TOTO 総合人事情報システム

TOTO Human Support and Service System

篠 田 博 水

要 約 東陶機器株式会社向けに開発した「総合人事情報システム」の概要と，大規模 CSS ＆データベース・システムの高速応答性と高い生産性とを実現するために EUC & RAD ツールとして開発した「RADs（RDB Application Development supports）」について述べる．また本システムの開発で実施したデータベース・システムのチューニングについても述べる．

Abstract This paper outlines the“ TOTO Human Support and Service System ”we developed for TOTO Co., Ltd. It also describes the “ RDB Application Development supports（RADs）” developed as an EUC & RAD tool to realize quick-response and high-productivity of the large CSS and large database systems. In this paper, we describe the techniques for tuning a database system based on our trial efforts in the development of this system.

1. は じ め に

東陶機器株式会社（以降，TOTO と略す）では，二十数年前よりホスト・コンピュータで稼働中の「人事・給与システム」を以下の理由で再構築することにした．

・システムが陳腐化し，人事制度の変更に対応できなくなっている
・間接部門の業務効率化の促進
・2000 年対応

1994 年から約 1 年をかけ TOTO 社内で構想をまとめ，開発を日本ユニシスが受託した．

「TOTO 総合人事情報システム（TOTO Human Support & Service System，略して HSS システム）」は，1995 年 6 月より 5 か月をかけ「要求定義」をまとめ，12 月より開発に着手，1996 年 12 月より拠点を限定して試行本番，1997 年 4 月より順次本番を行い，1998 年 4 月に全面本番を迎えた．

HSS システムの特徴は，約 230 業務にのぼる人事業務すべてをシステム化し，11,300 人の従業員自らが約 2,400 台の端末（既設置/新規設置の PC）を使用し人事部門への諸届を行うことである．工場のラインでは，従業員は業前/業後/休憩時間帯を利用してエントリを行っている．そのために，使いやすさと応答性が求められた．また，230 業務のシステム化により画面/帳票数は 2,000 を超え，この開発を効率よく行うことも求められた．

本稿では，HSS システムの概要と本システムの統合インフラとして開発した「RADs（RDB Application Development supports）」について報告する．

RADs は大規模 CSS＆データベース，使いやすさ，応答性，高生産性，信頼性，汎用性，そしてシンプルなシステムを追求して開発された 3 層アーキテクチャ統合インフラである．

また，アプリケーション・プログラムのチューニングと本システムのデータベース・ソフトである Oracle データベースのチューニングについても合わせて報告する．

2. システムの要件

2.1 システムの目的

再構築するシステムの目的は以下の通りである．

・人事制度の変更に柔軟に対応できるシステム
・間接業務効率化によるコスト削減
・セルフ・マネージメント
・従業員への情報サービス強化（人事情報のタイムリーな提供）
・社員情報バンク（人材育成，適材適所，情報提供）

2000 年問題がシステム再構築の大きな要因ではある．加えて，21 世紀を間近に控えてますます厳しくなる企業間競争を勝ち抜いていくためには，「従来の年功序列型から成果重視型への人事制度の転換, 多様化する社員の価値観に対応できる人事制度」が求められているが，二十数年前のコンピュータ・システムではそれらの人事制度の変更に対応できなくなっていた．

また，直接部門（製造，販売）の業務効率化に比べ間接部門（コーポレート・スタッフ）の業務効率化が遅れており，競合力のある製品価格とするためにあるいは企業利益を確保するために，間接部門の業務効率化によるコスト削減が求められている．そのためには，「自らの業務の効率化による人事担当の人員削減，個々の従業員のセルフ・マネージメントによる庶務担当の大幅な人員削減」が必要である．
従業員によるセルフ・マネージメントは従業員と所属長に新たな負担を強いることになる．そのために，使いやすい応答性のよいシステムであると同時に人事情報のタイムリーな提供による従業員への情報サービス強化が望まれる．

従来の人事・給与システムは勤務報告，人事考課・昇給・昇格，給与・賞与・年末調整・退職金計算が主であったのに対して，これからの人事情報システムは個々の従業員が入社から退社するまでの企業にとって必要な情報をすべてデータベース化し，人材育成，適材適所，マネージメントへの情報提供に生かして行かなければならない．計算システムから情報システムへ，さらにはトップ・マネージメントの意思決定支援システムへの転換が求められている．

2.2 システムに求められる機能と性能

「2.1 システムの目的」を達成するためにシステムに求められる機能と性能は以下の通りである．

・230 業務のシステム化
・従業員端末（簡便な操作，キーボードレス＆マウスレス）
・大規模システム＆大規模 DB（従業員数 11,300 人，端末数 2,400 台）
・5 秒以内の応答（分散 CSS＆レプリケーション DB）
・ワークフロー（従業員→所属長→人事）
・人事通達（人事→所属長）
・24 時間稼働（3 交替勤務対応，夜間バッチ，無人運転）

一口に人事業務といってもその範囲は人事，給与，勤務，福利厚生，教育，保険までにおよび，TOTO の場合で約 230 業務である．これらの業務処理をすべて見直しシステム化することにより，本社・支社・工場の人事担当の人員削減をはかる．

また，従来は各部門の庶務担当が補助していた個々の従業員と所属長の人事への届出業務を，システムを利用して従業員と所属長がセルフ・マネージメントすることにより庶務担当の人事関連業務を大幅に削減しなければならない．

そのために，だれでもが簡単に使える端末が必要である．なかでも日ごろパソコンと縁のない従業員にとっては，キーボードやマウスを使用しないでも操作できる端末，例えばタッチパネル端末が必須となる．

本システムの大きな特徴は，従来のようにシステムへのエントリを専門家が行うのではなく，11,300 人の従業員が 2,400 の端末を使って自らエントリすることである．これには，接続端末数が 2,400 で最大 2,400 件のトランザクションを同時処理できる能力をもった大規模システムと，11,300 人の従業員の入社から退社までの情報を保存できる大規模データベースが必要とされる．

一般に，操作者がシステムを快適に使えるためには 5 秒以内の応答が必須といわれている．WAN のもとでの集中処理システムにおいて，11,300 人分の大規模データベースで，最大 480 件/秒（2,400 件/5 秒）のトランザクションを同時処理して，大量のデータを画面へ表示する処理（例えば，一か月の勤務票を画面へ部分表示）を 5 秒以内に終了させることは困難である．コンピュータの処理能力が追いついたとしても，WAN の転送能力が追いつかないのが現状である．（もっとも，超大型コンピュータと専用の光ファイバ網の大規模投資をするのであれば別であるが．）

つまり，5 秒以内の応答を確保するためには，転送能力の高い LAN のもとでの分散 CSS と，常に 11,300 人ではなく対象を絞り込んだレプリケーション DB の組み合わせが最適である．

加えて，応答のボトルネックとなるクライアントとサーバ間の転送データを必要最小限におさえるためには，アプリケーションをクライアント側に置いた従来の 2 層アーキテクチャからアプリケーションをサーバ側に移した 3 層アーキテクチャ[1]（プレゼンテーション層はクライアント側，ファンクション層とデータ層はサーバ側）への転換が求められている．また 3 層アーキテクチャは，クライアント側に能力の高い CPU および大量のメモリを必要としない．大部分の処理はサーバ側で行われるため，処理能力の高いサーバがあればよい．

従業員の届出は所属長の承認を経て人事で受け付けされる．（届出によっては直接人事で受け付けされるものもある．）

従業員がシステムに対して入力した届出を，その従業員の所属長へ自動的に送り，承認後は担当の人事部門へ自動的に送るためのワークフロー制御の機能が必要である．

TOTO では社内 OA システムが確立されており，所属長には一人一台のパソコンが割り振られている．当初，人事からの各部門への通達もこの社内 OA システムのメールを利用することを検討したが，機密性と優先度の問題で他の業務メールとは区別する必要性が生じた．また，人事からの通達は個人よりも組織宛であり，人事異動や組織異動によって都度メール・アドレスを変更しなくても自動的に該当する組織長

に通達を送る機能が必要とされた．
そのために，人事から各部門の組織長へ人事通達を送る機能を HSS システムに取り込むこととした．

TOTO では 3 交替勤務の工場があり，そのために 6 時～23 時までは従業員にシステムをオープンしなければならない．残りの時間でデータベースのバックアップと業務バッチを実行するために，無停止の 24 時間稼働が必須である．かつこれらが無人で自動運転されなければならない．

本システムでは特に，2,400 端末 11,300 人の大規模 CSS＆大規模 DB システムにおける 5 秒以内の応答性の確保と，230 業務の大規模開発における生産性が重要なポイントであった．

3. システムの概要

図 1 はシステムの概要図である．

HSS システムは，人事部門の主要な業務を遂行する「人事サーバ」を核に，従業員からのトランザクションを処理する全国 30 拠点に配置された「拠点サーバ」，健康保険組合の健保業務を遂行する「健保サーバ」，経理システムや他のシステムが利用するための公開人事データベースそして大量/特殊印刷や銀行をはじめとする外部団体とのインタフェースを受け持つ「ホスト・コンピュータ」，そして 60 台の人事部門クライアントと 2,300 台の部門クライアントが「TOTOΣネット」で有機的に結ばれた統合システムである．その他に全国 10 工場に設置されている「食堂 POS」が電話回線で人事サーバにつながっている．

人事サーバと人事部門クライアント間，拠点サーバと部門クライアント間，そして人事サーバと拠点サーバ間はプロセス間通信で，人事サーバとホスト・コンピュータ間，人事サーバと健保サーバ間はファイル転送で，そして人事サーバと食堂 POS 間は BSC 伝送制御手順でデータ転送を行う．

3.1 ハードウェアの構成

人事サーバ，拠点サーバ，バックアップ・サーバ，そして管理サーバには実績のある UNIX サーバに替えて PC サーバを採用した．その理由は，1997 年 4 月の本番時を見据えて，

・PC サーバに格段の進歩が予想される
・コスト・パフォーマンスにおいて PC サーバが優れている
・文字コードがシフト JIS で統一できる

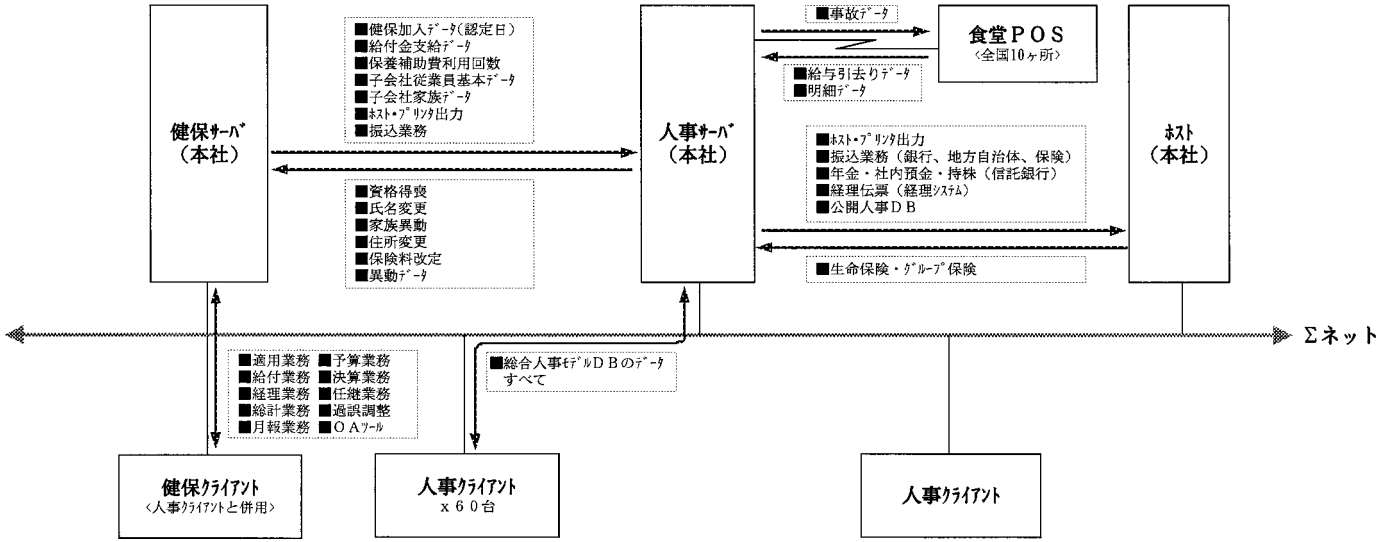
からである．

事実，1995 年 3 月提案時点における PC サーバの能力に比べて，1997 年 4 月本番時点の PC サーバの能力は 5 倍近いコスト・パフォーマンスを計測した．

3.2 ソフトウェアの構成

図 2 はソフトウェアの構成図である．一点鎖線で囲まれた各システム内の実線矩形部のプログラムが今回の開発対象である．

〈本社（人事部門）〉



〈本社（他部門）・支社・工場〉

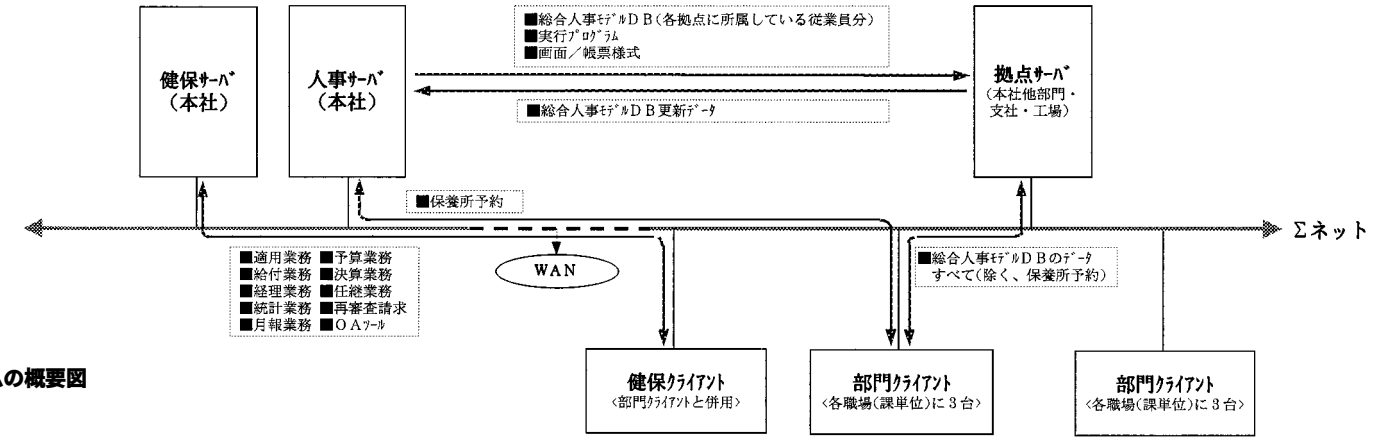


図 1　システムの概要図

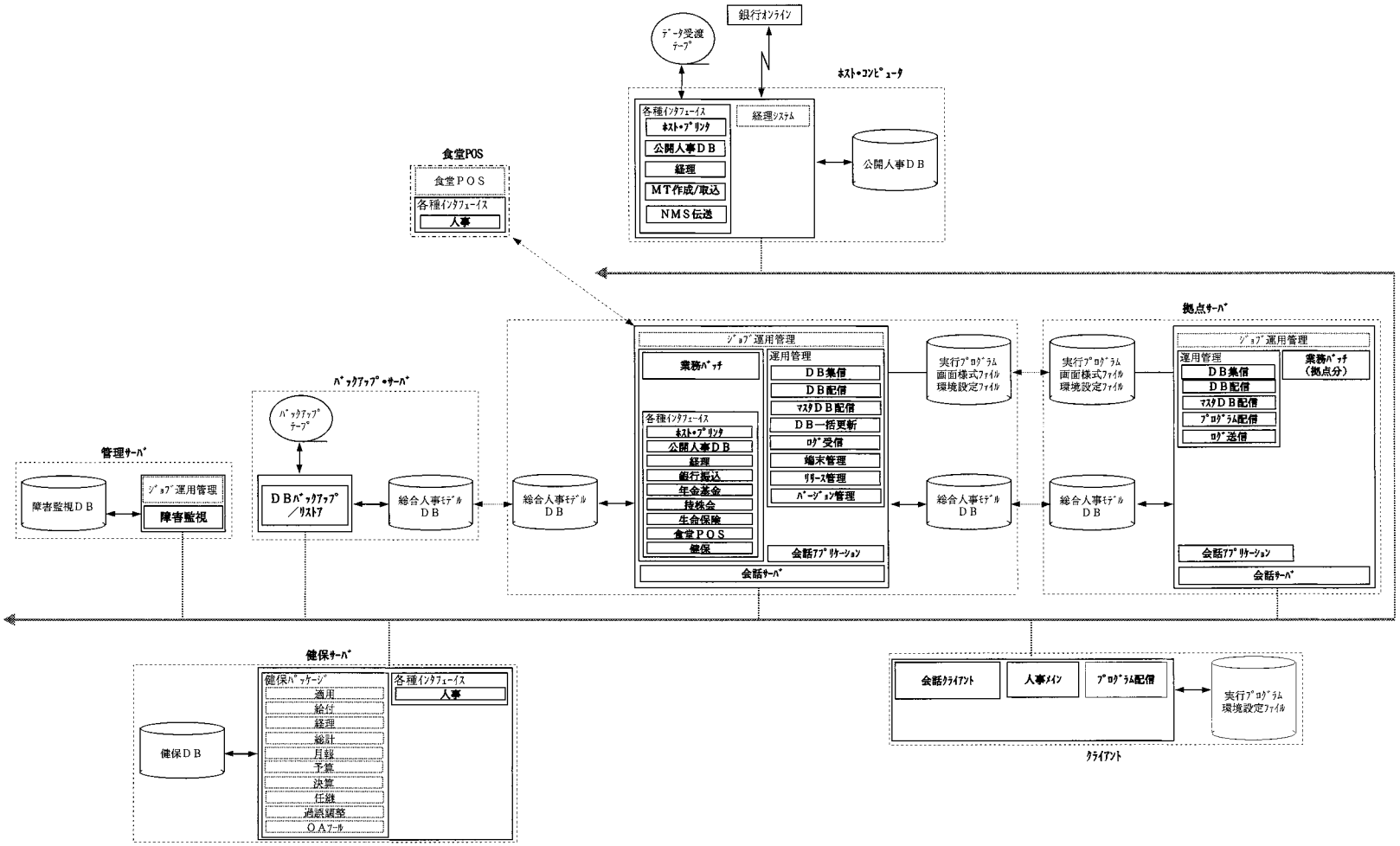


図 2　ソフトウェアの構成

3.2.1 人事サーバ

会話サーバ，会話アプリケーションはクライアントから送信された「従業員が入力する各種届出，所属長が入力する届出承認」に加えて「人事部門の届出受付，人事部門の管理統計，人事通達」を処理する．

業務バッチは給与計算などの一括処理であり，月初に月間スケジュールを決めジョブ運用管理に登録する．そして，設定された日時にジョブ運用管理によって実行される．実行指定時刻は主に夜間である．約 200 の業務バッチが用意されている．

各種インタフェースにはホスト・コンピュータとの間でテキスト・ファイルを転送する「ホスト・プリント，公開人事 DB，経理，銀行振込，年金基金，持ち株会，生命保険」，同様に健保サーバとの間でテキスト・ファイルを転送する「健保」，そして食堂 POS との間で BSC 伝送制御手順によりテキスト情報をやりとりする「食堂 POS」がある．

運用管理については「4．インフラ・ソフトの構成」で詳しく述べる．

3.2.2 拠点サーバ

会話サーバ，会話アプリケーションはクライアントからの「従業員が入力する各種届出の処理，所属長が入力する届出承認の処理」を実行する．

業務バッチ（拠点分）は人事サーバで実行される業務バッチのサブセットであり，DB の整合性が保てる処理で，処理結果を人事サーバから伝送するよりも拠点サーバで実行するほうが効率のよい一括処理である．人事サーバと同様に，月初に月間スケジュールを決めジョブ運用管理に登録する．そして，設定された日時にジョブ運用管理によって実行される．

運用管理については「4．インフラ・ソフトの構成」で詳しく述べる．

3.2.3 管理サーバ

管理サーバのジョブ運用管理が総元締めであり，設定されたジョブ・スケジュールにしたがって人事サーバと拠点サーバのジョブ運用管理に指令を送り各サーバのジョブ・ネットワークを実行する．

障害監視は人事サーバと各拠点サーバが正常に稼働中かどうかを定期的に調べ，異常があった場合は運用管理者にその状況をポケベル等で知らせる．

詳細は「4．インフラ・ソフトの構成」で述べる．

3.2.4 バックアップ・サーバ

バックアップ・サーバは人事サーバが停止したときのためのコールド・スタンバイ機であり，通常は人事サーバの DB のオンライン・バックアップと夜間のコールド・バックアップを行い，DB をバックアップ・テープへ保存する．

バックアップ・サーバの役割につては「6．DB バックアップ」，「7．障害対策」で詳しく述べる．

3.2.5 健保サーバ

健保サーバは「Unisys RX 7000 健保システム」であり，健保パッケージのほかに人事サーバとの間でテキスト・ファイルの転送を介して各種の情報をやりとりするインタフェースが含まれる．

3.2.6 クライアント

人事メインは入力された従業員のユーザ ID（社員番号）により，その従業員が属する拠点サーバに接続する．クライアントと LAN でつながっている拠点サーバに接続されるとは限らない．どこにいても個々の従業員の接続すべき拠点サーバは一意に決まっている．（正確には組織単位で拠点サーバが決められている．）例えば，出張先で HSS システムを使用する場合でも，接続先の拠点サーバは出張先の拠点サーバではなくその従業員の属する拠点サーバである．

会話クライアントは「従業員の各種届出，所属長の届出承認」に加えて「人事部門の届出受付，人事部門の管理統計，人事通達」の画面を表示し入力されたデータを接続サーバへ送信し，会話アプリケーションにより「総合人事モデル DB」を検索/更新する．画面/帳票数は約 2,000 である．

プログラム配信については「4．インフラ・ソフトの構成」で詳しく述べる．

3.2.7 総合人事モデル DB

人事業務にとって必要な「人（従業員，パート，嘱託），もの（施設，制度），金（給料，給付金，見舞金，退職金，控除金，...）」の情報を DOA（データ中心アプローチ）[2]によりモデル化したリレーショナル・データベースであり，人事モデル，給与モデル，勤務モデル，福利厚生モデル，教育モデル，保険モデル，組織モデルの七つのモデル（600 表，6,000 データ項目）からなる．

要求定義，基本設計でもちいた DOA によるモデル化技法については，本報告では割愛し次の機会に報告することとする．

3.3 データベースの構成

HSS システムは 5 秒以内の応答確保とハードウェア投資コストの削減のために分散 CSS の形態を取り，総合人事モデル DB も各拠点サーバに分散している．人事の DB は，本来は集中 DB が望ましいために，物理的には分散している DB を論理的には集中 DB として常に整合性を取っていかなければならない課題が生じる．DBMS の提供するレプリケーション機能を使うと DB の整合性は保たれるが，分散 WAN 環境におけるレプリケーションの場合に 5 秒以内の応答確保は可能か，障害時のデータの信頼性は十分か，人事サーバと拠点サーバのいずれかがダウンしていても運用はできるかなど新たな課題が想定され，使用を断念した．以下は人事業務の特性を生かした分散 DB の整合性問題の解決法である．

クライアントから入力されたトランザクションは，拠点サーバに蓄えられたのちまとめて人事サーバに送られる．人事サーバでは，これらのトランザクションのうち，総合人事モデル DB を更新するための条件が整っているもののみが総合人事モデル DB を更新する．例えば，従業員の届出トランザクションは所属長が承認するまでは総合人事モデル DB を更新しない．未承認のトランザクションは承認されるまで拠点サーバと人事サーバに置かれ，所属長が承認し更新可能年月日に到達すると人事サーバの総合人事モデル DB を更新する．

更新結果はトランザクションが発生した拠点サーバに配信され，拠点サーバの総合人事モデル DB が更新される．同時にトランザクションは削除される．

このデータベースの更新方法を「グローバル非同期更新」[3]という．

この更新方法により，人事データベースの即時更新は行わないものの，レプリケーション機能を使った場合の応答時間への悪影響やリカバリの難しさを回避した．

総合人事モデル DB の各表は属性に配信タイプをもつ．配信タイプには「全拠点へ配信する，該当拠点（該当する従業員が属する拠点）へ配信する，拠点へ配信しない」の 3 タイプがある．表の更新結果はこの配信タイプによってトランザクションが発生した拠点サーバ以外の全拠点サーバに送られることもあるし，いずれの拠点サーバに送られないこともある．

バックアップ・サーバは総合人事モデル DB をオンライン・バックアップと毎夜間のコールド・バックアップで不慮の障害に備えている．詳細は「6．DB バックアップ」で述べる．

4.　インフラ・ソフトの構成

図 3 はインフラ・ソフトの構成，表 1 はその一覧である．

4.1　オペレーティング・システム

クライアントのオペレーティング・システムは Windows 3.1 と Windows 95 の混在である．すでに社内 OA システム用の PC が大半の管理職に配置されており，それらはすべて Windows 3.1 である．新規設置の PC は Windows 95 で統一した．

そのため，クライアントの VC＋＋でプログラムされたプレゼンテーション層は機能，性能とも Windows 3.1 に合わせざるを得なかった．

サーバのオペレーティング・システムは PC サーバを選択したために必然的に Windows NT である．Windows NT の性能については当初からまったく心配していなかったが，安定性について若干の不安があった．

開発期間を含めて 2 か年使用したが，メモリに十分余裕があり本番のアプリケーションしか実行されない限定した環境においては問題は発生せず，安定しているといえる．

以下は HSS システムから見たサーバのオペレーティング・システムの要件である．

・マルチ・プロッセシング
・バッチ・ジョブ実行環境
・TCP/IP とプロセス間通信
・ディスク共有
・動的メモリ管理
・データベース・システムの稼働実績
・C 言語プログラミング

4.2　データベース

データベース・マネージメント・システムには最も実績のある Oracle 7 を選択した．Oracle 7 もメモリに十分余裕がある環境では安定している．性能についても当初の期待以上の値が得られている．Oracle 7 のチューニングについては「8．効率改善」で詳しく述べる．

データベース開発支援ツールとして「Designer 2000」と「Erwin」を検討したが，以下の理由で使用していない．（注：1996 年初の詳細仕様検討時における Designer

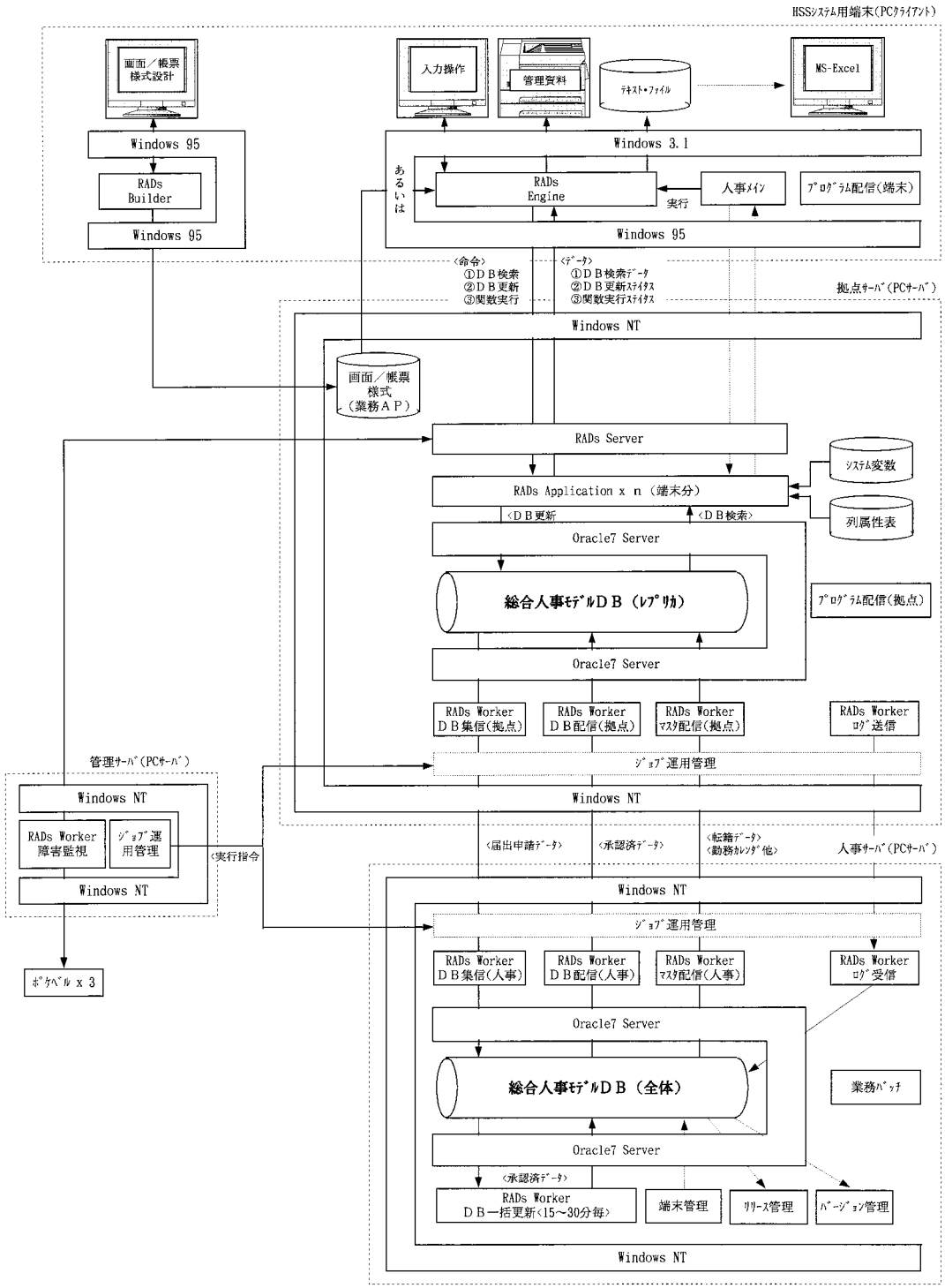


図 3　インフラ・ソフトの構成

表 1　インフラ・ソフト一覧

分類	ソフトウェア名	内容
1. オペレーティング・システム	Windows 3.1	GUI(グラフィック・ユーザインターフェイス)機能をもったマルチ・ウィンドウズ システム環境を提供する16ビットPCクライアントのオペレーティング・システム. (注:通信ソフトTCP/IPは含まず. 別にSoliton TCP/IP、Winsockが必要.)
	Windows 95	強力なGUI(グラフィック・ユーザインターフェイス)機能をもったマルチ・ウィンドウズ システム環境を提供する32ビットPCクライアントのオペレーティング・システム. (注:通信ソフトTCP/IPを標準に提供する。)
	Windows NT	GUI(グラフィック・ユーザインターフェイス)機能をもったマルチ・ウィンドウズ システム環境とマルチ・ジョブ環境を提供する32ビットPCサーバのオペレーティング・システム. (注:通信ソフトTCP/IPを標準に提供する。)
2. データベース	Oracle7 Server	PCサーバ用のリレーショナル・データベース・マネージメント ソフトウェア. →3層構造CSSのデータ層. (C言語用プログラム・インターフェイスとしてPro*C、会話型ユーザ・インターフェイスとしてSQL/PLUSを提供する.)
3. EUC&RADツール	RADs Builder	PC端末から様々な形式でRDBの内容を検索, 更新, 挿入するための「RADs(RDB Application Development supports)」の開発モジュール.
	RADs Engine	RADs実行モジュールのクライアント側ソフトウェア. →3層構造CSSのプレゼンテーション層.
	RADs Server	接続要求のあった RADs Engine とアイドルな RADs Application を結び付け, クライアントとサーバの稼動状況を管理する.
	RADs Application	RADs Engine からの要求により, RDBの検索, 更新, 挿入, およびアプリケーションの関数の実行を行い, その結果をかえす. 業務APIは関数として登録する. →3層構造CSSのファンクション層.
	RADs Worker DB集信	DBのグローバル非同期更新のため, 拠点サーバ側で画面より入力されたトランザクションを人事サーバに集信する. (注:ジョブ運用管理により制御される.)
	RADs Worker DB配信	DBのグローバル非同期更新のため, 人事サーバ側でマスタDBへ反映されたデータ(DB配信表)を拠点サーバへ配信する. (注:ジョブ運用管理により制御される.)
	RADs Worker DB一括更新	DBのグローバル非同期更新のため, 承認済みトランザクションを人事サーバのマスタDBへ反映するとともにDB配信表へ書込む. (注:ジョブ運用管理により制御される.)
	RADs Worker マスタ配信	人事サーバ側のバッチ・ジョブでマスタDBへ反映されたデータをDB同期のため拠点サーバへ配信する. (注:ジョブ運用管理により制御される.)
	RADs Worker ログ送信	クライアントのログイン/ログアウト時に, それぞれログイン情報/ログアウト情報を人事サーバ側のログ受信へ送信する.
	RADs Worker ログ受信	拠点サーバのログ送信よりログイン情報/ログアウト情報を受取りDBへ保存する. ログイン情報/ログアウト情報は利用状況の把握と障害発生時の対応に使われる.
	RADs Worker 障害監視	あらかじめ設定された時間間隔で人事サーバ, バックアップ・サーバ, そしてすべての拠点サーバの RADs Server に応答を要求し, 異常応答あるいはタイムアウトが発生したときポケベルで運用管理者に知らせる. (注:ジョブ運用管理により制御される.)
4. 端末・バージョン管理	端末管理	人事サーバ側に集められた使用者ログより, 端末の諸情報を抜き出し端末管理表に書込むことにより端末を管理する.
	リリース管理	バージョンアップのため, 画面／帳票様式, 実行プログラム, 環境ファイルのリリースとその管理を行う. (リリース用ホルダへコピーした後は, プログラム配信により各拠点サーバ＆端末へ配信される.)
	バージョン管理	バージョンを管理し, 人事サーバ側＆拠点サーバ側のリリース用ディレクトリより旧バージョンの全端末に配信済のファイルを削除する.
	プログラム配信	①拠点サーバ:人事サーバのリリース用ホルダをマップしてバージョンをチェックした後, バージョンアップされているファイルを自分の実行環境とリリース用ホルダへコピーする. (原則:1回／日, 夜間に実行される.) ②端末:拠点サーバのリリース用ホルダをマップしてバージョンをチェックした後, バージョンアップされているファイルを自分の実行環境へコピーする. (人事システムを立ち上げた都度実行される.)

5. その他	ｼﾞｮﾌﾞ運用管理	ｶﾚﾝﾀﾞｰにｼﾞｮﾌﾞﾈｯﾄを設定し，設定された日時でｼﾞｮﾌﾞﾈｯﾄを実行・監視する.
	人事ﾒｲﾝ	Human S&S ｼｽﾃﾑを起動するﾌﾟﾛｸﾞﾗﾑ. ユーザ'IDとﾊﾟｽﾜｰﾄﾞの入力を受けつけ，そのﾕｰｻﾞの該当拠点ｻｰﾊﾞを探し接続する.
	画面/帳票様式	RADs Builder によって作成した業務ｱﾌﾟﾘｹｰｼｮﾝの画面／出力帳票／出力ﾃｷｽﾄ・ﾌｧｲﾙの様式と処理手続き(会話型業務AP). RADs Engine によって解読され実行される.
	業務ﾊﾞｯﾁ	主に夜間実行される業務一括処理ﾌﾟﾛｸﾞﾗﾑ.（勤務集計，給与計算，賞与計算，年調計算，. . . etc）（注:ｼﾞｮﾌﾞ運用管理により制御される. ）
	ｼｽﾃﾑ変数	Human S&S ｼｽﾃﾑの運用関連＆業務関連の変数名と値が設定されているﾃｷｽﾄ・ﾌｧｲﾙ.（人事ｻｰﾊﾞ名，拠点ｻｰﾊﾞ名，各種ﾎﾟｰﾄ番号，. . . ，当日，当年度，当月，給与計算月度，. . . etc） これらの変数は，運用管理ﾌﾟﾛｸﾞﾗﾑ，業務ﾊﾞｯﾁ，そして画面／帳票様式の処理手続きで参照される. また，変数の値は毎日の夜間ﾊﾞｯﾁで更新される.
	列属性表	すべての表の列属性が設定されているﾃｷｽﾄ・ﾌｧｲﾙ. （表名，列名，ﾃﾞｰﾀの型，ﾃﾞｰﾀの長さ，ｷｰ有無，最小値，最大値，参照ｾｷｭﾘﾃｨ，更新ｾｷｭﾘﾃｨ） ﾃﾞｰﾀﾍﾞｰｽ・ｽｷｰﾏの変更にともない自動更新される.

2000 と Erwin の機能で検討した結果である.）

- 表名に業務で用いる日本語を使用し，かつ表名がデータの論理的階層を表しているために，Oracle 7 の制限である 30 バイトを超える．そのため，外部表名と内部表名の対応テーブルをもち，アプリケーションにたいしては外部表名，Oracle 7 にたいしては内部表名で操作できるようにした.
- 600 表のうち多くの表が「キー：社員番号」で関係づけられるが，表数が多すぎて ER 図に描ききれない.
- 20 人以上の開発者が日々スキーマを変更する．開発時点では誰でもが簡単に修正できるツールでなければならない.

そのため自前で Ms-Excel によるスキーマ・シートを作成し，このスキーマ・シートから Oracle 7 の表を生成するツールも作成した.

Oracle 7 はレプリケーション DB の機能を提供しているが，前述（3.3 データベースの構成）の理由で使用していない．そのかわりに「グローバル非同期更新」で分散 DB を実現している．グローバル非同期更新については「5. RADs」で詳しく述べる.

また，ストアード・プロシージャも以下の理由で使用していない.

- 複雑な条件処理が多く，SQL＋でプログラムすることが難しい.
- アプリケーションは外部表名で DB を操作するが，ストアード・プロシージャには外部表名は記述できない.
- 複雑な処理を行うと Pro＊C＋C プログラムに比べて CPU を多く消費する.

すべてのアプリケーションからの DB へのアクセスは，C プログラムの関数呼び出しで「基本 DB 操作関数」を通して行われる．基本 DB 操作関数は Pro*C の部分を外部表名から内部表名への変換を含めて C プログラムの関数として提供するもので，アプリケーションには外部表名でかつ Pro*C を知らなくても DB にアクセスできるようにした.

4.3 EUC & RAD ツール

230 業務，2,000 の画面/帳票を作成するには生産性の高い RAD ツールが必要である．従来のようにプログラム仕様書を作成し，それを専任のプログラマが VB あるいは VC＋＋でプログラムしていたのでは高生産性は望めない．業務仕様書をもとに業務担当者あるいはシステム・エンジニアが直接に画面/帳票を作成できる EUC & RAD ツールが求められる．

我々は HSS システムの開発のために，汎用性が高く，高生産性が期待でき，マルチ・サーバ，マルチ・データベース，分散 CSS，分散 DB をサポートする 3 層アーキテクチャの EUC & RAD ツール「RADs（RDB Application Development supports）」を開発した．詳細は「5．RADs」で述べる．

4.4 ジョブ運用管理

ジョブ運用管理の要件は以下の通りである．

- 異なるサーバ間で実行されるジョブのネットワーク（ジョブネット）が作成できること．
- 各サーバで実行されるジョブネットの管理を画面で会話形式に行えること．
- 管理は 1 年間とし，年度，半期，四半期，月次，日々にそれぞれ実行するジョブネットを登録できること．
- ジョブネットは設定日の設定時刻に自動的に実行されること．（ただし，先行ジョブネットが終了している場合に限る．）
- 実行終了，実行中，実行待ちのジョブネットは画面でモニタでき，リアルタイムに状況を把握できること．
- 状況に応じて実行スケジュールが変更でき，優先順位の変更，中断，再開，強制終了などの柔軟な運用ができること．
- ジョブにエラーが発生した場合，アラーム機能として運用管理者にポケベル等の呼出しができること．

これらの要件に対し，必要条件を満たしている市販品を評価し，良い結果を得たので採用した．

4.5 端末・バージョン管理

開発管理者は定期リリースや緊急リリースで，人事サーバ上のリリース管理画面よりリリース情報（バージョン名とリリースする実行プログラム，画面/帳票様式，環境設定ファイルのパス名）を入力/選択する．リリース情報はデータベースのバージョン管理表へ書かれ，毎夜間に実行される「リリース管理プログラム」によりバージョン名と各リリース・ファイルが配信ホルダへ複写される．

拠点サーバ側で毎夜間実行される「プログラム配信」は人事サーバの配信ホルダを共有化し，人事サーバの配信ホルダのバージョン名と自分の配信ホルダのバージョン名を比較する．そしてバージョン名が異なるときは，新しいバージョン名と作成日付が異なるファイルを自分の配信ホルダと実行ホルダへ複写する．

クライアント側で HSS システムが起動されるたびに接続先の拠点サーバの配信ホルダを共有化し，拠点サーバの配信ホルダのバージョン名と自分の実行ホルダのバージョン名を比較する．そしてバージョン名が異なるときは，新しいバージョン名と作

成日付が異なるファイルを自分の実行ホルダへ複写する．

以上の処理によって，当日リリースした新バージョンは翌日以降の使用時に各拠点サーバとクライアントに届き，使用時点では最新バージョンが保証される．

しかしこのままでは配信ホルダにリリース・ファイルがたまってしまい，クライアントの HSS システム立ち上げ時のバージョン・チェックに多くの時間を費やすことになる．すべてのクライアントにリリース済みの古いファイルは人事サーバと拠点サーバの配信ホルダより削除しなければならない．

各クライアントのバージョンはプログラム配信処理後のログイン情報で人事サーバに集められる．このバージョンを調べればリリース済みのバージョンで最も古いバージョン名がわかる．「端末管理」と「バージョン管理」は定期的にこの古いバージョン名のリリース・ファイルを人事サーバと拠点サーバの配信ホルダから削除する．

4.6 障 害 監 視

障害監視は管理サーバの役割である．障害監視は設定された時間間隔で人事サーバから順にすべての拠点サーバに対して Ping を発行しネットワークと Windows NT の稼働状況を調べ，接続可能であれば各サーバの RADs Server にステイタスの問い合わせを行いアプリケーションの状態を調べる．接続不能あるいは RADs Server から異常ステイタスがかえされたとき（含む，応答なし）は運用管理者にその状況とサーバ名を知らせる．

5. RADs

5.1 RADs の概要

RADs（RDB Application Development supports）は大規模分散 CSS 環境と大規模分散データベース環境において，開発の高生産性と高速な応答性を満たすオープン・システムにおける汎用の統合インフラである．以下にその特徴をあげる．

- RDB 一体型システム
- 3 層アーキテクチャによる大規模分散 CSS
- グローバル非同期更新による大規模分散データベース
- マルチ・サーバ
- マルチ・データベース
- ワークフロー制御
- 障害監視機能
- 豊富な出力（画面，帳票，テキスト・ファイル）
- 日本語環境（表名，列名，関数名，変数名，引数名）
- オブジェクト・イベント・ドリブン

RADs は以下の 3 モジュールで構成される．

- RADs 開発モジュール（RADs Builder）
- RADs 実行モジュール（RADs Engine＋RADs Server＋RADs Application）
- RADs 運用モジュール（RADs Worker）

5.2 RADs 開発モジュール

業務処理のうち会話処理の部分を RADs Builder で作成する．作成された処理手続

きを「画面/帳票様式」という（図 4 と図 5）.
画面/帳票様式は画面，帳票，テキスト・ファイルに出力する場合いずれも同じ形式である．ただし出力領域の座標系は，画面は仮想ラスタ座標系（モニタの種類にかかわらず 1,024×768 がフル画面）で，帳票は用紙種・用紙方向と mm 座標系で，テキスト・ファイルは文字数と行数座標系で設定する.

画面/帳票様式は様式本体とデータ本体から成る. 様式本体は画面/帳票/テキスト・ファイルに貼り付けられるオブジェクトの集合であり，データ本体はデータベースに対する検索条件によって結合限定された論理表に対応する．論理表の列がデータ本体の項目である.

様式本体のオブジェクトとデータ本体の項目は Ms-Excel のオブジェクトと同様の操作で画面に配置し属性を設定する．画面にはグリッドが表示されており，配置されたオブジェクトと項目はグリッドに丸められる（図 6）.
また，画面/帳票様式は n 階層の子画面/帳票様式呼び出しが可能である.
RADs は VB との比較において，倍以上の生産性が期待できる.

5.3 RADs 実行モジュール

RADs Server と RADs Application×接続クライアント数分はサーバの立ち上げでメモリに常駐される．RADs Engine は RADs Server によりアイドル状態の RADs Application と接続を確立して，引数で指定された画面/帳票様式を解釈し「初期化演算式」を実行後，各データ本体に設定されている「DB 検索条件式」によりデータベースを検索し結果のデータを表示する．以降は操作者によるイベントを処理し RADs Application に登録されている関数の実行，「DB 更新条件式」によるデータベースの更新，あるいは子画面/帳票様式の呼び出しを行う．図 7 は出張届の画面/帳票様式の実行例である.

同時 30 台実行の環境で，RADs は残業届，休暇欠勤届，住所変更届などの単一行を検索し表示する処理を 1 秒以内，月間勤務票のように複数行を検索し表示する処理を 2 秒以内で終了することができ，目標の 5 秒以内を十分にクリアした.

データベースの検索，更新，挿入機能は RADs Application が標準に提供するが，業務特有の機能は C プログラムにより関数を作成し，関数ディスパッチャとともに RADs Application に DLL リンクしなければならない（図 8）.

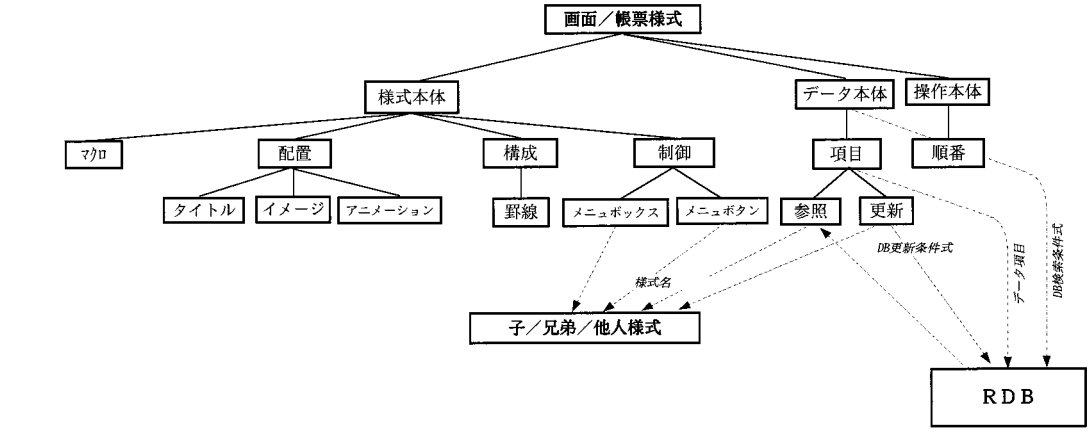
5.4 RADs 運用モジュール

RADs Worker は次の四つのグループに大別される.

・グローバル非同期更新を制御する DB 集信，DB 配信，DB 一括更新.
・DB を拠点へ配信するマスタ配信.
・ログ情報を管理するログ送信とログ受信.
・ネットワークとサーバおよびアプリケーション・システムの障害監視.

クライアントから入力されたデータベース更新のためのトランザクションは拠点サーバの「更新 DB」に貯えられる.「グローバル非同期更新」のサイクルが開始されると,

1） 拠点サーバの DB 集信は未転送のトランザクションを更新 DB より集め人事サーバの DB 集信へ送信する．人事サーバの DB 集信はそれらのトランザクション


画面／帳票様式
様式本体
データ本体
操作本体
ﾏｸﾛ
配置
構成
制御
項目
順番
タイトル
イメージ
アニメーション
罫線
メニュボックス
メニュボタン
参照
更新
様式名
DB更新条件式
データ項目
DB検索条件式
子／兄弟／他人様式
RDB

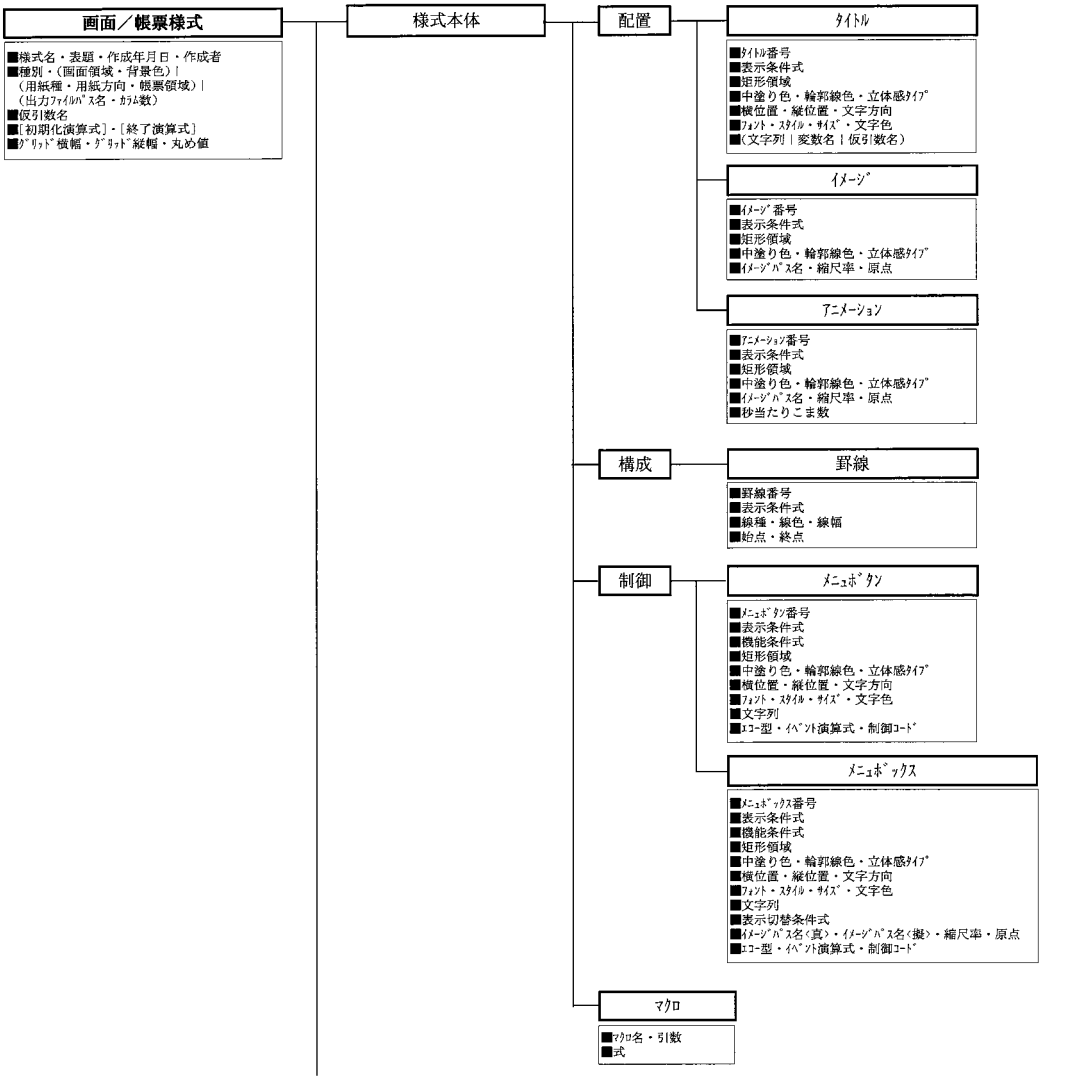

画面／帳票様式
■様式名・表題・作成年月日・作成者
■種別・(画面領域・背景色)|
(用紙種・用紙方向・帳票領域)|
(出力ﾌｧｲﾙﾊﾟｽ名・ｶﾗﾑ数)
■仮引数名
■[初期化演算式]・[終了演算式]
■ｸﾞﾘｯﾄﾞ横幅・ｸﾞﾘｯﾄﾞ縦幅・丸め値
様式本体
配置
ﾀｲﾄﾙ
■ﾀｲﾄﾙ番号
■表示条件式
■矩形領域
■中塗り色・輪郭線色・立体感ﾀｲﾌﾟ
■横位置・縦位置・文字方向
■ﾌｫﾝﾄ・ｽﾀｲﾙ・ｻｲｽﾞ・文字色
■(文字列|変数名|仮引数名)
ｲﾒｰｼﾞ
■ｲﾒｰｼﾞ番号
■表示条件式
■矩形領域
■中塗り色・輪郭線色・立体感ﾀｲﾌﾟ
■ｲﾒｰｼﾞﾊﾟｽ名・縮尺率・原点
ｱﾆﾒｰｼｮﾝ
■ｱﾆﾒｰｼｮﾝ番号
■表示条件式
■矩形領域
■中塗り色・輪郭線色・立体感ﾀｲﾌﾟ
■ｲﾒｰｼﾞﾊﾟｽ名・縮尺率・原点
■秒当たりこま数
構成
罫線
■罫線番号
■表示条件式
■線種・線色・線幅
■始点・終点
制御
ﾒﾆｭﾎﾞﾀﾝ
■ﾒﾆｭﾎﾞﾀﾝ番号
■表示条件式
■機能条件式
■矩形領域
■中塗り色・輪郭線色・立体感ﾀｲﾌﾟ
■横位置・縦位置・文字方向
■ﾌｫﾝﾄ・ｽﾀｲﾙ・ｻｲｽﾞ・文字色
■文字列
■ｺｰﾄﾞ型・ｲﾍﾞﾝﾄ演算式・制御ｺｰﾄﾞ
ﾒﾆｭﾎﾞｯｸｽ
■ﾒﾆｭﾎﾞｯｸｽ番号
■表示条件式
■機能条件式
■矩形領域
■中塗り色・輪郭線色・立体感ﾀｲﾌﾟ
■横位置・縦位置・文字方向
■ﾌｫﾝﾄ・ｽﾀｲﾙ・ｻｲｽﾞ・文字色
■文字列
■表示切替条件式
■ｲﾒｰｼﾞﾊﾟｽ名〈真〉・ｲﾒｰｼﾞﾊﾟｽ名〈擬〉・縮尺率・原点
■ｺｰﾄﾞ型・ｲﾍﾞﾝﾄ演算式・制御ｺｰﾄﾞ
ﾏｸﾛ
■ﾏｸﾛ名・引数
■式

図 4　つづく

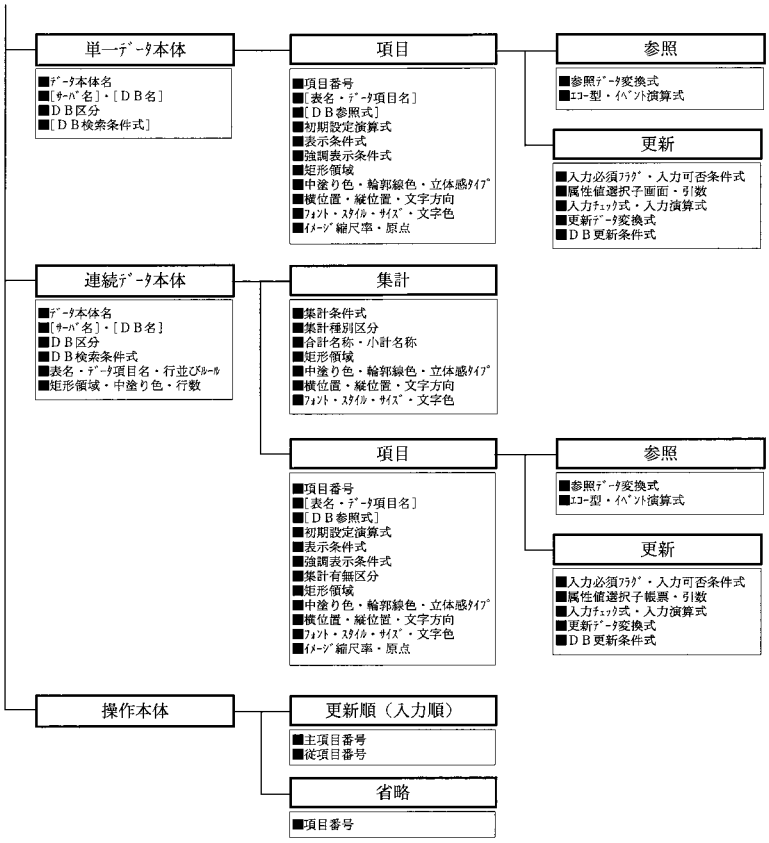
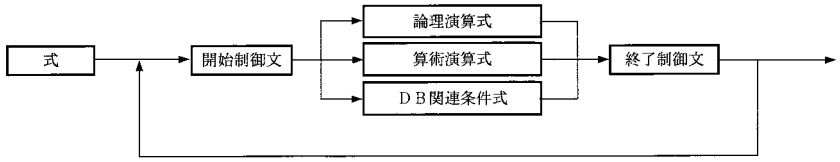
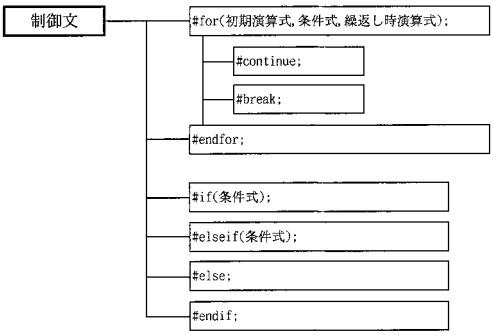

単一ﾃﾞｰﾀ本体
■ﾃﾞｰﾀ本体名
■[ｻｰﾊﾞ名]・[ＤＢ名]
■ＤＢ区分
■[ＤＢ検索条件式]
項目
■項目番号
■[表名・ﾃﾞｰﾀ項目名]
■[ＤＢ参照式]
■初期設定演算式
■表示条件式
■強調表示条件式
■矩形領域
■中塗り色・輪郭線色・立体感ﾀｲﾌﾟ
■横位置・縦位置・文字方向
■ﾌｫﾝﾄ・ｽﾀｲﾙ・ｻｲｽﾞ・文字色
■ｲﾒｰｼﾞ縮尺率・原点
参照
■参照ﾃﾞｰﾀ変換式
■ｴｺｰ型・ｲﾍﾞﾝﾄ演算式
更新
■入力必須ﾌﾗｸﾞ・入力可否条件式
■属性値選択子画面・引数
■入力ﾁｪｯｸ式・入力演算式
■更新ﾃﾞｰﾀ変換式
■ＤＢ更新条件式
連続ﾃﾞｰﾀ本体
■ﾃﾞｰﾀ本体名
■[ｻｰﾊﾞ名]・[ＤＢ名]
■ＤＢ区分
■ＤＢ検索条件式
■表名・ﾃﾞｰﾀ項目名・行並びﾙｰﾙ
■矩形領域・中塗り色・行数
集計
■集計条件式
■集計種別区分
■合計名称・小計名称
■矩形領域
■中塗り色・輪郭線色・立体感ﾀｲﾌﾟ
■横位置・縦位置・文字方向
■ﾌｫﾝﾄ・ｽﾀｲﾙ・ｻｲｽﾞ・文字色
項目
■項目番号
■[表名・ﾃﾞｰﾀ項目名]
■[ＤＢ参照式]
■初期設定演算式
■表示条件式
■強調表示条件式
■集計有無区分
■矩形領域
■中塗り色・輪郭線色・立体感ﾀｲﾌﾟ
■横位置・縦位置・文字方向
■ﾌｫﾝﾄ・ｽﾀｲﾙ・ｻｲｽﾞ・文字色
■ｲﾒｰｼﾞ縮尺率・原点
参照
■参照ﾃﾞｰﾀ変換式
■ｴｺｰ型・ｲﾍﾞﾝﾄ演算式
更新
■入力必須ﾌﾗｸﾞ・入力可否条件式
■属性値選択子帳票・引数
■入力ﾁｪｯｸ式・入力演算式
■更新ﾃﾞｰﾀ変換式
■ＤＢ更新条件式
操作本体
更新順（入力順）
■主項目番号
■従項目番号
省略
■項目番号

図 4　画面様式の構成


式
開始制御文
論理演算式
算術演算式
ＤＢ関連条件式
終了制御文


制御文
#for(初期演算式,条件式,繰返し時演算式);
#continue;
#break;
#endfor;
#if(条件式);
#elseif(条件式);
#else;
#endif;

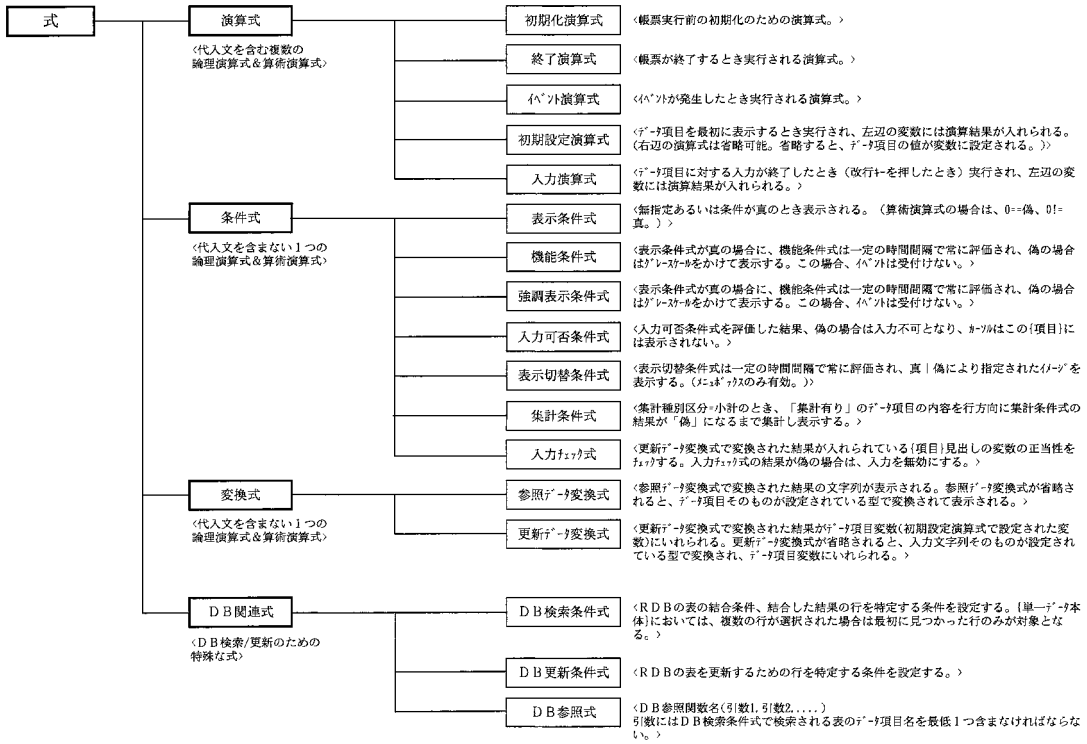

式
演算式
〈代入文を含む複数の
論理演算式＆算術演算式〉
初期化演算式
〈帳票実行前の初期化のための演算式。〉
終了演算式
〈帳票が終了するとき実行される演算式。〉
ｲﾍﾞﾝﾄ演算式
〈ｲﾍﾞﾝﾄが発生したとき実行される演算式。〉
初期設定演算式
〈ﾃﾞｰﾀ項目を最初に表示するとき実行され、左辺の変数には演算結果が入れられる。(右辺の演算式は省略可能。省略すると、ﾃﾞｰﾀ項目の値が変数に設定される。)〉
入力演算式
〈ﾃﾞｰﾀ項目に対する入力が終了したとき（改行ｷｰを押したとき）実行され、左辺の変数には演算結果が入れられる。〉
条件式
〈代入文を含まない１つの
論理演算式＆算術演算式〉
表示条件式
〈無指定あるいは条件が真のとき表示される。（算術演算式の場合は、0==偽、0!=真。）〉
機能条件式
〈表示条件式が真の場合に、機能条件式は一定の時間間隔で常に評価され、偽の場合はｸﾞﾚｰｽｹｰﾙをかけて表示する。この場合、ｲﾍﾞﾝﾄは受付けない。〉
強調表示条件式
〈表示条件式が真の場合に、機能条件式は一定の時間間隔で常に評価され、偽の場合はｸﾞﾚｰｽｹｰﾙをかけて表示する。この場合、ｲﾍﾞﾝﾄは受付けない。〉
入力可否条件式
〈入力可否条件式を評価した結果、偽の場合は入力不可となり、ｶｰｿﾙはこの{項目}には表示されない。〉
表示切替条件式
〈表示切替条件式は一定の時間間隔で常に評価され、真｜偽により指定されたｲﾒｰｼﾞを表示する。(ﾒﾆｭｰﾎﾞｯｸｽのみ有効。)〉
集計条件式
〈集計種別区分=小計のとき、「集計有り」のﾃﾞｰﾀ項目の内容を行方向に集計条件式の結果が「偽」になるまで集計し表示する。〉
入力ﾁｪｯｸ式
〈更新ﾃﾞｰﾀ変換式で変換された結果が入れられている{項目}見出しの変数の正当性をﾁｪｯｸする。入力ﾁｪｯｸ式の結果が偽の場合は、入力を無効にする。〉
変換式
〈代入文を含まない１つの
論理演算式＆算術演算式〉
参照ﾃﾞｰﾀ変換式
〈参照ﾃﾞｰﾀ変換式で変換された結果の文字列が表示される。参照ﾃﾞｰﾀ変換式が省略されると、ﾃﾞｰﾀ項目そのものが設定されている型で変換されて表示される。〉
更新ﾃﾞｰﾀ変換式
〈更新ﾃﾞｰﾀ変換式で変換された結果がﾃﾞｰﾀ項目変数(初期設定演算式で設定された変数)にいれられる。更新ﾃﾞｰﾀ変換式が省略されると、入力文字列そのものが設定されている型で変換され、ﾃﾞｰﾀ項目変数にいれられる。〉
ＤＢ関連式
〈ＤＢ検索/更新のための
特殊な式〉
ＤＢ検索条件式
〈ＲＤＢの表の結合条件、結合した結果の行を特定する条件を設定する。{単一ﾃﾞｰﾀ本体}においては、複数の行が選択された場合は最初に見つかった行のみが対象となる。〉
ＤＢ更新条件式
〈ＲＤＢの表を更新するための行を特定する条件を設定する。〉
ＤＢ参照式
〈ＤＢ参照関数名〈引数1,引数2,....〉
引数にはＤＢ検索条件式で検索される表のﾃﾞｰﾀ項目名を最低１つ含まなければならない。〉

図 5　つづく

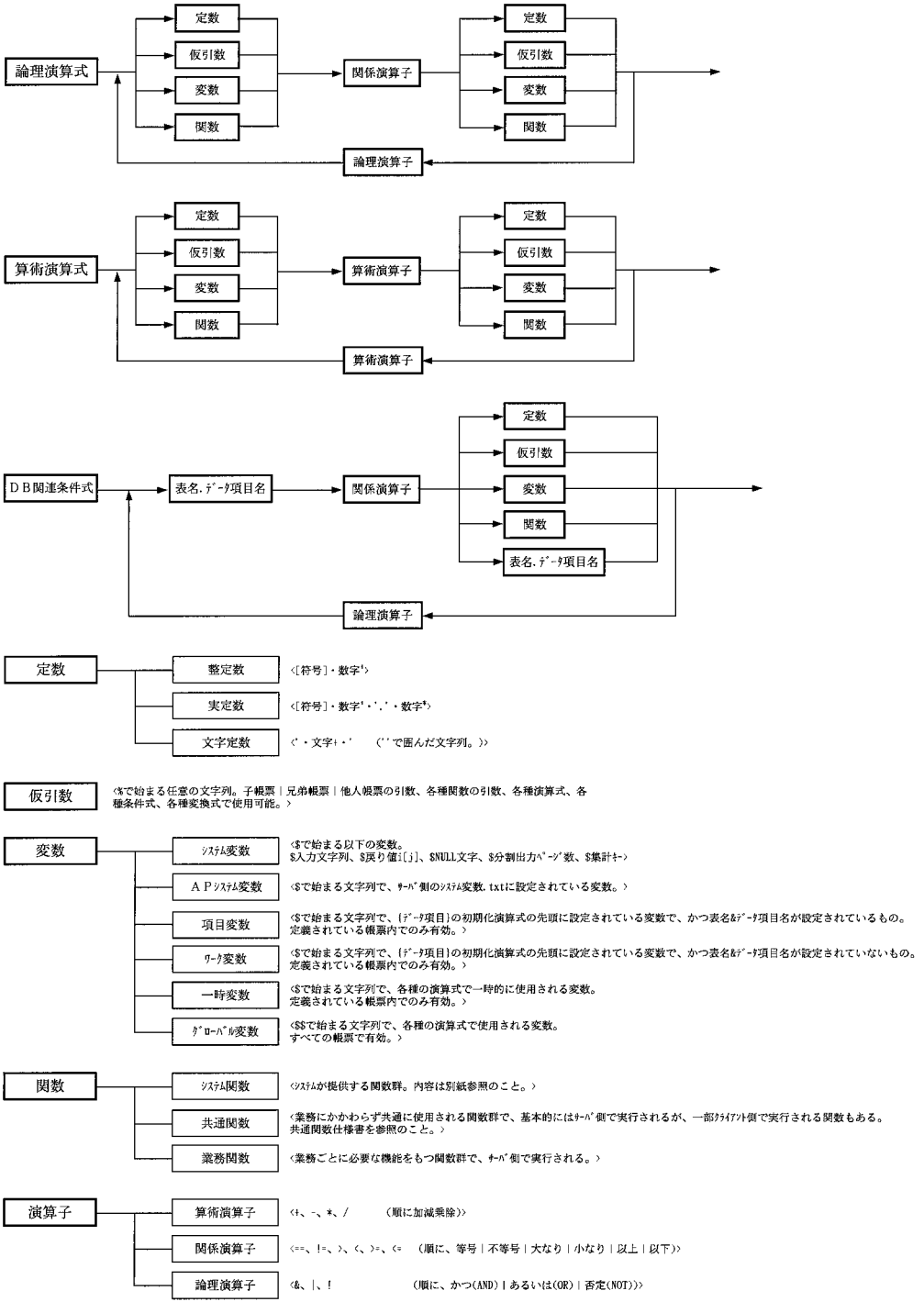

論理演算式
定数
仮引数
変数
関数
関係演算子
定数
仮引数
変数
関数
論理演算子
算術演算式
定数
仮引数
変数
関数
算術演算子
定数
仮引数
変数
関数
算術演算子
ＤＢ関連条件式
表名.ﾃﾞｰﾀ項目名
関係演算子
定数
仮引数
変数
関数
表名.ﾃﾞｰﾀ項目名
論理演算子
定数
整定数
〈[符号]・数字+〉
実定数
〈[符号]・数字+'.'・数字*〉
文字定数
〈'・文字+・'　('で囲んだ文字列。)〉
仮引数
〈%で始まる任意の文字列。子帳票｜兄弟帳票｜他人帳票の引数、各種関数の引数、各種演算式、各種条件式、各種変換式で使用可能。〉
変数
ｼｽﾃﾑ変数
〈$で始まる以下の変数。
$入力文字列、$戻り値[j]、$NULL文字、$分割出力ﾍﾟｰｼﾞ数、$集計ｷｰ〉
ＡＰｼｽﾃﾑ変数
〈$で始まる文字列で、ｻｰﾊﾞ側のｼｽﾃﾑ変数.txtに設定されている変数。〉
項目変数
〈$で始まる文字列で、[ﾃﾞｰﾀ項目]の初期化演算式の先頭に設定されている変数で、かつ表名&ﾃﾞｰﾀ項目名が設定されているもの。
定義されている帳票内でのみ有効。〉
ﾜｰｸ変数
〈$で始まる文字列で、[ﾃﾞｰﾀ項目]の初期化演算式の先頭に設定されている変数で、かつ表名&ﾃﾞｰﾀ項目名が設定されていないもの。
定義されている帳票内でのみ有効。〉
一時変数
〈$で始まる文字列で、各種の演算式で一時的に使用される変数。
定義されている帳票内でのみ有効。〉
ｸﾞﾛｰﾊﾞﾙ変数
〈$$で始まる文字列で、各種の演算式で使用される変数。
すべての帳票で有効。〉
関数
ｼｽﾃﾑ関数
〈ｼｽﾃﾑが提供する関数群。内容は別紙参照のこと。〉
共通関数
〈業務にかかわらず共通に使用される関数群で、基本的にはｻｰﾊﾞ側で実行されるが、一部ｸﾗｲｱﾝﾄ側で実行される関数もある。
共通関数仕様書を参照のこと。〉
業務関数
〈業務ごとに必要な機能をもつ関数群で、ｻｰﾊﾞ側で実行される。〉
演算子
算術演算子
〈+、-、*、/　　(順に加減乗除)〉
関係演算子
〈==、!=、>、<、>=、<=　(順に、等号｜不等号｜大なり｜小なり｜以上｜以下)〉
論理演算子
〈&、|、!　　(順に、かつ(AND)｜あるいは(OR)｜否定(NOT))〉

図 5　式の説明

図 6　データ項目の入力例

図 7　出張届入力例

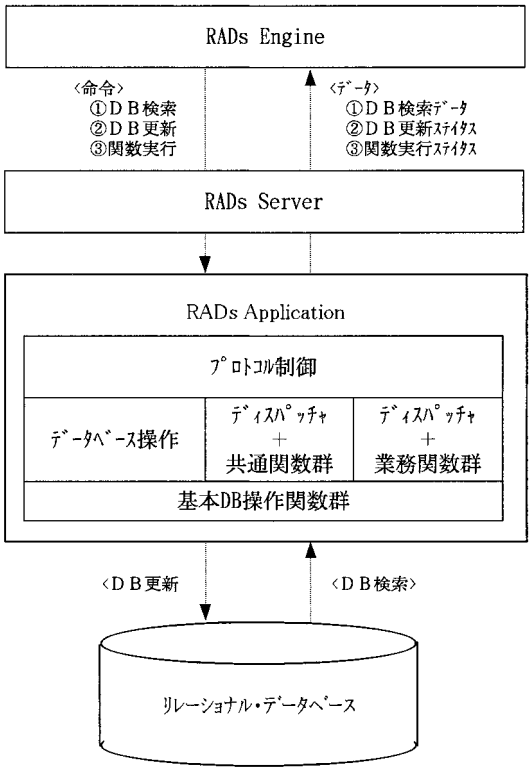


図 8　RADs Application の構成

を自分の更新 DB に保存する．その後で拠点サーバの DB 集信はそれらのトランザクションを転送済みにする．この処理は拠点サーバの数分並列に実行される．

2)　DB 一括更新は更新 DB のトランザクションを表単位で読み込み,「マスタ DB」を更新するための条件が満たされているもの（例えば，届出の場合は所属長が承認し更新可能年月日に到達したトランザクション）に対しマスタ DB を更新し，更新結果をあて先付で「配信 DB 表」に書き込む．通常，あて先はトランザクションの発生した拠点サーバであるが，表の配信タイプが「全拠点サーバに配信」となっている場合はすべての拠点サーバあてに更新結果を書き込む．その後で更新 DB のトランザクションを削除する．

3)　拠点サーバの DB 配信は人事サーバの DB 配信に更新結果の送信を要求する．人事サーバの DB 配信は配信 DB 表より該当拠点分の更新結果を読み込み，拠点サーバへ送信する．拠点サーバの DB 配信は更新結果でマスタ DB を更新し，更新 DB に残っている対応するトランザクションを削除する．その後で人事サーバの DB 配信は配信 DB 表の更新結果を削除する．

以上の処理がジョブ運用管理により一定時間間隔で実行される（図 9）．

グローバル非同期更新は拠点サーバ主導型である．一部の拠点サーバが停止していてもシステムは正常に作動する．停止していた拠点サーバは復旧次第未転送のトランザクションを人事サーバに送り，配信 DB 表の更新結果を人事サーバに要求し受け取る．

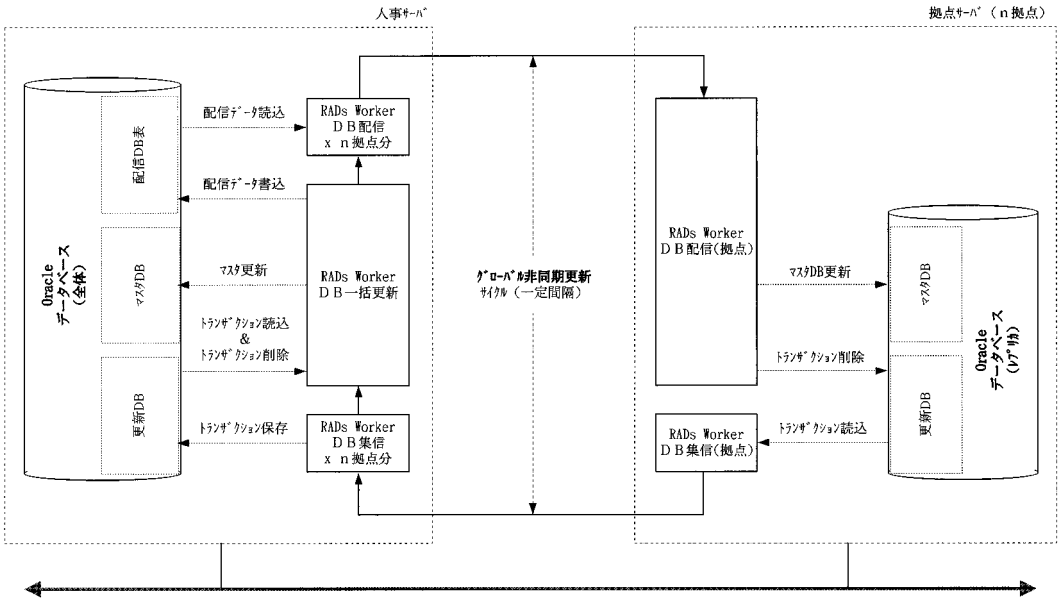


図 9　グローバル非同期更新

一方，業務バッチは夜間実行され，直接マスタ DB を更新する．業務バッチは一括更新処理であり，更新 DB にトランザクションを書き込まない．「マスタ DB 配信」は主に業務バッチで更新されたマスタ DB を拠点サーバに配信する．

6.　DB バックアップ

HSS システムでは以下の二つの方法でデータベースをバックアップして障害に備えている．

・アーカイブ・ログによるオンライン・バックアップ

・毎日の物理データベース・ファイルの保存によるコールド・バックアップ

人事サーバのデータベースは上記の二つの方法でバックアップ・サーバにバックアップされるが，拠点サーバのデータベースはバックアップしていない．なぜなら，拠点サーバに障害が発生しデータベースが破壊されても，人事サーバのデータベースより再生できるからである．

6.1　オンライン・バックアップ

人事サーバのデータベースは，バックアップ・サーバによりオンライン・バックアップされる（図 10）．バックアップ・サーバのデータベースは人事サーバのデータベースと同じ構成であるため，Oracle の 2 フェーズ・コミットによるバックアップも試行したが，効率が大幅に低下するため断念した．オンライン・バックアップはアーカイブ・ログによる方法が最善である[4]．

グローバル非同期更新のサイクル中に DB 一括更新の前後 2 回，人事サーバのアーカイブ・ログ・ファイルはバックアップ・サーバへコピーされ保存される．この方法は完全なバックアップではない．以下の問題を含んでいる．

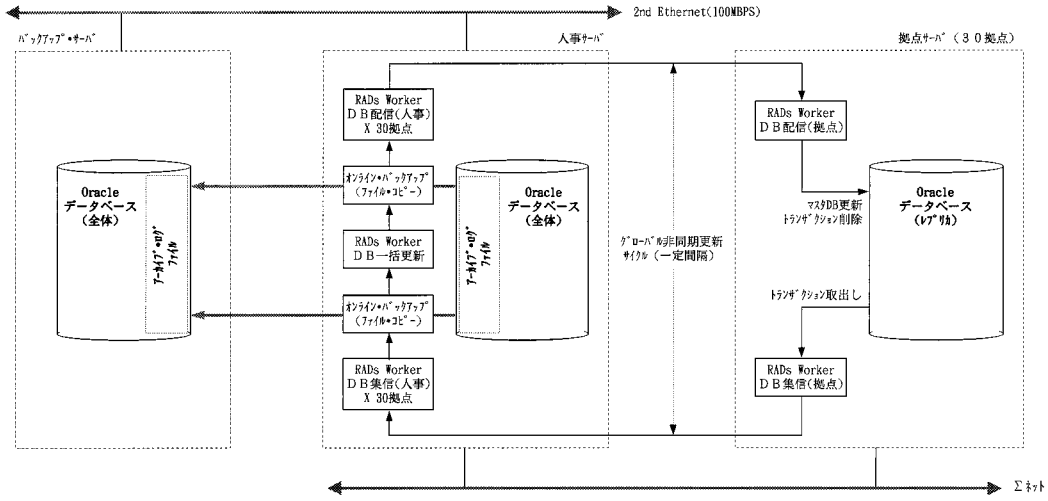


図 10　オンライン・バックアップ

・拠点サーバのデータベースが破壊された場合：更新 DB に貯えられている未転送のトランザクションは消失し回復できない．人事サーバのログ情報により，ログアウトしていない使用者と最近の DB 集信後にログアウトした使用者にデータベース復旧後データの確認と再入力を要請する必要がある．

・人事サーバのデータベースが破壊された場合：最近のオンライン・バックアップ以降の人事サーバのトランザクションは消失し回復できない．ログ情報により，ログアウトしていない使用者と最近のオンライン・バックアップ以降にログアウトした使用者にデータベース復旧後データの確認と再入力を要請する必要がある．

しかしこれらの問題は人事業務では運用でカバーできる範囲であり，かつデータベースが破壊されることはまれである（Oracle の致命的なエラーかディスクが壊れるときであるが，ディスクは最も信頼性の高い RAID 5 で構成している）．そのため投資コストの削減と応答性を優先した．

6.2　コールド・バックアップ

毎晩オンライン停止後，人事サーバのデータベース・ファイルは 100 MBPS の 2 nd Ethernet 経由でバックアップ・サーバへコピーされ，DLT オートチェンジャのテープにバックアップされる．この時点で人事サーバのデータベースとバックアップ・サーバのデータベースは同一になる．

破壊された人事サーバのデータベースは，バックアップ・サーバのデータベース・ファイルとアーカイブ・ログ・ファイルを人事サーバにコピーし，Oracle 7 を立ち上げることで復旧される．

7.　障　害　対　策

障害には次の四つが考えられる．

・使用者の PC クライアントの障害

・ネットワークの障害
・人事サーバの障害
・拠点サーバの障害

使用者の PC クライアントの障害とネットワークの障害の対策は全社のシステム運用の問題であるため，ここでは人事サーバの障害対策と拠点サーバの障害対策について述べる．

7.1 人事サーバの障害対策

バックアップ・サーバは人事サーバのコールド・スタンバイ機である．人事サーバに障害が発生し復旧に長時間を要する場合，運用管理者の判断でバックアップ・サーバを人事サーバに切り換えることができる．

人事サーバの電源をオフし，バックアップ・サーバの IP アドレスに人事サーバ IP アドレスを設定して再立ち上げすれば，バックアップ・サーバは人事サーバに切り換わる．ただし，人事サーバが停止中でも拠点サーバは正常に稼働できるため，人事サーバと接続しなければならない特殊な業務を除いて運用は継続できる．

7.2 拠点サーバの障害対策

拠点サーバに障害が発生しクライアントから接続不能になった場合，運用管理者の判断で人事サーバにその拠点サーバの代替をさせることができる．

人事サーバで拠点サーバ用の RADs Server と RADs Application を稼働させ，クライアントを再立ち上げすると人事メインが人事サーバと自動的に接続してくれる．ただしこの場合に，復旧後，拠点サーバのデータベースが正常だった場合は，更新 DB の未転送のトランザクションが人事サーバに送られてしまうことに注意しなければならない．

8. 効 率 改 善

データベース・システムにおいて最も効果的な効率改善の方法はメモリの増設である．メモリが十分でない状況で Oracle の共有プール領域やデータベース・バッファ，アプリケーションのデータ領域をむやみに増やすと当然のことながらスワップが発生し，ディスク I/O によりスループットは大幅に低下する．その結果 CPU の稼働率は 30% に満たない状況になり，いくら高速の CPU を使っても 3 分の 1 の能力しか引き出せないことになる．

また，PC サーバのディスク・システムは汎用機のディスク・システムに比べて能力が大幅に劣っていることを認識しておくべきである．

8.1 アプリケーション チューニング

表 2 は業務バッチの効率改善結果である．以下にチューニングの手順を述べる．

・SQL 文 where 句の結合・限定条件は，表の結合前に対象を十分限定し，表の結合は対象の少ない表から順にする（表 2–No. 10）．
・Index がはられているかどうか確認し，Index が対象の SQL 文 where 句の限定条件と一致しているかどうかを確認する（表 2–No. 3）．Index はむやみに設定すると Insert の効率を著しく低下させる．有効な Index のみに限定しなければならない．

表 2　アプリケーションの効率改善

No.	業務	種別	改善前	改善後	Up率	改善内容
1	勤務ｶﾚﾝﾀﾞ作成	C	0:57:43	0:38:41	67%	①勤務ｶﾚﾝﾀﾞ作成の前後にDropIndexとCreateIndexをはさむことにより、12,000*2回の行挿入の効率改善。
2	勤務月次集計	C	1:46:45	0:33:53	32%	①12,000回のﾙｰﾌﾟの中で個別に削除するのではなくて、最初に全件削除を行う。→ Delete*48,000回を削減。
3	勤務月次確定	C	2:30:00	0:16:18	11%	①検索している更新DBの4表に Index を作成する。 ②12,000回のﾙｰﾌﾟの中で個別に削除するのではなくて、最後に全件削除を行う。→ Delete*48,000回を削減。
4	本実労働時間計算	C	0:56:00	0:47:13	84%	①勤務ｶﾚﾝﾀﾞ作成であらかじめ本実労働時間に値を設定しておく。(更新時にﾚｺｰﾄﾞ長が増えないようにする。)
5	給与_控除課税計算	C	1:36:52	1:05:22	68%	給与_控除課税.xlsを参照のこと。 Select*132,000回、Insert*24,000回、Delete*120,000回削減。
6	賞与_実績計算	C	0:45:00	0:27:43	62%	賞与_実績計算.xlsを参照のこと。 Select*240,000回削減。
7	賞与_支給計算	C	0:17:23	0:11:59	69%	賞与_支給計算.xlsを参照のこと。 Select*12,000回、Update*12,000回削減。
8	賞与_支払計算	C	1:00:00	0:28:26	47%	賞与_支払計算.xlsを参照のこと。 Select*444,000回削減。
9	退職金_引当金計算	C	0:35:00	0:20:00	57%	①必要なﾃﾞｰﾀはあらかじめ表結合で検索。②ﾃｰﾌﾞﾙはあらかじめすべて読込み自前で検索。 以上の対策で、Select*60,000回削減。
10	給与_控除金情報	C	0:20:00	0:03:00	15%	給与_控除金情報.xlsを参照のこと。 ①表の結合前に対象を絞り込む(検索条件式の改良)。②一時表へのInsertを一回で行う。 ③複数の検索を一つにまとめる。

注) Up率 = (改善後／改善前) * 100

表 3　人事サーバ測定結果

No	調査項目	初期設定パラメータ名（現在値）	測定開始（15:00）	測定終了（16:00）	差分（1時間）	考察
1	ﾗｲﾌﾞﾗﾘ･ｷｬｯｼｭ（SQL領域）のﾋｯﾄ率	SHARED_POOL_SIZE(64,000,000)	0.597	0.603	0.674	ﾋｯﾄ率<80%であるため改善の必要あり。社員番号のみが異なるSQL文が多いため、社員番号を含めAPの変数をﾊﾞｲﾝﾄﾞ変数にすることにより、①SQL領域のﾋｯﾄ率向上、②SQL文解釈のｽﾙｰﾌﾟｯﾄ向上が期待できる。
2	ﾃﾞｰﾀ･ｷｬｯｼｭのﾋｯﾄ率	DB_BLOCK_BUFFERS(60,000*2,048)	0.990	0.990	0.994	ﾋｯﾄ率>80%であるため、問題なし。
3	ﾊﾞｯﾌｧ ﾋﾞｼﾞｰ待ち率	FREELIST(default)	0.004	0.003	0.000	ﾋﾞｼﾞｰ待ち率<4%であるため、問題なし。（ﾋﾞｼﾞｰ待ちはほとんど発生していない。）
4	ﾒﾓﾘでのｿｰﾄ率（ﾒﾓﾘ/（ﾒﾓﾘ+ﾃﾞｨｽｸ））	SORT_AREA_SIZE(2,000,000)	1.000	1.000	1.000	ﾒﾓﾘでのｿｰﾄ率>90%であるため、問題なし。（ほとんどﾒﾓﾘでのｿｰﾄ。）
5	ｴﾝｷｭ待ち	ENQUEUE_RESOURCES(default)	2,961	3,186	225	ENQUEUE_RESOURCESを増やす必要あり。
6	Redo Allocation ﾗｯﾁ	LOG_SMALL_ENTRY_MAX_SIZE(default)	0.995	0.995	0.999	ﾋｯﾄ率>85%以上であるため、問題なし。
7	行連鎖		244,411	254,969	10,558	10,558回／時間は多い。Indexと更新されるTable（勤務関連）のPCTFREEの値を大きくし、Export/Importで再作成すれば行連鎖は減る。 しかしながら、PCTFREEの値を大きくすることによる弊害（ﾃﾞｨｽｸ容量の増大、ﾃﾞｰﾀ･ｷｬｯｼｭのﾋｯﾄ率の低下）も考えられるため、とりあえずはExport/Importによる再作成のみとし様子を見る。
8	ｱﾌﾟﾘｹｰｼｮﾝの効率（全件検索の割合）		0.006	0.006	0.002	全件検索の割合が少なく、問題なし。
9	表領域とﾌｧｲﾙI/O	<表は省略>				特にMST（TOTO_MST3.ora）が多い。MSTとUPDを別ﾃﾞｨｽｸ･ﾄﾞﾗｲﾌﾞに、TableとIndexを分離し別ﾃﾞｨｽｸ･ﾄﾞﾗｲﾌﾞにする必要があるが、現在すべてのﾃﾞｨｽｸをRAID5一本で組んでいるため、OSのﾊﾞｰｼﾞｮﾝｱｯﾌﾟあるいはﾃﾞｨｽｸ増設時に再編成を行い分離する。
10	ﾛｰﾙﾊﾞｯｸ･ｾｸﾞﾒﾝﾄ	<表は省略>				すべてのﾛｰﾙﾊﾞｯｸ･ｾｸﾞﾒﾝﾄの待ち率<4%であるため、問題なし。
11	ｴｸｽﾃﾝﾄの状況	<表は省略>				ｴｸｽﾃﾝﾄ数>5の表が206もある。表のｻｲｽﾞを再見積りし、INITIALﾊﾟﾗﾒｰﾀを増やして、Export/Importにより再作成が必要。
12	表領域の空き状況	<表は省略>				空き領域は十分で、現在のところ特に問題なし。

・大量の Insert を行う場合は，処理の前後に DropIndex と CreateIndex をはさみ，Insert 中に Index の作成をさせないようにする（表 2–No. 1）.
・更新処理でほとんどの行に対して値が設定されるような NUMBER 型の列は，NULL 値を許さないかもしくは最初にダミー値を設定しておく（表 2–No. 4）.
・プログラムを見直して SQL の発行回数を減らす（表 2–No. 2）.
・データ領域を拡張しすべてのデータを一度に検索して，あとは自前でメモリ検索し，Oracle の Select を使わないようにする．同時に，Insert もまとめて一回で行うようにする（表 2–No. 5，6，7，8，9）．ただし，スワップが発生しない程度のデータ領域拡張に留めなければならない.

8.2 Oracle データベース　チューニング

表 3 は現在の人事サーバの状況である．これらのパフォーマンス情報は Oracle が提供するスクリプト UTLBstat/UTLEstat で収集することができる[4].

表 3 から人事サーバの状況は概ね良好であることがわかるが，もっと効率を上げるには以下の 3 点の改良が必要である.

・行連鎖の削減.
・エクステントの削減.
・ライブラリ・キャッシュのヒット率向上.

HSS システムで発行される SQL は社員番号が異なることを除いて同じ SQL が多い．これは逆に社員番号が異なるため Oracle にとってはすべて異なる SQL として処理されることを意味する．そのため，SHARED_POOL_SIZE を大きくしても時間の経過とともに共有プール領域は使い果たされていき，ライブラリ・キャッシュのヒット率は低下していく.

もし社員番号を Oracle のバインド変数にすれば，社員番号が異なっても同じ SQL として扱われ，「SQL の翻訳と実行計画の作成」ステップが省略でき全体のスループットの向上につながるはずである．ただし，SQL の翻訳と実行計画の作成ステップのコストが未確認のため，どれだけの効果があるかは検証できていない.

人事のデータベースは日々成長している（データが日々増加している）．半年先に今回と同じ良好な状況である保証はない．データベース管理者は最低でも 1 回/3 か月，Import/Export によるデータベースの再編成と，パフォーマンス情報の収集/分析，そしてデータベースの改善に努めなければならない.

9. お　わ　り　に

1997 年 4 月から本番を開始した．1997 年 8 月に人事部が全社員を対象に実施した HSS システムのアンケートによれば，「HSS システムを使うことにより便利になった，または HSS システムが役立っている」と答えた社員が，一般社員では 83%，庶務担当では 86%，課長では 75%，部長では 76% に達している.

また届出件数も 10 万件/月で，従業員一人あたり平均 10 回/月 HSS システムを利用していることになる（表 4）.

RADs という EUC & RAD ツール＋統合インフラにより，オール PC による大規模 CSS&分散データベース・システムが実現できた．応答性も当初の目標を十分に

表 4　月間届出集計表　(2 月度)

順位	届出内容	件数
1	勤務報告	83390
2	出張届	3187
3	ﾘｿﾞｰﾄﾊｳｽ予約	2463
4	ﾕﾆﾌｫｰﾑ申請	923
5	源泉徴収票発行	800
6	家族扶養異動届	461
7	社内預金	408
8	諸会陶友会	329
9	財形受付	299
10	銀行口座	269
11	貸付	203
12	定期券代補助	201
13	住所変更	169
14	社宅寮	159
15	ﾊﾟｽﾜｰﾄﾞ変更	148
16	通勤費補助	139
17	住宅費補助	119
18	退職願	100
19	個人積立年金	59
	その他	3027
	合計	96853

クリアした.

HSS システムの開発にあたり，RADs の開発提案を承諾していただき，また今回の本誌への発表を快く承諾していただいた TOTO の人事部と情報システム部に心より感謝する．また，現行人事システムからのデータ移行とホスト側の開発を全面的に担当していただいた情報システム部にお礼を申しあげる.

RADs は当初から汎用性を重視して設計されている．今後はいろいろな分野に適用されていくことを期待している.

参考文献
［1］ 星野友彦,“ 大規模 C/S は 3 層でつくる ”, 日経コンピュータ, 1995. 12. 25
［2］ 林　衛,「DOA と RAD のためのシステム分析・設計技法」, ソフト・リサーチ・センター
［3］ 安斉紘司,「オープンシステム・メソドロジ」, 工業調査会, 1994. 05
［4］ Michael J. Corey 他著, 小幡一郎訳,「Oracle データベースチューニング」, 翔泳社
［5］ 小幡一郎,「Oracle 7 プロフェッショナルテクニック」, ソフト・リサーチ・センター

執筆者紹介　篠　田　博　水（Hiromi Shinoda）
1947 年生．1970 年法政大学工学部経営工学科卒業．同年日本ユニシス(株)入社．以来，グラフィックスと CAD の開発を経て人事情報システムの開発に従事．1998 年 1 月に独立のため退社．現在，(株)ネットワーク・システムズ代表取締役.