BAPAS-DB: Das offene Realtime-Datenbanksystem für Arbeitsplatzrechner

Rolf Blumenthal, Werum GmbH
Glogauer Straße 2 A, 2120 Lüneburg

Zusammenfassung

Der zunehmende Einsatz von PCs als "kleine Leitrechner" und Arbeitsplatzrechner in Automatisierungssystemen erfordert die Verfügbarkeit von Realtime-Datenbanksystemen auch auf diesen Rechenanlagen. Aus diesem Grunde hat die Werum GmbH, Lüneburg, ihr portables, offenes Realtime-Datenbanksystem BAPAS-DB auf Arbeitsplatzrechner portiert.

BAPAS-DB wurde speziell für technische Systeme mit hohen Anforderungen bezüglich Zugriffszeiten, Ausfallsicherheit und Flexibilität entwickelt. Mehr als 50 Installationen innerhalb umfangreicher Automatisierungssysteme auf Prozeßrechnern haben gezeigt, daß BAPAS-DB diese Anforderungen erfüllt. Die für dieses Ziel relevanten Realtime-Eigenschaften stehen durch die Portierung auf Arbeitsplatzrechner selbstverständlich auch dort zur Verfügung.

Trotz der hohen technischen Anforderung aus dem Automatisierungsbereich wurde bei der Entwicklung auf Komfort für den Benutzer nie verzichtet. Das konzeptionelle Schema der Daten wird problemorientiert definiert, eine Query Language erlaubt wahlfreie Zugriffe auf die Daten, und zur effizienten Ansprache aus Realtime-Programmen steht eine einfache Call-Schnittstelle zur Verfügung. Deshalb erhält auch der datenbanksystem-unerfahrene PC-Benutzer einen leichten Zugang zu BAPAS-DB.

1. Einleitung

Bei der Automation technischer Prozesse werden in zunehmendem Maße auch Arbeitsplatzrechner mit dem Leistungsumfang moderner PCs eingesetzt. Beispiele hierfür sind Fertigungssysteme sowie Prozeßwarten- und -leitstandsysteme.

Hierbei, und insbesondere bei rechnergestützten Dispositionssystemen entfällt der größte Teil der Programmierung auf die Organisation und Ansprache von Datenmengen auf internen und externen Speichermedien, d.h. auf

- Aufbau
- Verkettung
- Verwaltung und
- Prüfung von Dateien
- konkurrierenden Zugriff mehrerer Rechenprozesse und Bediener
  sowie auf die damit verbundenen
- Sicherungs- und Wiederanlaufroutinen.

Mit anderen Worten, jedes der genannten Beispiele erfordert ein umfangreiches, komplexes Datenhaltungssystem.
Da die Software-Entwicklungskosten zum einen unabhängig von der Größe der Rechner sind und zum anderen bei der Entwicklung kleiner Systeme gering gehalten werden müssen, kann nur der Einsatz eines Datenbanksystems zu kostengünstigen Lösungen führen.

2. Zielsetzung

BAPAS-DB (Basis für Prozeßautomationssysteme - Datenbanksystem) wurde mit der Zielsetzung entwickelt, ein offenes Datenbanksystem auf Prozeß- und Minirechnern bereitzustellen, das die hohen Anforderungen technischer Systeme bezüglich Zugriffszeiten, Ausfallsicherheit und Flexibilität erfüllt. Diese Anforderungen gelten jedoch auch bei kleineren Automatisierungssystemen, die heute bereits mit leistungsfähigen PCs realisiert werden können. Deshalb gilt die ursprüngliche Zielvorstellung auch für den Einsatz von BAPAS-DB auf Arbeitsplatzrechnern.

Da Arbeitsplatzrechner eine große Verbreitung auch bei datenbanksystem-unerfahrenen PC-Benutzern finden, muß das Datenbanksystem in der Benutzung und Handhabung unkompliziert sein. Bei der Entwicklung von BAPAS-DB wurde trotz der hohen technischen Anforderungen auf Komfort für den Benutzer nie verzichtet.

3. Wesentliche Merkmale von BAPAS-DB

3.1 Einsatz unter Echtzeitbedingungen

Die Rechenprozesse (Tasks) der Realtime-Programme können konkurrierend zueinander auf die Daten der Datenbank zugreifen.

Der Zugriff auf die Daten erfolgt unter der Regie des betreffenden Rechenprozesses, d.h. die Bearbeitung der Zugriffe von weniger wichtigen Rechenprozessen wird automatisch zugunsten des Zugriffs eines wichtigeren Rechenprozesses verzögert.

Konkurrierend zu den Realtime-Programmen können mehrere Bediener mittels der Abfragesprache QL auf die Daten der Datenbank zugreifen. Der Zugriff auf Daten wird zur Durchführung konsistenter Updates implizit auf Satzebene synchronisiert.

Die Zugriffsverfahren sowie zuverlässige, moderne Techniken für Konsistenz, Integrität, Datenschutz und Datensicherheit gewährleisten einen reorganisationsfreien und ausfallsicheren Betrieb rund um die Uhr.

Nach Störungen erfolgt ein automatischer Wiederanlauf mit konsistentem Zustand.

Die Daten können je Datei wahlweise auf Plattenspeicher oder im Hauptspeicher gehalten werden.

3.2 Sicherungsmaßnahmen

Die Erhaltung der Konsistenz der Daten und die Wahrung der Prozeßintegrität wird durch folgende Maßnahmen gewährleistet:

- Datentyp-Prüfungen
- Schnittstelle für Plausibilitätskontrollen
- Implizite, deadlockfreie Aussperrung und Koordinierung simultaner Zugriffe auf Satzebene
- Implizites Setzen von Sicherungspunkten in Verbindung mit automatischem Wiederanlauf ohne Datenverlust.

Für den Datenschutz stehen Paßwortmechanismen auf Datenbank- und Dateiebene sowie Datentyp-Prüfungen zur Verfügung.

3.3 Einsatz in verteilten Systemen oder Mehrprozessor-Systemen

BAPAS-DB kann in verteilten Systemen und Mehrprozessor-Konfigurationen eingesetzt werden; Beispiele sind:

- BAPAS-DB ist auf einem Arbeitsplatzrechner installiert. Die Ansprache der Datenbank erfolgt von allen anderen Arbeitsplatzrechnern auf den zentralen Datenbankrechner.
- Alle Arbeitsplatzrechner können sowohl auf ihre lokale Datenbank als auch auf die Datenbanken der anderen Rechner zugreifen.

3.4 Zugriffsverfahren

Als Zugriffsverfahren stehen in BAPAS-DB zur Verfügung:

- HASH : Direkter Zugriff mittels eines Hash-Verfahrens über einen eindeutigen Schlüssel und beliebige weitere Nebenbedingungen.
- B*BAUM : Direkter Zugriff mittels eines mehrstufigen Indexlistensystems über einen eindeutigen oder mehrdeutigen Schlüssel und beliebige weitere Nebenbedingungen. Logisch sequentieller Zugriff in der Reihenfolge der Schlüsselwerte.
- INDSQL : Index-sequentieller Zugriff über einen Schlüssel und beliebige weitere Nebenbedingungen. Dieses Verfahren ermöglicht auch die Behandlung von Sätzen variabler Länge. Außerdem können die Sätze mit gleichem Schlüssel in einem Umlaufpuffer organisiert werden.
- MULTIKEY : Direkter Zugriff über mehrere Schlüssel und beliebige weitere Nebenbedingungen. Für jeden Schlüssel ist eines der obigen Verfahren getrennt wählbar.
- FIFO : Sequentieller Zugriff nach der Regel "first in first out".
- LIFO : Sequentieller Zugriff nach der Regel "last in first out".

Diese Zugriffsverfahren sind pro Datenbankdatei vom Benutzer wählbar.

4. Problemorientierte Benutzerschnittstelle

Die Definition der Datenbanken, Dateien, Views und Satzstrukturen erfolgt mittels einer interaktiven, problemorientierten Data Description Language DDL.

Die Abfragesprache QL (Query Language) erlaubt Bedienern das interaktive Einfügen, Anwählen, Ändern und Löschen von Daten. Dabei sind als Suchkriterien wahlfreie Prädikate zugelassen, die sich nicht nur auf die als Schlüssel vereinbarten, sondern auf alle Komponenten eines Satzes beziehen können.

Realtime-Programme benutzen einfache Prozeduraufrufe der jeweiligen Programmiersprache zum Einfügen, Anwählen, Ändern und Löschen von Daten, z.B. PEARL oder PASCAL.

4.1 DDL - Kommandos

Die DDL bietet folgende Funktionen:

- Datenbanken, Dateien und Views
  einrichten, entfernen, definieren, modifizieren.
- Zugriffsverfahren
  definieren, modifizieren, (Dateien) zuordnen, entfernen.
- Privilegien
  vergeben, modifizieren, aufheben.

Beispiel:

In der Datenbank LAGER soll ein neues File mit dem Namen ARTIKEL eingerichtet werden (Kommando **NDBF :** neues Datenbank-File). Jeder Satz soll aus den Komponenten NUMMER, ANZAHL, DATUM und LAGERORT bestehen.

Diese Satzstruktur wird als benutzer-definierter Typ unter dem frei gewählten Namen TYP_ARTIKEL eingeführt (Kommando **NREC :** neuer Recordtyp).

Das Datenbank-File ARTIKEL soll dem Platten-File ARTDAT zugeordnet werden, das für ein Volumen von 3000 Sätzen ausgelegt wird. Die Zugriffe sollen mittels des Zugriffsverfahrens INDSQL erfolgen, wobei NUMMER der Schlüssel sei (Kommando **STRAT :** Zugriffsstrategie). Der VERKAUF soll die Daten von ARTIKEL nur lesen können, die LEITUNG soll alle Privilegien bekommen (Kommando **PRIV :** Zugriffsprivilegien).

```
DB = LAGER ;
NREC = TYPARTIKEL STRUCT [ NUMMER    FIXED (15),
                           ANZAHL    FIXED (15),
                           DATUM     CHAR(6),
                           LAGERORT  KOORDINATE ] ;

NDBF = ARTIKEL,
       FILE   = ARTDAT / VOL = 3000,
       REC    = TYP_ARTIKEL,
       STRAT  = INDSQL / KEYPOS = 1 / KEYLAE = 1 ;
       PRIV   = VERKAUF / READ, LEITUNG / ALL ;
```

4.2 QL - Kommandos

Unter Berücksichtigung komplexer Nebenbedingungen können mittels der QL Sätze aus einem Datenbestand ausgewählt werden. Auf diese ausgewählten Sätze sind dann die Funktionen Listen, Ändern und Löschen anwendbar. Auch ist das Einfügen von Sätzen mit Hilfe der QL möglich. Die angewählten Datenbestände können sowohl aus Datenbank-Dateien als auch aus Views stammen.

Beispiel:

Der VERKAUF benötigt alle Sätze der Datei ARTIKEL, die dem Suchkriterium "NUMMER gleich 389 und ANZAHL kleiner als 10" genügen.

```
ANFANG NAME = VERKAUF,
       PASS = PASSWORT ;
LIST DB = LAGER, DATEI = ARTIKEL MIT NUMMER = 389 & ANZAHL < 10 ;
```

4.3 Ansprache der Datenbank aus Realtime-Programmen

In der jeweiligen Programmiersprache steht eine Sprachschnittstelle zur Verfügung, mit der alle gängigen Datenbank-Operationen aus Realtime-Programmen durchführbar sind.
Einfache Typdefinitionen zur Definition der angewählten Datenbank-Datei und

ihrer Satzstruktur, die Deklaration dieser Objekte und der Aufruf der entsprechenden Prozeduren reichen für die programmierte Ansprache der Daten aus.

Ein Beispiel in PEARL:

```
/* Definition der Datenbank-Datei */

TYPE TYP_DATEI STRUCT [
        MENGEN_IDF      FIXED(15),
        PROZESS_IDF     FIXED(15),
        FEHLER          TYP_FEHLER,
        SPERRE          BIT(16) ] ;

/* Definition des zugehörigen Satzes */

TYPE TYP_ARTIKEL STRUCT [
        NUMMER          FIXED(15),
        ANZAHL          FIXED(15),
        DATUM           CHAR(6),
        LAGERORT        KOORDINATE ] ;

/* Definition der gesamten Datenbank-Datei */

TYPE TYP_ARTIKEL_DATEI STRUCT [
        DATEI           TYP_DATEI,
        SATZ            TYP_ARTIKEL ] ;

/* Deklaration der Datenbank-Dateivariablen */

DCL ARTIKEL TYP_ARTIKEL_DATEI ;
```

Alle Sätze der Datei ARTIKEL mit NUMMER = 389 und ANZAHL kleiner als 10 sollen gelöscht werden.

```
ARTIKEL_CONDITION.KENNUNG := C_SEQUENTIELL ;
        /* Sequentieller Zugriff auf die Datei */
ARTIKEL.DATEI.SPERRE := C_EXCLUSIVE ;
        /* Der Satz ist für weitere Anfragen gesperrt */
```

```
WHILE DBVFORALL ( ARTIKEL, ARTIKEL_CONDITION ) REPEAT
            /* Anwahl der Sätze.                                  */
            /* Der Satz steht in ARTIKEL.SATZ zur Verfügung       */

    IF ARTIKEL.SATZ.NUMMER = 389 AND ARTIKEL.SATZ.ANZAHL < 10
    THEN
        CALL DBVERASE ( ARTIKEL ) ;
            /* Löschen des angewählten Satzes */
    ELSE
        CALL DBVQUITT ( ARTIKEL ) ;
            /* Freigabe des angewählten Satzes */
    FIN ;
END ;
```

5. Implementierung von BAPAS-DB auf IBM PC/AT RTX

BAPAS-DB steht auf dem IBM PC/AT unter dem Realtime-Betriebssystem AT/RTX der Firma RTCS, USA zur Verfügung. Weitere Hardware-Voraussetzungen sind eine 20 MB Winchester, 640 KB Hauptspeicherausbau und der Floating Point Prozessor 8087/80287 von Intel.

Die bisherige Erfahrung zeigt, daß für die meisten Anwendungen ein vorkonfigurierter Standard ausreicht, der folgende Bausteine umfaßt:

- Systemkern mit
  - 16 KB für das Data Dictionary
  - Standard-Puffersystem mit 10 Puffern à 1024 Bytes
  - maximal 20 Dateien
- Strategien FIFO, HASH und Indexsequentiell mit einem Schlüssel
- Recovery mit Checkpointing und Before Image
- Programmschnittstelle zur Ansprache aus PASCAL-, PEARL- und PL/M-Programmen
- DDL Data Description Language
- QL Query Language.

Als weitere Ausbaumöglichkeiten sind jedoch lieferbar:

- Konfigurierung (Voraussetzung: RTX konfigurierbar) mit dateispezifischen Puffersystemen
- Strategien
  - -- B*Baum für direkten Zugriff über einen Schlüssel mit Sortierung
  - -- Indexsequentieller Zugriff mit Umlaufpuffer
  - -- Indexsequentieller Zugriff mit variabler Satzlänge
  - -- Zusatz MULTIKEY für Zugriff über mehrere Schlüssel
- Transaktionen zum Recovery auf Transaktionsebene
- Kommunikationsbaustein zum Einsatz in verteilten Systemen
- Views
- Rekonfigurationsprogramm REKONF.

Durch die Konfigurierbarkeit von BAPAS-DB sind alle relevanten Größen des Datenbanksystems ausschließlich durch die maximal darstellbare Größe ihres Datentyps beschränkt. Deshalb hat BAPAS-DB theoretisch folgende Grenzen:

- Anzahl Dateien	FIXED(15)	äquivalent	32.768
- Anzahl Sätze in Dateien	FIXED(31)	"	2.147.483.647
- Anzahl Seiten in Dateien	FIXED(15)	"	32.768
- Größe einer Seite in 2 Byte	FIXED(15)	"	32.768
- Anzahl Puffersysteme	FIXED(15)	"	32.768
- Anzahl Puffer pro System	FIXED(15)	"	32.768
- Größe eines Puffers in 2 Byte	FIXED(15)	"	32.768
- Größe des Data Dictionary in 2 Byte	FIXED(15)	"	32.768
- Maximale Satzlänge	Seitengröße minus 20 Bytes		

Durch die Festlegung der Systemgrößen wird der Speicherbedarf für Haupt- und Plattenspeicher bestimmt. Deshalb ergibt sich die eigentliche Beschränkung der Systemgrößen durch das Zusammenspiel mit der Hardware-Ausstattung des Rechners.

6. Weitere Verfügbarkeit

Generell ist BAPAS-DB verfügbar für folgende Rechenanlagen:

- AEG 80-30
- GOULD CONCEPT32
- INTEL 286/310 - Systeme
- PCS CADMUS - Serie
- PEARL Engine 68.000
- SIEMENS R/M - Serie

Implementierungen für weitere Arbeitsplatzrechner, wie Hewlett-Packard HP 9000 - 320 und Siemens PC16/20 sowie VAX 11/750 sind in Vorbereitung.