit „eigenständiger" Serverkontrolle (sog. „Auto-Refreshing"). Stellt der Manager eine kurze Pause seit der letzten Tastenbetätigung fest – schließlich soll kein laufender Eingabeprozeß unterbrochen werden –, so prüft er auf dem Server in regelmäßigen – voreinstellbaren – Zeitabständen nach, ob sich der Bildschirmausschnitt der aufrufenden Station auf dem aktuellen Stand befindet. Ist dies nicht der Fall, so erfolgt per „automatischem" Serverzugriff sofort die Aktualisierung. Wie auch beim Rollen einer Browserdarstellung wird hierbei lediglich eine vom Betriebssystem sehr schnell verarbeitete Schreibsperre (= reiner Lesezugriff) gesetzt.

Die sinnvolle Verwendung einfacher Dateisperren ...

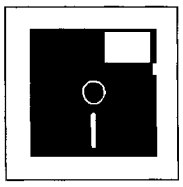
„record-read-locking"

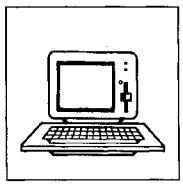
Ein sehr allgemeiner Grundsatz der Netzwerkprogrammierung zum Schutze sensibler, gemeinsam genutzter Datenbestände ist (für jede Arbeitsstation) die Maxime, Dateien so kurz wie nötig geöffnet zu halten, da geschlossene Dateien grundsätzlich keiner Inkonsistenzgefahr ausgesetzt sind. Allerdings benötigen die aufwendigen Betriebssystemfunktionen für das Öffnen und Schließen von Dateien relativ viel Zeit, so daß hintereinander folgende Mehrfachzugriffe auf Daten einen längerfristigen Öffnungszustand notwendig machen. So können moderne Programme, die Datensätze per Sichtbalkensteuerung in einer zeilenweise aufgebauten Liste verwalten („Browser"), nicht bei jeder Balkenbewegung eine Datei öffnen und wieder schließen; hier muß – wie bereits ausgeführt – eine jede Balken- bzw. Hintergrundbewegung im Netz durch eine Datensatzsperre („record-read-locking") gesichert sein, die während der Bewegung Lesezugriffe, jedoch keine Schreibzugriffe anderer Arbeitsstationen zuläßt. Der ausschließliche Einsatz einer Dateisperre genügt mithin lediglich bei einer internen Dateneinlesung, die ein sofortiges Schließen der Datei nach der Einlesung zuläßt. Soll etwa eine Datei mit BRAGO-Werten während eines Berechnungsvorganges zur Auslesung geöffnet werden, so wird sich hier eine einfache Schreibsperre auf Dateiebene anbieten, wenn die Datei geöffnet, ausgelesen und sofort wieder geschlossen werden kann; soll die gleiche Datei durch das Programm verändert werden, so muß die Software für die Zeit des Schreibzugriffs jegliche Lese- und Schreibzugriffe anderer Arbeitsstationen verbieten, indem es auf Dateiebene eine entsprechende Sperre setzt.

Spezielle Probleme der Textverarbeitung ...

Problematisch: Parallele Text-Aktualisierungen

Der Hauptanwendungsfall einer Sperrfunktion ist das sachgerechten Sperren von Daten, die durch ein datenverwaltendes Programm gelesen, geschrieben bzw. verändert werden. Allerdings müssen auch Texte in textverarbeitender Netzwerk-Software entsprechend geschützt werden. Schreibvorgänge werden bezüglich des bezeichneten Textes, wie bereits dargelegt, durch eine Schreib- und Lesesperre geschützt. Problematisch sind bei der Textbearbeitung jedoch parallele Aktualisierungen: Sekretärin A ruft Text T auf und bearbeitet ihn, worauf Sekretärin B den gleichen Text zwecks Bearbeitung einliest; speichert Sekretärin A den veränderten (gleichnamigen) Text ab, so werden ihre Veränderungsbemühungen durch einen folgenden, von Sekretärin B ausgelösten Schreibvorgang zunichte gemacht. Hier sind verschiedene Lösungswege denkbar. Zum einen kann eine aufwendige Listenverwaltung im Textverarbeitungsprogramm Buch führen über die Texteinladung der verschiedenen Stationen, um alsdann Warnhinweise bei der stationsübergreifenden Einlesung des gleichen Textes auszugeben. Zum anderen gibt es die durchaus akzeptable Möglichkeit, das Betriebssystem an einem wirksamen Schutzmechanismus zu beteiligen. Liest ein Anwender einen Text ein, so wird gleichzeitig vom Betriebssystem der Zeitpunkt des letzten Schreibvorganges in die Datei übernommen und gespeichert. Soll eine Veränderung des Textes zurückgeschrieben werden, so greift das Programm auf den gleichnamigen (alten) Text zu und vergleicht dessen aktuelles Datum (des letzten Schreibzugriffs) mit dem zwischengespeicherten Datum. Eine Abweichung signalisiert einen „Zwischenbearbeiter", so daß auf eine entsprechende Warnanzeige hin die Abspeicherung unter einem anderen Dateinamen ratsam erscheint.





„Echte" Netzwerksoftware: Nicht „nebenbei" zu erledigen

Fazit ...

Vorliegend wurde der Versuch unternommen, konzeptionelle Grundsätze aus dem Bereich „Netzwerksicherheit" zu erörtern, deren Außerachtlassung schon manchen „abgestürzten" Anwender leider mehr an der EDV als an seinem Programm zweifeln ließ; zusätzlich sollte ein grober Überblick über die Funktionalität zeitgemäßer, netzwerkfähiger Software gegeben werden. Nicht eingegangen werden konnte daher auf die gesamte Palette der programmierbaren Schutzmechanismen, die ein Betriebssystem wie NOVELL „seinen" Programmen anbietet. Es dürfte allerdings deutlich geworden sein, daß „echte" Netzwerksoftware nicht ohne weiteres von einem Wochenendprogrammierer erstellt werden kann, der durchaus ein Gebührenprogramm zustande zu bringen vermag; dies darf angesichts der Qualitätsunterschiede angebotener Anwaltssoftware durchaus einmal ausgesprochen werden. Selbst „Blackbox"-Dateiverwaltungssysteme, die gerne für wirtschaftliche Programmprojekte herangezogen werden, bieten oft von ihren Funktionen her bereits nicht die Möglichkeit, differenzierte Schutzreaktionen im Netz vorzusehen.

Konsequenzen für den Test von Anwaltssoftware unter „Voll-Last"

Schließlich sei dem Verfasser die Feststellung erlaubt, daß auch Anwaltssoftware in softwaretechnischer Hinsicht eingehender getestet werden sollte, wobei die Überprüfung der Netzwerksicherheit und Arbeitsqualität eines Programmes unter praxistypischer „Voll-Last" erfolgen müßte; nur der gemeinsame (kollisionäre) Datenzugriff im Netz auf eine praxisgerechte Anzahl von Datensätzen über eine Referenzhardware zeigt, was eine *Software* tatsächlich leistet.

jurpc.zip – jurpc.zip – jurpc.zip – jurpc.zip – jurpc.zip – jurpc.zip – jurpc.zip

Anmerkung zu KG Berlin, Urteil vom 30. März 1993 (5 U 7655/92)

(jur-pc 11/93, S. 2365, in diesem Heft)

Da der „Clementinen"-Fall des BGH (25.6.1992, I ZR 136/90, GRUR 1992, 858–860) vom KG als Präjudiz diskutiert wird, rechtfertigt sich die Frage, wie man aus einer für Clementinen formulierten Regel etwas für Notebooks gewinnen kann.

Im Clementinen-Fall fehlte es an den beworbenen Clementinen. Ausgesprochen wurde deswegen zunächst ein auf Lebensmittel bezogenes Werbeverbot (für den Fall mangelnden Vorrats). Der BGH hat das als zu weitgehend angesehen und den Unterlassungsanspruch auf den Bereich von Obst und Gemüse begrenzt:

„In der Rechtsprechung des Bundesgerichtshofs ist anerkannt, daß im Interesse eines hinreichenden Rechtsschutzes gewisse Verallgemeinerungen zulässig sind, sofern auch in dieser Form das Charakteristische der konkreten Verletzungsform zum Ausdruck kommt (...). Dem liegt die Erwägung zugrunde, daß eine in bestimmter Form begangene Verletzungshandlung nicht nur die Wiederholung der genau identischen Verletzungsform vermuten läßt, sondern auch eine Vermutung für die Begehung zwar leicht abgewandelter, aber in ihrem Kern gleicher Handlungen begründet, wobei allerdings eine den Bestimmtheitsanforderungen genügende Grundlage für die Vollstreckung auch bei abweichenden Handlungsformen vorliegen muß.

Diesen Anforderungen wird der vom Berufungsgericht bestätigte Verbotsausspruch des Landgerichts nicht gerecht. Zu Recht beanstandet die Revision, daß das Verbot zu weit gefaßt ist. Aus dem Nichtvorhandensein der Clementinen, lediglich einem kleinen Teil des mit der angegriffenen Anzeige unter 'Obst und Gemüse' beworbenen Gesamtangebots der Beklagten, kann nicht die Vermutung abgeleitet werden, daß die Beklagte hinsichtlich sämtlicher anderer Waren und in allen Warenbereichen und Abteilungen ihres Unternehmens mangelhaft disponieren und insoweit wettbewerbswidrig werben werde. Das begehrte Verbot war deshalb auf Angebote der Produktgruppe Obst und Gemüse zu beschränken."

Heißt das, daß man bei fehlendem Notebook-Vorrat im Unterlassungsanspruch auf EDV-Waren generalisieren darf oder nicht? Die Frage stellen, heißt die Unklarheit der Entscheidungskriterien zu bemerken. Parallelisiert man „Clementinen -> Obst und Gemüse -> Lebensmittel" mit „Notebooks -> EDV-Waren", so kommt man eher zu der Einschätzung, daß ein Unterlassungsanspruch allenfalls für eine Teilmenge der EDV-Waren gegeben sein könnte. Das KG entwickelt im Kontrast dazu eine Homogenitäts-/Inhomogenitätstheorie bezogen auf Warenangebote, die so der BGH-Entscheidung nicht zu entnehmen ist: Bei inhomogenem Warenangebot („Clementinen-Fall": Frischwaren vs. sonstige Lebensmittel) kommt eine Generalisierung nur innerhalb des vom Vorratsmangel betroffenen Bereichs in Frage, bei homogenem Warenangebot ist eine stärkere Generalisierung erlaubt. Nur: Welches Kriterium bestimmt über die Homogenität eines Warenangebots? Wir wissen jetzt, wie das Kammergericht es im Falle von Notebooks und EDV-Waren diesbezüglich hält – die genaue Methode kennen wir indessen nicht.

Maximilian Herberger