Informatikpraktikum
188.430

Zweistelligen Jahreszahlen und das Jahr 20**00**

Claus Reichel*
9300654

17. Dezember 1996

*http://stud1.tuwien.ac.at/~e9300654

1 Einleitung

Im Rahmen dieser Lehrveranstaltung beschreibt jeder Teilnehmer die Lösung des „Jahr 2000 Problems" für ein von ihm frei gewähltes Unternehmen. Dabei steht die technische Lösung des Problems im Vordergrund.

Ich habe mich für das Unternehmen RACON (Raiffeisen Automation Communication Online Network) entschieden. Auf diesem Wege möchte ich mich bei Herrn DI Franz Biber für seine Unterstützung bei dieser Arbeit bedanken. Bevor ich jedoch mit den Fakten über das „Jahr 2000 Problem" beginne, werde ich noch einen kurzen Überblick über das Unternehmen geben.

RACON ist eine Gesellschaft im Raiffeisenbankensektor und führt für diesen die Softwareentwicklung und -wartung durch. Das Unternehmen unterhält vier Standorte Wien, Linz, Salzburg und Innsbruck, die ursprünglich einzelne eigenständige Abteilungen auf Landesverbandsebene darstellten und nun in einer Ges.m.b.H. vereint wurden. Die im folgenden erwähnten Daten beziehen sich nur auf den Standort Wien.

Der aktuelle Mitarbeiterstand des Unternehmens beträgt cirka 120 Mitarbeiter. Ein kurzer Überblick über die von dem Unternehmen eingesetzte Hardware am Standort des „Raiffeisenhauses Wien" (RHW). Die Daten in den Klammern beziehen sich auf den Entwicklungs- und Backup-Standort im „Zentrum am Stadtpark" (ZAS):

RHW	ZAS	Hardware (Stand: Anfang 1996)
1	(1)	Multiprozessor Comparex 99/952 (99/821) - 280 (118) MIPS, 1,75 (1) GB Speicher
1	(1)	Magnetbandsteuerungen
2	(2)	Magnetbandstationen - 1.600/6.250 BPI
1	(1)	Magnetbandkassettensteuerungen
2	(2)	Magnetbandkassettenstationen
5	(4)	Plattenspeicherdoppelsteuerungsanlagen - aktuelle Kapazität 900 (560) GB
2	(1)	Datenübertragunsvorrechner
1	(1)	OCRB-Drucker - 120.000 Zeilen/Stunde
3	(1)	Laserdrucker - à 1.060.000 Zeilen/Stunde
8	(5)	Bildschrimmehrfachsteuerungsanlagen mit
88	(21)	angeschlossenen Bildschirmarbeitsplätzen
37	(1)	Terminaldrucker

2 Problembeschreibung

2.1 Allgemein

Der Grund für das „Jahr 2000 Problem“, hat seine Wurzeln in den unterschiedlichsten Gründen wie z.B.

- beschränkten Resourcen (Hauptspeicher, Rechenleistung, externe Speichermöglichkeiten, zur Verfügung stehendes Budget)
- dem allgemein üblicher Gebrauch von nur 2-stelligen Jahreszahlen
- der Umsetzung von gedruckten Formularen mit vorgedrucktem Jahrhundert auf Bildschirmeingabemasken

Dabei wurden nur Datumsformate mit 2-stelligen Jahreszahlen eingesetzt. Im EDV-Bereich dürfte aber der erste Punkt den Hauptausschlag für diese Entwicklung gegeben haben.

Alle Fälle, die nun die Grenze der möglichen Bandbreite von 100 Jahren, wie z.B. Verträge mit einer Laufzeit von mehr als 100 Jahren oder Personen, die älter als 100 Jahre sind, überschreiten müssen bereits in den heute eingesetzten Systemen gelöst sein. Somit stellen all diese Fälle kein „neues“ Problem dar.

Als Beispiel für einen solchen Fall möchte ich die österreichnischen Sozialversicherungsträger erwähnen. Da der Bezug von diversen Sozialleistung an das Alter gebunden ist, das Geburtsdatum aber ausschließlich aus einer 10-stelligen Nummer (4-stellige fortlaufende Nummer und dem Geburtsdatum im Format TTMMJJ) abgelesen werden kann, wird dieses Problem mit Hilfe einer Tabelle gelöst, in der alle über 100 Jahre alten Personen eingetragen sind. Aus diesem Beispiel darf man aber nicht schließen, daß die in der Folge erwähnten Probleme bei den Sozialversicherungsträgern nicht vorhanden sind.

2.1.1 Sortierproblem

Wenn die Datumsfelder in einem 6-stelligen Datumsformat, wie z.B. JJMMTT, vorliegen und man einen Sortieralgorithmus darauf anwendet, so werden die Daten von (20)00 und den nachfolgenden Jahren vor den Daten aus dem 20. Jahrhundert, wie z.B. (19)99, zu liegen kommen. Dieser Fehler stellt im Vergleich zu den anderen „Fehlern“, die auf uns zukommen bzw. bereits eingetreten sind ein relativ „harmloses“ Problem dar.

2.1.2 Berechnungssproblem

Warum stellt der „Sprung“ in ein neues Jahrhundert mit zweistelligen Feldern für das Jahr ein Problem dar?
Man muß sich nur das Ergebnis der folgenden Pseudocode-Funktion vorstellen, wenn sie mit zwei Daten aufgerufen wird, und `d2` ist `00`. Annahme: man kann davon ausgehen, daß das ältere Datum immer als erster Parameter und das jüngere als zweiter Parameter übergeben wird.

```
int tage(int d1, int d2)
{

    ...
    erg = d2 - d1;
    ...

};
```

In diesem Fall ergibt sich eine negative Zeitspanne, die den Programmierer auf einen Berechnungsfehler aufmerksam machen sollte.

2.1.3 Vergleichsproblem

Ein nahezu gleiches Problem, wie das Berechnungsproblem ergibt sich, wenn es darum geht zu prüfen, ob ein gewisser Zeitpunkt bereits erreicht bzw. überschritten wurde. Dies könnte man mit einer leicht modifzierten Variante der obigen Funktion erreichen. Die Modifikation beschränkt sich lediglich darauf, das aktuelle Datum als ersten Parameter sowie das zu vergleichende Datum als zweiten Parameter zu übergeben. Wenn nämlich die modifizierte Funktion einen Wert ≤ 0 liefert, dann wurde der Zeitpunkt bereits erreicht bzw. überschritten. Wie man allerdings leicht erkennen kann gilt dies nicht für Daten aus unterschiedlichen Jahrhunderten, in einem solchen Fall würde man einen positiven Wert erhalten, obwohl der Zeitpunkt bereits überschritten wurde.

Diese Fehler treten allerdings „nur“ dann in dieser Form auf, wenn der zweistellige Anteil der Jahreszahlen des älteren Datums größer ist als der des neuen Jahrhunderts. In allen anderen Fällen würde sich „bloß“ ein Irrtum um ein oder mehrere Jahrhunderte ergeben, und diese Fälle müssen wie bereits erwähnt auch heute schon extra behandlet werden.

2.1.4 99/00-Problem

Eingegebene Daten werden vor ihrer Weiterverarbeitung meistens auf ihre Gültigkeit überprüft. Was ereignet sich aber, wenn der Bereich auf dem die Werte für das zweistellige Jahr gültig sind nur wie folgt $[0; 98]$ bzw. $[1; 99]$ definiert ist? Der Benutzer sieht sich einer Situation gegenüber, in der er „spezielle“ Daten (99 bzw. 0) nicht eingeben kann.

Oft wurde 99/00 auch verwendent, um Daten als ∞-gültig zu kennzeichnen. In diesem Fall können zahllose Datenmengen unwiderbringlich verloren gehen, wenn abgelaufene Datensätze automatisch gelöscht werden.

2.1.5 Wochentagsproblem

Der 1. Jänner 1900 war ein Montag, aber der 1. Jänner 2000 wird ein Samstag sein. Deshalb werden alle Programme, die vom Wochentag abhängig sind, und diesen aus einem Datum mit 2-stelliger Jahreszahl ermitteln falsche Werte liefern. Dieser Fall tritt dann ein, wenn der Basistag um 100 Jahre überschritten wird. Aber selbst, wenn als Basistag der Berechnung nicht der 1. Jänner 1900, sondern ein späterer Zeitpunkt gewählt wurde kann ich mir bei einigen Programmen Probleme mit Daten aus dem folgenden Jahrhundert vorstellen, wenn plötzlich das Jahr für den zu berechnenden Tag „vor“ dem Basisjahr liegt. Ich möchte bereits hier einen kleinen Vorgriff auf den folgenden Abschnitt machen, denn auch bei diesem Problem koennte man sich mit einer Art „Fenster“ bei dem der Basistag weitergeschoben wird behilflich sein.

2.1.6 Gregorianischer Kalender

Nebenbei weißt der Gregorianische Kalender noch die Besonderheit von Schalttagen auf. Solch ein zusätzlicher Tag wird eingeführt, wenn das Jahr ohne Rest durch 4 teilbar ist. Aber keine Regel ohne Ausnahme, den dies gilt nur dann wenn es kein Jahr ist, das ohne Rest durch 100 teilbar ist. Aber auch diese Ausnahme gilt nur dann, wenn das Jahr nicht auch noch ohne Rest durch 400 teilbar ist. Und eben dieser Spezialfall tritt im Jahr 2000 ein, auf das alle drei Regeln angewendet werden müssen. Schließlich stellt man fest, daß im Gegensatz zum Jahr 1900, das Jahr 2000 einen Schalttag haben wird. Im Zusammenhang mit dem Wochentagsproblem wird diese Besonderheit des Jahres 2000 die Wochentage ab dem 1. März 2000 um einen zusätzlichen Tag verschieben.

2.2 Unternehmensspezifisch

Für das Unternehmen stellen sich das Sortier-, das Berechnungs- sowie das Vergleichsproblem als Probleme dar. Das 99/00-Problem ergibt sich für das Unternehmen nicht, da ∞-gültige Datensätze nicht mit 31.12.99, sondern mit 99.99.99 gekennzeichnet wurden.

Man könnte sich heute fragen, warum man nicht bereits bei der Programmentwicklung (zahlreiche Routinen wurden bereits vor mehr als einem Jahrzehnt geschrieben) auf dieses Problem aufmerksam geworden ist?
Aber auf diese Frage gibt es wie bereits erwähnt eine Menge an Antworten. Die vorhandene Rechenleistung und der zur Verfügung stehende Hauptspeicher im Unternehmen waren derart begrenzt, sodaß bereits geringfügige Erweiterungen das Laufzeitverhalten derart negativ beeinflußten, sodaß man an allen „Ecken und Enden" versuchte soviel wie möglich einzusparen. Wenn man desweiteren bedenkt, daß pro Konto zwischen 30 und 60 Datumsfelder verwaltet werden, so hätte eine Erweiterung auf 4-stellige Jahreszahlen eine beträchtliche Menge an zusätzlicher Speicherkapazität erfordert. Weiters gab es in Cobol keine Datumsfunktion, die eine vierstellige Jahreszahl geliefert hätten (z.B. der Großteil der Cobolprodukte von IBM ist bereits Jahr-2000 tauglich[1] bzw. die Gesamteproduktpalette wird es spätestens bis zum Jahresende 1996 sein).

Der aktuelle Sourcecodebestand des Unternehmens umfaßt cirka 19.000 Programme (inklusive Unterprogrammen) und hat einen Umfang von 10 Millionen Lines of Code (inklusive Kommentaren, ...).

Programmiersprache	%-Anteil
Assembler	15 %
Cobol	65 %
C/C++	6 %
SQL-Abfragen	11 %
Terminalsprachen	3 %

Wenn man diese Zahlen mit den durchschnittlichen Werten ([2] ca. 60 % Assembler, ca. 30 % Cobol bzw. PL/1) im Bankensektor vergleicht, so fällt der hohe Cobol und relativ geringe Assembler Anteil auf.

[1]The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation, 27^{th} August 1996, IBM, ftp://lscftp.pok.ibm.com/pub/year2000/y2kpaper.txt als ASCII-Text oder y2kpaper.exe (als Self-extracting ZIP-File)

[2]Mag. Robert Kromer - Softlab

3 Diskutierte Technische Lösungen

3.1 Konvertierung von 2-stelligen auf 4-stellige Jahreszahlen

Für diesen Lösungsweg muß sowohl die Datenstruktor, als auch der Programmcode geändert werden. Das heißt, aber auch das man zur einfacheren Verwaltung auch die bereits bestehenden Daten mirgrieren sollte. Es müssen alle Funktionen, Berechnungen und externe Programme die auf die geänderte Datenstruktur zugreifen an die neue Situation der 4-stelligen Jahreszahlen angepaßt werden. Desweiteren müssen alle Programme (Eingabemasken, Datenstrukturen, Programmcode) simultan auf die neue Situation umgestellt werden. Neben allen Nachteilen, die diese Lösung mit sich bringt, ist sie die einzige vollständige und dauerhafte Lösung.

3.2 Kompression

Die Möglichkeit der Kompression bietet sich nur dann für ein Unternehmen, wenn die Daten noch nicht binär gespeichert sind. Diese Möglichkeit ist auch von der eingesetzten Hardware abhängig, da sich die Laufzeit für die Programme. Bei jedem Zugriff auf ein Datumsfeld muß das darin enthaltene Datum komprimiert/dekomprimiert werden.

3.2.1 Binäre Verschlüsselung

Wenn man davon ausgeht, das die Daten bisher als 6-stellige Zeichenketten gespeichert wurden, so hätte man $6 * 8 = 54$ bit für die Speicherung eines Datums zur Verfügung. Selbst wenn man von diesen 54 bits großzügig noch 5 bits für ein Tagesfeld und 4 bits für ein Monatsfeld abzieht bleiben noch 2^{45} Möglichkeiten für das Jahr über.

3.2.2 Hybridlösung

Bei diesem Ansatz wird davon ausgegangen, daß auf „99“ nicht „00“, sondern „A0“ folgt bzw. daß auf das 12. Monat ein 13. Monat folgt. Durch die Komplexität dieser Lösung werden auch enorme Kosten bei der Umstellung verursacht, besonders im Hinblick auf die Folgekosten, da diese Lösung eine „ordentliche“ Umstellung nur hinauszögern. Diese Möglichkeit besteht aber auch nur dann, wenn die Daten alphanumerisch abgelegt wurden.

3.3 Fenstertechnik

Bei dieser Technik ist immer ein „Fenster“ von 100 Jahren offen, mit dessen Hilfe entschieden werden kann, in welches Jahrhundert bestimmte Daten fallen. Die Fälle die über dieses 100jährige Fenster hinreichen können wie bisher gelöst werden. Als Pseudocode könnte man sich dies folgendermaßen veranschaulichen.

```
int jahrhundert(int jj)
{
    if (jj < grenze)
        return (aktuelles_jahrhundert+1);
    else
        return (aktuelles_jahrhundert);
};
```

3.3.1 Fixed-Window Technik

Bei dieser Variante ist immer ein fixes Fenster geöffnet. Folglich wäre der Wert für `grenze` fix definiert, z.B. 50, wo alle Daten, die kleiner als 50 sind als Daten des nächsten Jahrhunderts interpretiert werden.

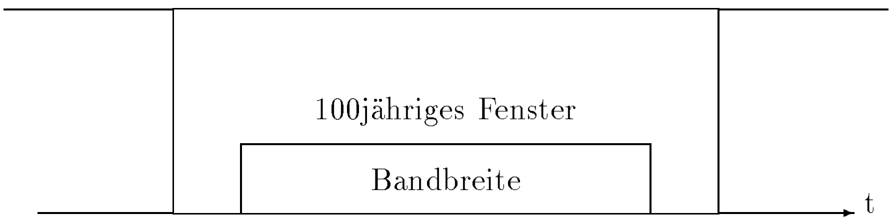
3.3.2 Sliding-Window Technik

Bei dieser Variante wird das geöffnete Fenster „kontinuierlich“ weitergeschoben. Ich habe „kontinuierlich“ unter Anführungszeichen geschrieben, da das Fenster natürlich nicht jedes Jahr um ein Jahr weitergeschoben werden muß. Es reicht, wenn das Fenster spätestens alle n Jahre weitergeschoben wird, wobei max die maximale Bandbreite in Jahren (Laufzeit, ...) bezeichnet. Sollte n gleich 0 sein, dann muß das Fenster jährlich weitergeschoben werden.

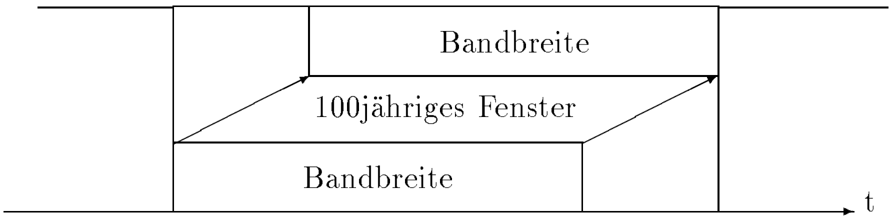
$$n = 100 - max$$

Wenn die maximale Bandbreite < 100 Jahre beträgt, dann kann man den Vorgang auf unterschiedliche Weisen wie z.B.

- um den Mittelpunkt, des sich „kontinuierlich“ verschiebenden 100jährigen Fensters zentriert halten.



- wie bei einem geostationären Sateliten, der innerhalb eines bestimmten Ausschnitts solange in eine Richtung wandert, bis er die Grenze erreicht. Sobald das Fenster die untere Grenze erreicht wird es solange entlang der Zeitachse verschoben bis es die obere Grenze erreicht.



organisieren.

4 Umstellung

Aufgrund der Vorteile, daß jedes Programm seperat nach und nach umgestellt werden kann und die bestehenden Datenbestände auf mehr als 25.000 Magnetbandkasseten nicht verändert werden müßen setzt RACON die „Fixed-Window Technik“ ein, um zu entscheiden ob es sich um ein Jahr im 20. oder 21. Jahrhundert handelt.

Das Problem mit ∞-gültigen Datensätzen wurde, wie bereits erwähnt, dadurch umgangen, daß ein für „normale“ Daten ungültiges Datum verwendet wurde. Solche Datensätze tragen das Datum 99.99.99. Dadurch, daß auch für das Monat und den Tag 99 verwendet wurden, werden sich auch für die Zukunft keine Probleme mit solchen Datensätzen ergeben.

Fehlinterpretationen und Fehlverarbeitungen werden durch eine eingeschränkte Eingabe verhindert. Dabei werden allerdings keine Abbruchbedingungen wie 0 oder 99 eingesetzt.

Dadurch, daß das Unternehmen Software für den Bankensektor entwickelt wurde man bereits vor über 15 Jahren auf dieses Problem aufmerksam. Der Grund dafür war die Vergabe von langfristigen Krediten, deren Laufzeit über das Jahr 1999 hinausgeht. Nur damals dachte man, daß die Programme bis zu diesem Zeitpunkt längst ersetzt wären und modifizierte aus Kostengründen nur den angesprochenen Bereich. Was aus dem Umstand, daß Vorausbuchungen auf Konten maximal ein Monat im voraus erfolgen damals auch nicht notwendig war.

Der Grund warum nicht bereits beim Erkennen des Problems auf vierstellige Datenfelder für das Jahr umgestellt wurde ist, daß bei 30 bis 60 Datumsfeldern pro Konto eine beträchtliche Anzahl an zusätzlichen Resourcen benötigt worden wäre. Mit Resourcen sind in diesem Zusammenhang nicht nur zusätzliche Kapazitäten auf Magnetbändern gemeint, sondern auch des nicht zu vernachlässigenden Punktes des Laufzeitverhaltens auf den Großrechenanlagen vor 15 Jahren. Zusätzliche Rechenkapazitäten wurden zum damaligen Zeitpunkt aus Kostengründen von Management nicht genehmigt.

Aufgrund der Erfahrungen aus dem genannten Bereich und der Erkenntnis, daß doch nicht alle Programme vor dem Jahr 2000 durch neue ersetzt werden können schuf man rechtzeitig eine Möglichkeit zur Lokalisierung aller Datumsroutinen in den diversen Programmen. Desweiteren wird nur eine zentrale Datenaufbereitungsroutine für Datumsfelder verwendet. Anhand der übergebenen Parametern erkennt diese Funktion in welchem Format das Datum von der aufrufenden Funktion gewünscht wird. Die Umstellung auf die Jahr 2000 Kompatibilität stellt auf der einen Seite nur eine Erweiterung dieser Routine dar, die bereits mehrfach erweitert wurde. Natürlich genügt es nicht nur diese Aufbereitungsfunktion mit der „Fixed-Window Technik“ zu modifizieren, sondern es müssen selbstverständlich auch die Funktionen, in denen diese Routine verwendet wird erweitert werden. Die darauf aufbauenden Berechnungen in den Funktionen müssen an die erweiterten Möglichkeit der Aufbereitungsfunktion wahrscheinlich zu großteil nicht geändert werden, aber sie müssen trotzdem überprüft werden.

Daraus resultiert, daß die meisten Programmpakete geändert werden müssen, sodaß sie für das kommende Jahrtausend gerüstet sind. Alle neu zu entwickelnden Anwendungen verwenden in den Berechnungen nur mehr ein JJJJMMTT-Datumsformat, das durch die eben erwähnte Funktion aufbereitet wurde. Auf den Datenträgern selbst wird aber weiterhin nur das bisher verwendete Datumsformat mit 2-stelliger Jahreszahl verwendet. Zur Sicherstellung der „einheitlichen“ Speicherung dient eine Funktion, die als Gegenstück zu der erwähnten Aufbereitungsfunktion gesehen werden kann.

Die Umstellung wurde so geplant, daß sie in 3 Etappen erfolgt, wobei die ersten beiden bereits realisiert wurden, und die dritte und letzte Etappekommendes Jahr über die Bühne gehen wird. Die Aufwandsschätzung für diese letzte Umstellungsetappe beträgt 450 Manntage, die wie die beiden ersten ebenfalls ohne externe Zusatzkräfte bewältigt werden wird.

Die Systementwickler testen die umgestellten Funktionen mit einem kleinen Satz von Testdaten. Anschliessend führt eine 6-köpfige Testgruppe einen Eingabetest durch. Schließlich erfolgt ein automatischer Abnahmetest bei dem eine Simulation auf einer großen Anzahl an Testkonten abläuft.

Neben dieser Anpassung der EDV im Bankensektor gibt es im Zusammenhang mit der bevorstehenden Währungsunion weitere Projekte. Die Umstellung auf den EURO erfolgt parallel und wird bis Anfang 1998 vollzogen sein.