OUKA
Osaka University Knowledge Archive

| | |
|---|---|
| Title | 高度計算機ネットワーク環境におけるデータベースシステムに関する研究 |
| Author(s) | 春本, 要 |
| Citation | 大阪大学, 1998, 博士論文 |
| Version Type | VoR |
| URL | https://doi.org/10.11501/3144178 |
| rights | |
| Note | |

*Osaka University Knowledge Archive : OUKA*

https://ir.library.osaka-u.ac.jp/

Osaka University

2742-4

# 高度計算機ネットワーク環境における データベースシステムに関する研究

1997年12月

春本　要

# 高度計算機ネットワーク環境における
# データベースシステムに関する研究

1997年12月

春本　要

# 内容梗概

本論文は，筆者が平成4年4月から平成6年3月まで大阪大学大学院基礎工学研究科情報ネットワーク学講座において，引き続いて平成6年4月以降大阪大学工学部情報システム工学科知識システム工学講座（現在，大阪大学大学院工学研究科情報システム工学専攻社会情報システム工学講座情報ベース工学領域）において行なった，高度計算機ネットワーク環境におけるデータベースシステム関する研究成果をまとめたものである．

データベース技術は，計算機システムの中で永続的データの管理を行なう最も重要な基盤技術であり，オンライン・トランザクション処理をはじめとしてさまざまな分野で幅広く利用されてきた．近年，計算機ネットワーク技術の発展，および，インターネットに代表される世界規模のネットワークの整備により，データベース技術の応用分野はますます広がりつつある．

データベースシステムは，大きく分けて，データベース自身の管理およびデータベースへのアクセス手段を提供するデータベース管理システム，データベースを使用するアプリケーションプログラム，データベースをアクセスするエンドユーザから構成される．本論文では，この三つの構成要素の各々に対応した観点から，近年の計算機ネットワーク環境の変化に適応したデータベースシステム構築技術について論じる．まず，エンドユーザの観点から，グループウェアシステムへのデータベース技術の適用について議論する．グループウェアシステムは，計算機が低価格化してきたこと，および，LAN (Local Area Network) が容易に構築できるようになってきたことから急速に普及しつつあり，複数の作業者の協調作業を支援する際の共有データの管理技術は非常に重要である．次に，データベースにアクセスするアプリケーションプログラムの観点から，通信アプリケーションからのデータベースの利用を容易にするためのデータベースシステムの構成方法について論じる．通信アプリケーションでは，異機種間の通信を行なうために共通のデータ表現方法が用いられることが多いため，特にデータ表現とプログラミング言語を中心に議論する．最後に，データベースシステムの中核をなすデータベース管理システムの観点から，

ATM (Asynchronous Transfer Mode, 非同期転送モード) ネットワークに代表される広帯域ネットワークを介した分散データベースシステムにおけるトランザクション処理手法について論じる．広帯域ネットワークは従来のネットワークと異なる特性をもっており，従来の分散データベースシステム技術をそのまま適用したのでは不十分なことを指摘し，その改善方法について議論する．

本論文は，全5章から構成される．第1章で序論を述べ，第2章ではグループウェアシステムとして協調作業支援環境を取り上げ，その基盤となるデータベース管理技術について議論する．第3章では通信アプリケーションのためのデータベース管理技術について述べ，第4章では広帯域ネットワークを前提とした分散データベース処理技術について論じる．第5章で以上の章の結論をまとめる．各章の内容をより詳しく記すと次のようになる．

第1章の序論では，データベースシステムの重要性を述べた上で本論文の目的を明確にする．さらに，本論文の論点とその構成について述べる．

第2章では，協調作業支援環境における長時間トランザクションの処理の問題について述べ，協調作業に適したデータベースモデルおよびトランザクションモデルを提案する．また，共同作業者の作業の非同期性などを考慮したトランザクションの実行制御モデルを提案する．

第3章では，データベース機能を必要とする通信アプリケーションの開発における問題点について言及し，その解決策としてネットワーク上でのデータ表現に着目したデータベースシステムの設計および実装手法について論じる．特に，アプリケーション開発に用いるプログラミング言語，および，実際のデータ格納部の実現方法について論じる．

第4章では，近年の計算機ネットワークの広帯域化に注目し，計算機ネットワーク上で動的にデータベースを移動させることによって分散データベース処理の高速化が可能であることを示す．

第5章では，本論文の内容を要約し，本研究結果の応用分野について示す．最後に今後の研究課題について述べる．

# 関連発表論文

## 1. 学術論文誌掲載論文

(1) 春本 要, 八幡 孝, 西尾章治郎: "協調作業支援のためのデータ管理モデル," 電子情報通信学会論文誌 D-I, Vol.J79-D-I, No.5, pp.271–279 (1996 年 5 月).

(2) 春本 要, 塚本昌彦, 西尾章治郎: "OODBMS を用いた ASN.1 データベースの実現とその評価," 電子情報通信学会論文誌 D-I, Vol.J79-D-I, No.11, pp.975–983 (1996 年 11 月).

(3) 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "ATM ネットワークにおけるデータベース移動のためのデータベース位置管理手法," 電子情報通信学会論文誌 D-I, Vol.J80-D-I, No.2, pp.137–145 (1997 年 2 月).

(4) 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "データベース移動を用いた ATM ネットワークにおけるトランザクション処理," 電子情報通信学会論文誌 D-I, Vol.J80-D-I, No.6, pp.505–513 (1997 年 6 月).

(5) Budiarto, Harumoto, K., Tsukamoto, M., Nishio, S., and Takine, T.: "On Strategies for Allocating Replicas of Mobile Databases," IEICE Transactions on Information and Systems (1998 年, 掲載予定).

## 2. 国際会議会議録掲載論文

(1) Harumoto, K., Tsukamoto, M., and Nishio, S.: "A Database System Based on ASN.1: Its System Architecture and Database Programming Language," *Proc. International Symposium on Advanced Database Technologies and Their Integration (ADTI'94)*, pp.160–167, Nara, Japan (1994 年 10 月).

(2) Budiarto, Harumoto, K., Tsukamoto, M., Nishio, S., and Takine, T.: "Replica Allocation Strategies in Mobile Computing Environment," *Proc. International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC'96)*, Volume I, pp.99–102, Seoul, Korea (1996 年 7 月).

(3) Harumoto, K., Tsukamoto, M., and Nishio, S.: "Design and Implementation of an Information Repository for ASN.1-based Distributed Applications," *Proc. International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC'96)*, Volume II, pp.681–684, Seoul, Korea (1996 年 7 月).

(4) Budiarto, Harumoto, K., Tsukamoto, M., and Nishio, S.: "On Transaction Management for Mobile Clients and Servers," *Proc. International Workshop on Multi-Dimensional Mobile Communications (MDMC'96)*, pp.402–406, Seoul, Korea (1996 年 7 月).

(5) Hara, T., Harumoto, K., Tsukamoto, M., and Nishio, S.: "Database Migration for Transaction Processing in ATM Networks," *Proc. 11th International Conference on Information Networking (ICOIN-11)*, Vol.1, pp.1B-4.1–1B-4.10, Taipei, Taiwan, R.O.C. (1997 年 1 月).

(6) Hara, T., Harumoto, K., Tsukamoto, M., and Nishio, S.: "Location Management Methods of Migratory Data Resources in ATM Networks," *Proc. ACM Symposium on Applied Computing (ACM SAC'97)*, pp.123–130, San Jose, California, U.S.A. (1997 年 2 月).

(7) Budiarto, Harumoto, K., Tsukamoto, M., and Nishio, S.: "Position Locking: Handling Location Dependent Queries in Mobile Computing Environment," *Proc. International Conference on Worldwide Computing and Its Applications (WWCA'97)*, pp.C-3-2-1–C-3-2-15, Tsukuba, Japan (1997 年 3 月).

(8) Hara, T., Harumoto, K., Tsukamoto, M., Nishio, S., and Okui, J.: "Main Memory Database for Supporting Database Migration," *Proc. 1997 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'97)*, Volume 1, pp.231–234, Victoria, B.C., Canada (1997年8月).

(9) Budiarto, Harumoto, K., Tsukamoto, M., and Nishio, S.: "Relocation Decision Policies for Mobile Replicas," *Proc. International Symposium on Digital Media Information Base (DMIB'97)*, Nara, Japan (1997年11月, 掲載予定).

(10) Hara, T., Harumoto, K., Tsukamoto, M., and Nishio, S.: "DB-MAN: A Distributed Database System based on Database Migration in ATM Networks," *Proc. 14th International Conference on Data Engineering (ICDE'98)*, Orlando, Florida, U.S.A. (1998年2月, 掲載予定).

## 3. 国内重要会議会議録掲載論文

(1) 澤村卓哉, 日比野哲也, 春本 要, 塚本昌彦, 西尾章治郎: "通信アプリケーション記述のためのASN.1データベースプログラミング言語の拡張," 電子情報通信学会第6回データ工学ワークショップ (DEWS'95) 論文集, pp.175–182 (1995年3月).

(2) 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "データベース移動を用いた分散データベースシステムの並行処理制御について," 情報処理学会マルチメディア通信と分散処理ワークショップ論文集, pp.179–186 (1996年10月).

## 4. 国内研究会，全国大会発表論文

(1) 春本 要, 仲秋 朗, 塚本昌彦, 西尾章治郎, 宮原秀夫: "抽象構文記法1 (ASN.1) に基づくデータベースシステム," 情報処理学会第97回データベースシステム研究会報告 (94-DBS-97), pp.41–50 (1994年3月).

(2) 仲秋 朗, 春本 要, 齋藤 淳, 塚本昌彦, 西尾章治郎: "OODBMS を用いた ASN.1 データベースの実現," 情報処理学会第 65 回マルチメディア通信と分散処理研究会報告 (94-DPS-65), pp.151–156 (1994 年 5 月).

(3) 春本 要, 塚本昌彦, 西尾章治郎: "ASN.1 データベースプログラミング言語," 情報処理学会第 99 回データベースシステム研究会報告 (94-DBS-99), pp.249–256 (1994 年 7 月).

(4) 八幡 孝, 春本 要, 西尾章治郎: "汎用協調作業支援システム COCOA の設計およびその応用," 日本ソフトウェア科学会第 11 回大会論文集, C5-2, pp.209–212 (1994 年 11 月).

(5) 齋藤 淳, 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "ASN.1 データベースシステムにおけるデータ格納方式," 情報処理学会第 101 回データベースシステム研究会報告 (95-DBS-101), pp.17–24 (1995 年 1 月).

(6) 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "ASN.1 データベースシステムにおけるインデックス機構の比較," 情報処理学会第 104 回データベースシステム研究会報告 (95-DBS-104), pp.17–24 (1995 年 7 月).

(7) Budiarto, 春本 要, 塚本昌彦, 西尾章治郎: "移動体計算環境におけるトランザクション制御方式," 電子情報通信学会データ工学研究会報告, DE95-51, pp.9–16 (1995 年 10 月).

(8) 西尾章治郎, 塚本昌彦, 春本 要, 下條真司: "広帯域ネットワークにおけるデータベースシステムの可能性," 電子情報通信学会データ工学研究会報告, DE95-56, pp.49–56 (1995 年 10 月).

(9) 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "ATM 環境におけるデータベース移動に基づくトランザクション処理手法," 情報処理学会第 106 回データベースシステム研究会報告 (96-DBS-106), pp.49–56 (1996 年 1 月).

(10) 庄司加奈子, 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “広帯域ネットワークにおけるデータベース移動の実現について,” 電子情報通信学会データ工学研究会報告, DE96-10, pp.55–60 (1996 年 5 月).

(11) 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “ATM 環境内で移動するデータベースの位置管理について,” 情報処理学会第 109 回データベースシステム研究会報告 (96-DBS-109), pp.111–116 (1996 年 7 月).

(12) 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “データベース移動に基づく分散データベースシステム DB-MAN の性能評価,” 情報処理学会第 54 回全国大会講演論文集 (3), 1R-8, pp.3-243–3-244 (1997 年 3 月).

(13) 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “データベース移動に基づく動的複製配置法,” 電子情報通信学会データ工学研究会報告, DE97-23, pp.107–112 (1997 年 7 月).

(14) 秋山豊和, 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “データベース移動を用いた分散データベースシステムの設計と実装,” 情報処理学会第 55 回全国大会講演論文集 (3), 4AC-11, pp.3-469–3-470 (1997 年 9 月).

# 目 次


**1 序論** **1**

1.1 研究の背景 . . . . . 1

1.2 データベースシステムの構成 . . . . . 2

1.3 本論文の概要 . . . . . 4

1.3.1 協調作業環境のためのデータベース管理手法 . . . . . 5

1.3.2 通信アプリケーションのためのデータベースシステムの構築 . 7

1.3.3 広帯域ネットワークにおける分散データベース処理手法 . . . 9

1.4 本論文の構成 . . . . . 10

**2 協調作業支援のためのデータベース管理手法** **13**

2.1 まえがき . . . . . 14

2.2 関連研究 . . . . . 16

2.3 本論文のアプローチ . . . . . 18

2.4 協調作業データベースとジョブ . . . . . 19

2.4.1 領域一貫性制約に基づくデータベースモデル . . . . . 20

2.4.2 ジョブモデル . . . . . 21

2.5 ジョブの実行制御モデル . . . . . 23

2.5.1 ジョブの協調実行の方法 . . . . . 24

2.5.2 領域予約操作による予告通知 . . . . . 26

2.5.3 領域一貫性制約の考慮による通知対象の拡大 . . . . . 27

2.5.4 特定の領域に対する操作の取り消し . . . . . 29

2.6 COCOA System の実装 . . . . . 30
2.6.1 ジョブマネージャ . . . . . 32
2.6.2 協調実行マネージャ . . . . . 33
2.6.3 領域管理機構 . . . . . 34
2.6.4 制約検査機構 . . . . . 35
2.6.5 実際の協調作業への応用例 . . . . . 36
2.7 むすび . . . . . 39

**3 通信アプリケーションのためのデータベースシステム 41**
3.1 まえがき . . . . . 42
3.2 ASN.1 の概要 . . . . . 43
3.2.1 ASN.1 の利用目的 . . . . . 44
3.2.2 ASN.1 によるデータ構造定義 . . . . . 44
3.2.3 ASN.1 の符号化規則 . . . . . 49
3.3 ASN.1 データベースシステムの要件 . . . . . 50
3.4 ASN.1/PL: ASN.1 データベースプログラミング言語 . . . . . 53
3.4.1 ASN.1/PL の設計 . . . . . 53
3.4.2 ASN.1/PL の構文要素 . . . . . 54
3.4.3 プログラム例 . . . . . 58
3.4.4 ASN.1/PL 処理系 . . . . . 58
3.5 OODBMS を用いた ASN.1 データベース . . . . . 62
3.5.1 ASN.1 データベースの実現のための基本方針 . . . . . 63
3.5.2 OODBMS を対象とした ASN.1 コンパイラの実現方法 . . . . 66
3.5.3 DIB の構築によるシステム評価 . . . . . 71
3.6 ASN.1 データの格納方式 . . . . . 77
3.6.1 ASN.1 データのクラスタリング . . . . . 80
3.6.2 ASN.1 データに対するインデクシング . . . . . 84
3.7 むすび . . . . . 92

**4 広帯域ネットワークにおける分散データベース処理手法 95**

4.1 まえがき . . . . . 96

4.2 ATM 環境のモデル . . . . . 98

4.3 トランザクション処理手法 . . . . . 99

4.3.1 処理の処理時間の定式化 . . . . . 100

4.3.2 データベース移動に基づくトランザクション処理手法 . . . . . 102

4.4 シミュレーション . . . . . 106

4.4.1 各手法の性能比較 . . . . . 107

4.4.2 連続使用優先係数 $P$, 履歴依存係数 $K$ の影響 . . . . . 109

4.5 関連研究との比較 . . . . . 114

4.6 むすび . . . . . 115

**5 結論 117**

5.1 本論文のまとめ . . . . . 117

5.2 今後の課題 . . . . . 120

**謝辞 123**

**参考文献 125**

# 第 1 章

# 序論

## 1.1 研究の背景

計算機システムは，最近数十年間で急激な発展を遂げてきた．元来，数値計算を高速に行なう機器として利用された計算機システムは，現在では人間のさまざまな活動を支援する重要な役割を果たしている．計算機システムはその高速化・高機能化によって，数値計算にとどまらず知識処理や言語処理などの情報処理にも利用されるようになった．また，周辺技術としての計算機ネットワーク技術の進展は，計算機システムの利用形態に大きな変革をもたらした．従来は，多くの端末からの処理要求を一台の高速な計算機ですべて処理する集中処理型のシステムが主流であった．しかし，計算機ネットワーク技術によって複数の計算機を相互接続することが可能になったことにより，集中処理型からクライアント・サーバ型の処理形態へと移行し，計算機処理の分散化を行なうことが可能になった．さらに，個々のネットワークを相互接続したインターネット技術により，計算機を利用して世界中の人々がコミュニケーションをとることが可能になりつつある．

このような計算機システムの急速な発展の中で，データベースシステムはその環境やユーザの要求に適応しながら常に重要な役割を果たしてきた．これは，計算機システムが情報を処理するだけでなく，情報を蓄積・管理し，必要に応じて提供するという大きな役割を常に背負っているからである．具体的には，銀行のオンラインシス

テムの中核となるデータベースシステムが，初期のデータベースシステムの典型的な形態と考えることができる．また，知識処理や言語処理をサポートするものとして，知識ベースシステムと呼ばれる推論能力を備えたデータベースシステムが誕生した．さらに，計算機ネットワーク技術の発展に伴ってクライアント・サーバ型のデータベースシステム，データベースサーバを分散化した分散データベースシステムが出現した．このように，データベースシステムはアプリケーションの多様化や計算機ネットワークの発展に伴って進化を続けており，これまでにデータベースシステムに関する成書が多数出版されている [KS86, Papa86, BHG87, Ullm88, Ullm89, Date97]．

さらに近年，データベース技術の応用分野が拡大している．グループ内での知的生産活動を支援するグループウェアシステムがその一例である．また，計算機ネットワークを利用する通信アプリケーションでも，データベースシステムがもつデータ管理能力が必要となる．さらに，世界的な計算機ネットワークの広帯域化により，世界規模での分散データベースシステムが現実的に構築可能になりつつあるが，この変化はデータベース管理システムの設計手法にも大きな影響を与えている．

本論文では，このような状況を踏まえ，高度計算機ネットワーク環境におけるデータベースシステムに関して考察する．

## 1.2 データベースシステムの構成

データベースシステムの構成要素を表す概念図を図 1.1に示す．図に示したように，データベースシステムには以下の三つの構成要素が存在する [Date97]．

**データベース管理システム (Database Management System, DBMS)**

データベースシステムの中核となる構成要素であり，データベース自身の管理，および，データベースへのアクセス手段を提供する．データベース管理システムは，集中型データベース管理システムと分散データベース管理システムとに大きく分類できる．集中型データベース管理システムでは，ただ一台のデータベースサーバがアプリケーションプログラムやエンドユーザからのすべてのデータベース処理要求を実行する．これに対し，分散データベース管理

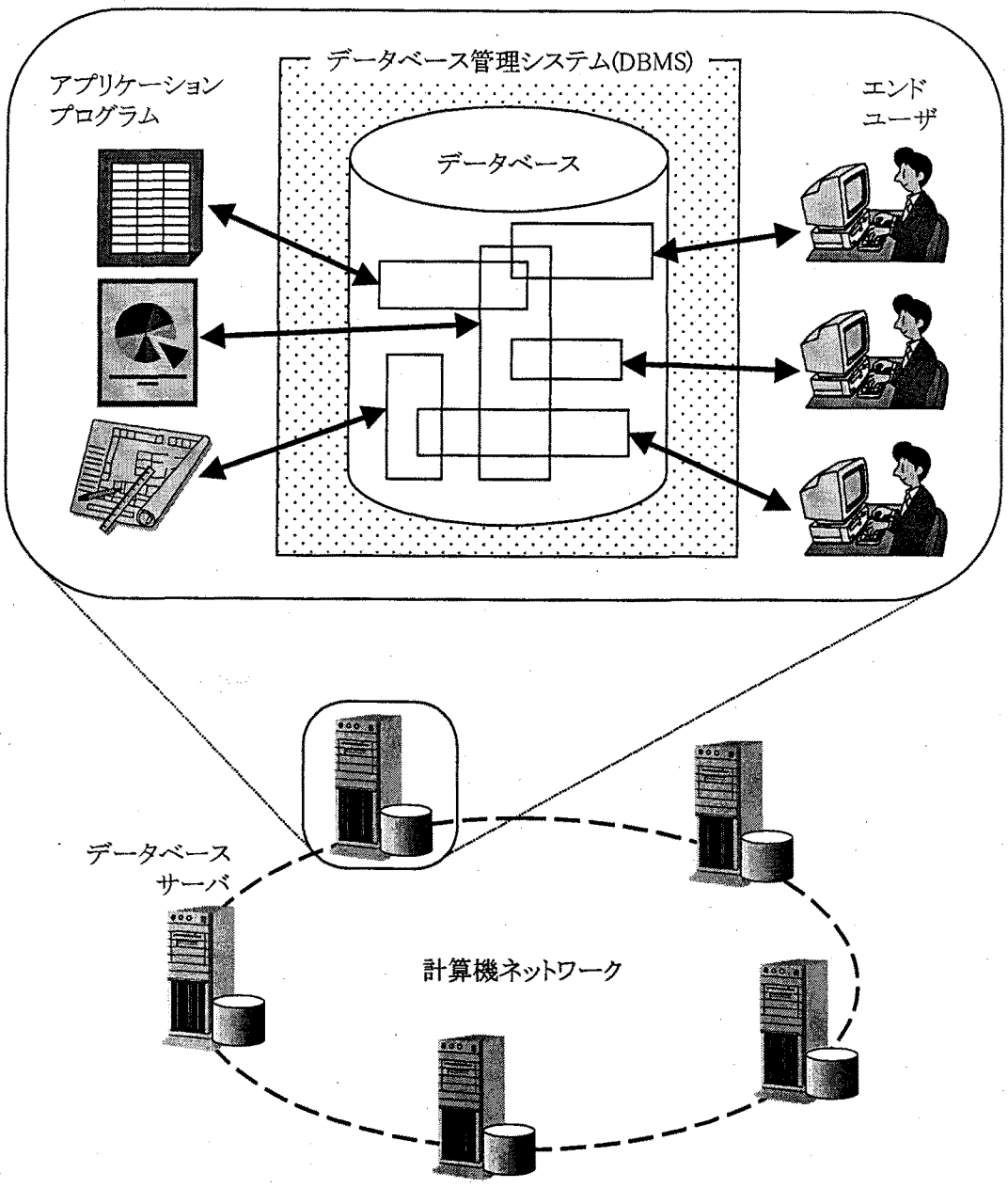



図 1.1: データベースシステムの概念図

システムでは，図1.1の下部に示したように複数のデータベースサーバが計算機ネットワークによって相互接続された形態をとり，データベースは各データベースサーバに分散配置される．アプリケーションプログラムやエンドユーザはデータベースがどのように分散配置されているかを意識することなく，どのデータベースにもアクセスできるように設計されている．

アプリケーションプログラム

データベースへのアクセスを通じて，さまざまな処理を行なうプログラムであり，データベース内のデータから統計処理を行なうアプリケーションや，ネットワークを介して情報提供を行なう通信アプリケーションなどが含まれる．

エンドユーザ

データベースに直接あるいは間接的にアクセスするユーザであり，問合せ言語SQL (Structured Query Language) を用いて直接データベースにアクセスするデータベース管理者や，アプリケーションプログラムを通じて間接的にデータベースにアクセスするオペレータ（一般ユーザ）などが含まれる．

## 1.3 本論文の概要

近年，計算機ネットワークの高度化がデータベースシステムに大きな影響を与えている．ここで，計算機ネットワークの高度化とは以下の事項を指す．

- LAN (Local Area Network) が容易に構築できるようになり，さらにインターネットに代表される世界規模の計算機ネットワークに接続することにより，世界中の計算機と容易に通信できる環境が整ってきたこと．
- 計算機ネットワークの帯域幅が急速に拡大し，大量のデータを短時間で転送可能になってきたこと．

本論文では，前節に示したデータベースシステムの三つの構成要素の各々に対応した観点から，上記のような計算機ネットワーク環境の高度化に適応したデータベー

スシステム構築技術について議論する．第一に，エンドユーザの観点から，グループウェアシステムへのデータベース技術の適用について議論する．第二に，データベースにアクセスするアプリケーションプログラムの観点から，通信アプリケーションからのデータベースの利用を容易にするためのデータベースシステムの構成方法について論じる．第三に，データベースシステムの中核をなすデータベース管理システムの観点から，ATM (Asynchronous Transfer Mode, 非同期転送モード) ネットワーク [Stal95] に代表される広帯域ネットワークを介した分散データベースシステムにおけるトランザクション処理手法について論じる．

### 1.3.1 協調作業環境のためのデータベース管理手法

近年，計算機の低価格化や，LAN が容易に構築できるようになってきたことから，データベースシステムは従来のビジネス分野での応用だけでなくエンドユーザの知的生産活動を支援するためにも利用されるようになってきている．その代表例が，計算機を用いた協調作業支援 (Computer Supported Cooperative Work) システムである [IM94]．CAD/CAM (Computer Aided Design/Computer Aided Manufacturing) における設計作業 [BKK85, KKB88]，CASE (Computer Aided Software Engineering) ツールを用いたソフトウェア開発，共同文書作成といった，非同期蓄積型の協調作業を必要としている分野は広がってきており，これらの協調作業を支援するグループウェアの研究開発が活発になってきている [Ishi91]．

非同期蓄積型の協調作業では，作業者は作業の対象を比較的独立したいくつかの部分に分割分担し，各作業者はそれぞれの作業結果を共有データとして共有することによって作業を進める．共有データには，協調作業を進める上で重要な多くの一貫性制約 (integrity constraints) が課せられている．このような共有データの一貫性制約の管理は協調作業支援システムに必須の機能である．データベース管理システムは，データベースに格納されている共有データの首尾一貫性を保証するためのシステムであり，データを共同利用する非同期蓄積型の協調作業支援システムにおいても基盤となる重要なシステム要素となる．

従来のデータベースシステムは，銀行や航空会社などで用いられているオンライ

ン・トランザクション処理に適するように設計・開発されている．このようなデータベースシステムでは，トランザクションは通常短時間で終了し，トランザクションを並行処理した結果として得られるデータベースアクセス処理の系列（スケジュール）が直列可能であることを保証することによって，データベースの一貫性を保つという手法が取られている [Gray78]．また，データベース上に課せられた一貫性制約を犯すようなトランザクション，あるいは，直列可能性というスケジュールの正当性の基準を保証することができなくなったトランザクションに対しては，そのトランザクションをアボートすることによってデータベースの一貫性を保証するという手法が取られている [KS86, Papa86, BHG87, Ullm88, Date97]．

このような従来のデータベースシステムにおける一貫性保持の手法は，CAD/CAM，CASE，共同文書作成などの応用には適さない．なぜなら，これらの応用ではユーザが実行するトランザクションが数時間から数日に及ぶ長時間トランザクションとなる傾向があるからである．このような長時間トランザクションに対して従来の並行処理制御手法を適用すると，デッドロックやシステム障害などの原因によりトランザクションをアボートしなければならないとき，長時間に及んだユーザの作業結果がすべて失われることになる．また，直列可能性を保証するためにトランザクションの実行に対してあまりに長い待ち時間を生じることが多く，トランザクション間のデータ共有が著しく妨げられる [BK91]．

このような理由から，長時間トランザクションに適した並行処理制御に関する研究が盛んに行なわれている [KL+84, KS88, Lync83, NZ90, PKH88, Weik91, XFS94]．これらの研究では，長時間トランザクションを何らかの方法でより小さな単位に分割し，直列可能性の概念を緩和することによってトランザクションの並行性を向上させようとしている．このような並行処理制御方式は理論的に有効であるとは考えられるが，トランザクションを実行するのが人間であるということを全く考慮しておらず，実際の協調作業支援システムに応用することは非常に困難である．したがって，エンドユーザを意識したより現実的な協調作業支援モデルが要求される．

CAD/CAM などの協調作業では，作業者は他の作業者と連絡をとりながら作業を進めなければならない．特に，一貫性制約を犯すような作業を行なった場合でも，

その作業を取り消すのではなく，他の作業者との協議によってそれを解決するのが妥当である．協調作業では作業の分担があるため，分担した作業領域にまたがって課せられた一貫性制約は一人の作業者による作業では満たすことができないことが多い．したがって，トランザクションが単独で実行されればデータベースの首尾一貫性を保持するという直列可能性の概念の前提条件は協調作業環境には当てはまらない．

さらに，世界規模の計算機ネットワークの発展により，世界規模での協調作業の支援が求められている．このような環境では，時差などにより作業者が同時に作業を行なっているというグループウェアシステムの仮定は成り立たない．

本論文では，データベースのエンドユーザの知的生産活動，特に非同期蓄積型の協調作業を支援するためのデータ管理手法を提案する．提案する手法では，データ間に課せられた一貫性制約に着目し，作業者が一貫性制約について完全には周知していなくても，作業者間の協議が必要になった際にシステム側から作業者に通知することにより，協調作業を支援するものである．さらに，作業内容が国際的になった場合の時差あるいは作業者の作業時間のずれなどにより，すべての作業者が同時に作業を行なっているとは限らないことを重視し，作業者間の協議が即座に行なえない場合にも作業を進行させることができるような制御モデルを提案する．

### 1.3.2 通信アプリケーションのためのデータベースシステムの構築

インターネットに代表される世界規模の計算機ネットワークの発展によって，データの管理は従来の集中型管理から分散型管理へ移行し，種々のデータをその管理母体に応じて分散配置することが可能になった．このような環境では，クライアントからのデータの検索要求や更新要求を処理するために，分散されたサーバ間，および，サーバとクライアント間で，要求・応答メッセージやデータの交換が必要になる．

データ管理の分散化に伴い，データの安全な共有を可能にするデータベース管理システムが果たす役割は，従来にも増して重要なものとなってきている．特に，従来の分散データベースシステムのようにデータベースシステムが計算機ネットワーク環境を利用するという形態ではなく，計算機ネットワーク環境の上で動作する通

信アプリケーションに対してデータベースシステムが支援するという形態でのデータベースシステムの役割が重要となってきている．このような支援を行なうデータベースシステムを設計する際には，通信アプリケーションからの要求に応えることをまず念頭におかなければならない．

計算機ネットワーク上にデータが分散化されることによって生じる大きな問題の一つは，それらのデータを管理している計算機システムの異種性である．異種のシステムに分散したデータをやりとりする通信アプリケーションでは，通常，ネットワーク上でのデータ表現を計算機システムのアーキテクチャに依存しない形に統一することによってこの問題に対処している．その代表的なものに，RPC (Remote Procedure Call, 遠隔手続き呼び出し)[Sun88] のために開発されたデータ表現形式である XDR (eXternal Data Representation)[Sun87] や，ISO (International Organization for Standardization, 国際標準化機構) で定められた ASN.1 (Abstract Syntax Notation One, 抽象構文記法 1 )[ISO87a, ISO87b] がある．

本論文では，通信アプリケーションで広く利用されている ASN.1 に焦点をあて，ASN.1 を利用する通信アプリケーションを支援するデータベースシステムの設計・構築手法に関して論じる．まず，通信アプリケーションからのデータベース機能の利用を容易にし，アプリケーション開発効率を向上させるためのデータベースプログラミング言語を提案する．提案するプログラミング言語は，ASN.1 のデータ構造を直接扱えることにより，従来からデータベースプログラミングにおける問題とされているインピーダンスミスマッチを解消し，アプリケーション実行時の型誤りを防ぐ．次に，通信アプリケーション支援に適したデータ格納部の実現方法として，オブジェクト指向データベース管理システム (Object-Oriented Database Management System, OODBMS) を利用することを考え，ASN.1 で定義されたデータ構造を OODBMS でのスキーマ定義に変換する ASN.1 コンパイラの実現方法について論じる．さらに，より高速なデータベース処理が必要となる通信アプリケーションの支援を考え，ASN.1 の符号化規則を考慮したデータ格納方式を提案する．

### 1.3.3 広帯域ネットワークにおける分散データベース処理手法

近年，ATM 方式を中心として，計算機ネットワーク技術が急速に発展している．特に著しいのが帯域幅の拡大であり，数百 Mbps や数 Gbps の帯域幅をもつ計算機ネットワークも構築されつつある．帯域幅の拡大は，大量のデータの短時間での転送を可能にし，クライアント・サーバ型のシステムでの応答時間の短縮をもたらすものと期待される．

一方，帯域幅の拡大は，分散処理において通信コストがボトルネックであったこれまでの状況が一変することを意味している．つまり，計算機ネットワーク上の分散処理に関する研究は，伝送データ量の削減による性能向上から，帯域幅の有効利用による性能向上を目的とした技術の開発へと移行する必要がある．データベース分野においても，広帯域ネットワークの特性をトランザクション処理の高速化に利用することが可能である [BLW93, HG+87]．従来の分散データベースシステムでは，データベースは特定のサイトに固定され，それらに対する処理はメッセージによる処理依頼で行なうなど，伝送データ量を削減する手法を開発することが一般的であった．しかし，広帯域ネットワークでは，データベース自体をネットワークを介して移動させること（以下，データベース移動と呼ぶ）も現実的に可能である．

データベース移動を用いることにより，これまでの分散データベース技術とは異なる新たな可能性が出てくる．例えば，従来のように各データベースをいずれかのサイトに固定して配置しておき，処理依頼および二相コミットプロトコル (2-phase commit protocol, 2PC)[Gray78] によってトランザクション処理を行なった場合，各サイトに分散するデータベースに対して処理依頼メッセージによる処理が複数回行なわれた後，二相コミットのためのメッセージ交換を二往復する必要がある．一方，データベース移動を用いた場合，アクセス対象となるデータベースをトランザクション発生サイトに移動してしまえば，その後のメッセージ交換の必要がない．広帯域ネットワークでは，大量のデータの転送は短時間で行なえるが，伝搬遅延は従来のネットワークとほとんど差がないことから，データベース移動による処理時間の短縮が期待できる．

ネットワークの帯域幅が急速に拡大しているのとは対照的に，通信の伝搬遅延に

関しては光の伝搬速度が決まっているために大幅な改善は不可能である．したがって，ネットワークの広域化，データベース処理の多様化が進んでいくにつれて，データベース移動が強力で有効なデータベース技術の一つになり，様々な用途に有効に活用されるものと考えられる．

本論文では，データベース移動を利用したトランザクション処理手法を提案する．また，すべてのトランザクションに関してデータベース移動による処理の方が2PCを用いたデータベース固定型の処理よりも効率的であるとは限らない．したがって，トランザクションの性質に応じて固定処理とデータベース移動の二手法のどちらを適用するかを選択する適応型の処理手法を提案する．

## 1.4 本論文の構成

本論文の以下の章は次のように構成される．

まず第2章では，協調作業支援環境における長時間トランザクションの処理の問題について述べ，協調作業に適したデータベースモデルおよびトランザクションモデルを提案する．また，共同作業者の作業の非同期性などを考慮したトランザクションの実行制御モデルを提案する．

第3章では，データベース機能を必要とする通信アプリケーションの開発における問題点について言及し，その解決策としてネットワーク上でのデータ表現に着目したデータベースシステムの設計および実装手法について論じる．特に，アプリケーション開発に用いるプログラミング言語，および，実際のデータ格納部の実現方法について論じる．

第4章では，近年の計算機ネットワークの広帯域化に注目し，計算機ネットワーク上で動的にデータベースを移動させることによって分散データベース処理の高速化が可能であることを示す．

第5章では，本論文の内容を要約し，最後に今後の研究課題について述べる．

なお，第2章は文献 [YHN94], [HYN96] で公表した結果に基づき論述する．第3章は文献 [HN+94], [NH+94], [HTN94a], [HTN94b], [SH+95a], [HH+95], [SH+95b],

[HTN96b], [HTN96a] で公表した結果に基づき論述する．第 4 章は文献 [NT+95], [HH+96a], [SH+96], [HH+97a], [HH+97d], [HH+97e], [HH+98] で公表した結果に基づき論述する．

# 第 2 章

# 協調作業支援のためのデータベース管理手法

近年，計算機の小型化と高機能化，および，計算機ネットワークの普及と高速化に伴い，データベースシステムの応用分野が拡大している．その一つに，エンドユーザの知的生産活動を支援するような応用でのデータベース技術の利用がある．

本章では，このようなエンドユーザの知的生産活動，特に非同期蓄積型の協調作業を支援するためのデータ管理技術について述べ，データベース管理システムに要求される要件，協調作業支援に適したデータベースモデルおよびトランザクションモデル，および，協調作業支援システムを実現する際の作業の制御モデルに関して議論する．

## 2.1 まえがき

近年，計算機の小型化と高機能化，および，計算機ネットワークの普及と高速化に伴い，データベースシステムの応用分野が拡大し，従来のビジネス分野での応用だけでなく，エンドユーザの知的生産活動を支援するためにもデータベースシステムが利用されるようになってきている．その代表例が，計算機を用いた協調作業支援 (Computer Supported Cooperative Work) システムである [IM94]．CAD/CAM (Computer Aided Design/Computer Aided Manufacturing) における設計作業 [BKK85, KKB88]，CASE (Computer Aided Software Engineering) ツールを用いたソフトウェア開発，共同文書作成といった，非同期蓄積型の協調作業を必要としている分野は広がってきており，これらの協調作業を支援するグループウェアの研究開発が活発になってきている [Ishi91]．

非同期蓄積型の協調作業では，作業者は作業の対象を比較的独立したいくつかの部分に分割分担し，各作業者はそれぞれの作業結果を共有データとして共有することによって作業を進める．共有データには，例えばソフトウェア開発では,「モジュール A とモジュール B の間で関数インタフェースが一致していなければならない」，あるいは機械の設計では,「部品 A と部品 B のネジ穴の位置が一致していなければならない」や「総部品点数がいくつ以下でなければならない」といった，多くの一貫性制約 (integrity constraints) が課せられている．このような共有データの一貫性制約の管理は協調作業支援システムに必須の機能である．データベース管理システムは，データベースに格納されている共有データの首尾一貫性を保証するためのシステムであり，データを共同利用する非同期蓄積型の協調作業支援システムにおいても基盤となる重要なシステム要素となる．

従来のデータベースシステムは，銀行や航空会社などで用いられているオンライン・トランザクション処理に適するように設計・開発されている．このようなデータベースシステムでは，トランザクションは原子性 (atomicity) をもつ短時間トランザクションであり，トランザクションを並行処理した結果として得られるデータベースアクセス処理の系列 (スケジュール) が直列可能であることを保証することによって，

データベースの一貫性を保つという手法が取られている [EG+76, Gray78, Gray81]. また，データベース上に課せられた一貫性制約を犯すようなトランザクション，あるいは，直列可能性というスケジュールの正当性の基準を保証することができなくなったトランザクションに対しては，そのトランザクションをアボートすることによってデータベースの一貫性を保証するという手法が取られている.

このような従来のデータベースシステムにおける一貫性保持の手法は，CAD/CAM, CASE，共同文書作成などの応用には適さない．なぜなら，これらの応用ではユーザが実行するトランザクションが数時間から数日に及ぶ長時間トランザクションとなる傾向があるからである．このような長時間トランザクションに対して従来の原子性の概念を要求することは，デッドロックやシステム障害などの原因によりトランザクションをアボートしなければならないとき，長時間に及んだユーザの作業結果がすべて失われることを意味する．また，長時間トランザクションに対して従来の直列可能性を保つという基準で並行処理制御を行なうと，そのためにトランザクションの実行に対してあまりに長い待ち時間を生じることが多く，トランザクション間のデータ共有が著しく妨げられる.

CAD/CAM などの協調作業では，作業者は他の作業者と連絡をとりながら作業を進めなければならない．特に，一貫性制約を犯すような作業を行なった場合でも，その作業を取り消すのではなく，他の作業者との協議によってそれを解決するのが妥当である．協調作業では作業の分担があるため，分担した作業領域にまたがって課せられた一貫性制約は一人の作業者による作業では満たすことができないことが多い．したがって，トランザクションが単独で実行されればデータベースの首尾一貫性を保持するという直列可能性の概念の前提条件は協調作業環境には当てはまらない.

以上の議論をまとめると，グループウェアシステムでの協調作業を支援するためのデータベースシステムでは，以下のような機能が要求されることになる.

- ユーザが実行する長時間トランザクションに対する原子性の概念を緩和する.
- データベース管理システムが直列可能性を保証するというシステム側での並

行処理制御だけでなく，ユーザ間の協調を促すことによってもデータベースの一貫性を保つような支援を行なう．

以下，2.2節では長時間トランザクションの処理に適した並行処理制御方式に関する関連研究について述べる．2.3 節では，本章で提案する協調作業支援のためのデータベース管理手法の基本的なアプローチについて説明する．2.4節では，協調作業データベースのモデル，および，そのデータベース上で実行されるトランザクション（ジョブ）のモデルを定義する．2.5節では，ジョブの協調実行を支援するための制御手法，および，協調作業が非同期に行なわれることを考慮した作業の取り消しの方法について論じる．また，提案するデータベース管理手法がさまざまな協調作業に応用できることを示すために設計・実装した，共同文書作成を支援するシステムである COCOA システムについて，2.6節で説明する．最後に 2.7節で本章のまとめとする．

## 2.2 関連研究

長時間トランザクションを扱うことを可能とし直列可能性の概念を緩和するための並行処理制御方式について，理論的な立場から多くの研究がなされている [BK91]．これらの並行処理制御方式の多くでは，トランザクションの原子性の概念を緩和した新しいトランザクションモデルを導入し，トランザクションの並行性を向上させながら，データベースの首尾一貫性を保つ制御手法を取っている．

協調作業の進行を考えると，まず作業全体はいくつかの部分作業に分割され，それぞれのジョブは一つの作業者グループあるいは一人の作業者に割り当てられる．さらに，ジョブはより小さな作業単位に分割されて実行されることが多い．

入れ子型トランザクションモデル (nested transaction model)[Moss85] は，このようにトランザクションを階層的に組織して，原子性の粒度を小さくし並行処理性能を向上させる枠組みを与えており，協調作業における長時間の作業に対する一つのトランザクションモデルを与えていると考えることができる．このモデルでは，各トランザクションは複数のサブトランザクションを並行して実行することができる．

トランザクションの実行の正当性の基準としては，すべてのサブトランザクションがそれらを生成した親トランザクションに対して直列可能でなければならないという基準を用いている．

しかし，入れ子型トランザクションモデルは，元来は既存のサブトランザクションをまとめてより大きなトランザクションを定義し，それぞれのサブトランザクションを分散実行することをねらったものであり，大きなトランザクションをより細かい単位に分割して実行することを考慮したものではない．また，共有データに対する一貫性は直列可能性という基準だけで保証するため，協調作業のようにユーザ間のインタラクションが必要となる分野にはそのまま適用できない．

トランザクション間のインタラクションを考慮した入れ子型トランザクションモデルの例として，文献[KL+84]で提案された協調設計作業用のトランザクションモデルがある．このモデルでは，データベース全体を共用データベース (public database)，半共用データベース (semi-public database)，個人データベース (private database) に分割して考える．共用データベースは，協調作業全体で共有されるデータが置かれるデータベースであり，常に一貫性が保たれているデータベースである．一方，個人データベースは一つのトランザクションの作業空間である．共有データを更新する際には共用データベースから個人データベースにデータをチェックアウトし，個人データベース内で更新作業を行なう．トランザクションがコミットする際には，更新された共有データを共用データベースにチェックインする．さらに，半共用データベースはトランザクションとそのサブトランザクションの間で更新途中の共有データを受け渡すために用いられるデータベースであり，半共用データベースにおいては一貫性が保たれていなくてもよいことになっている．このように半共用データベースを設置することにより，ユーザ間でのデータの受け渡しを可能にしている．

一方，トランザクション実行の正当性の基準である直列可能性の概念を緩和することについても多くの研究が行なわれている．

例えば，文献[Lync83]では多重レベル原子性 (multilevel atomicity) と呼ばれるトランザクションモデルが提案されている．このモデルではトランザクションの間に他のトランザクションがどのように割り込みできるかを指定することにより直列可

能性を緩和している．また，文献 [NZ90] では，パターン (pattern) とコンフリクト (conflict) と呼ばれる状態遷移図を指定することによりスケジュールの正当性を判断する方法が提案されている．しかし，これらのモデルでは，データベースにアクセスするトランザクションがプログラムとして記述されており静的であることが前提となっている．したがって，ユーザがアドホックに処理を進行させるような協調作業には適用できない．

文献 [PKH88] では，分割トランザクション (split-transaction) と呼ばれるトランザクションモデルが提案されている．このモデルでは，一つのトランザクション $T$ を，一方のトランザクション $T_A$ は他方のトランザクション $T_B$ の結果を読み書きできないが，$T_B$ は $T_A$ の結果を読み書きできるような二つのトランザクションに分割することにより，トランザクションの独立性を緩和する方法を提案している．また，多重レベルトランザクション (multilevel transaction)[Weik91] では，トランザクションの操作をいくつかの抽象化された操作に階層化し，それらの可換性を利用することにより並行性を向上させている．このような並行処理制御方式は理論的に有効であるとは考えられるが，実際の協調作業支援システムに応用することは非常に困難である．したがって，より現実的な協調作業支援モデルが要求される．

## 2.3 本論文のアプローチ

実際の協調作業の一般的な進行過程を追跡すると，「ある作業者があるデータ項目を書き換えたい場合，その書き換えを始める前あるいは書き換えた後に，その書き換えによって影響を受ける可能性がある作業者にメッセージを送り，場合によっては協議することによって全体の一貫性を保つようにする」というような顕著なパターンが考えられる．ただし，このような方法が可能であるためには，各作業者が作業の進行にしたがって変化していく一貫性制約すべてを周知していなければならない．

本論文では，データ間に課せられた一貫性制約に着目し，作業者が一貫性制約について完全には周知していなくても，作業者間の協議が必要となった際に作業者に通知することにより，協調作業を支援するデータ管理モデルを提案する．本モデル

では，共有データを格納するデータベースに領域一貫性制約の概念を新たに導入し，一貫性制約の課せられたデータの集合（領域）と一貫性制約条件を明示的に定義できるようにする．このデータベース上で作業者が行なう作業の単位を，通常のトランザクションの概念と区別するためにジョブと呼び，ジョブが書き込みを行なう際に領域一貫性制約が満たされることを保証することにより，データベースの一貫性の保持を実現する．

協調作業環境におけるもう一つの重要な問題は，作業内容が国際的になった場合の時差，あるいは作業者の作業時間のずれなどにより，共同の目的に向かって作業をしているものの，すべての作業者が同時に作業を行なっているとは限らないことである．したがって，協議が必要な場合でも即座に協議できないことがあり，その場合，作業が中断してしまう．本モデルではこの問題を重視し，即座に協議できない場合でも作業が続けられるような制御モデルを提案する．特に協議が行なわれない状態で継続した作業は，他の作業者に対して望ましくないものである可能性があるため，データベースの一貫性を保ちながら特定の領域に対する操作を取り消す方法について言及する．

## 2.4 協調作業データベースとジョブ

協調作業を支援するデータベース管理システムでは，その作業で共有されるデータ間の一貫性制約を保持する管理機構が特に必要である．特に，以下のような制御が可能なデータ管理機構が要求される．

1. トランザクションの実行によって一貫性制約が満たされなくなる場合に，これを検出し，そのトランザクションをアボートするのではなく一貫性制約を満たすように作業者に対してトランザクションの継続を促す．

2. 一貫性制約を複数のトランザクション，すなわち，複数の作業者による協調的な作業の実行によって満たすことを許す．

従来のデータベースモデルでは，一貫性制約は最低限満たされなければならないものだけが明示的に指定される．データベースに課せられた暗黙の一貫性制約に関しては，トランザクションが単独で実行されれば保持されるとの仮定に基づき，トランザクションの直列可能性を保証することによって維持されている．しかし，このようなデータベースモデルでは，暗黙の一貫性制約の存在のために上記 1. のような制御を行なうことが不可能であり，さらに，直列可能性の保証はトランザクション間の協調を阻害する要因となるため，上記 2. の要求を満たすことは不可能である．

協調作業に用いるデータベースでは，その上で作成する目的物に関して満たされなければならないすべての一貫性制約を明示的に記述することが可能であると考えられる．したがって，明示的に指定された一貫性制約をすべて保証する制御を行なうことにより，協調作業を阻害する一つの要因である直列可能性の概念を緩和することができる．

本節では，以上の考えに基づき，領域一貫性制約の概念を導入したデータベースモデルと，そのデータベース上で行なわれる作業の最小の単位であるジョブのモデルを定義する．

### 2.4.1 領域一貫性制約に基づくデータベースモデル

本節では，共有データベースを単なるデータ項目の集合と見るのではなく，すべての一貫性制約が明示的に記述されたデータベースモデルを定義する．ここでは，一貫性制約条件が課せられたデータ項目の集合を領域と呼び，その条件を明示的に指定することができる領域一貫性制約の概念を導入する．

**定義 1** データベース $D$ は，アクセスの対象となる最小の単位であるデータ項目の有限集合 $I = \{x_1, ..., x_n\}$ と領域一貫性制約の有限集合 $C = \{c_1, ..., c_m\}$ の二つ組

$$D = \langle I, C \rangle$$

で表される．領域一貫性制約 $c_i$ $(i = 1, ..., m)$ は，一貫性制約が課せられたデータ項目の集合 $\delta_i$ と，明示的に課せられた一貫性制約条件 $\pi_i$ の二つ組

$$c_i = \langle \delta_i, \pi_i \rangle$$

である．この$\delta_i$を領域と呼ぶ．

ここで，データ項目の集合$I$や，領域一貫性制約の集合$C$，さらにこの領域一貫性制約$c_i$ $(i = 1, ..., m)$は作業の進行に伴って変化し得るものである．また，データベース内のすべてのデータ項目に対して，何らかの一貫性制約条件が課せられているものとする．すなわち，

$$I = \bigcup_{i=1}^{m} \delta_i.$$

□

**例 1** ソフトウェア開発を行なう場合，ソースファイルがデータベースに格納される．ソースファイルへのアクセス単位を一つの関数定義とすると，図 2.1のように関数 `main(int argc, char **argv)`, `sub1(int x)`, `sub2(int x, int y)` の三つの関数定義が存在する場合，関数の呼び出し関係において，`main()` と `sub1(int x)` の間，および，`sub1(int x)` と `sub2(int x, int y)` の間に，それぞれ引数の数と型が一致しなければならないという一貫性制約条件が課せられている．

このような一貫性制約は，作業者自身が明示的に記述しなくても，すべてのソースファイル中の関数呼び出し関係を調べることにより抽出することができる． □

この例のように，領域一貫性制約に基づくデータベースモデルは，領域一貫性制約集合$C$を指定することによって，さまざまな実世界の協調作業アプリケーションにおける共有データベースの一貫性を記述することができる．

### 2.4.2 ジョブモデル

従来のデータベース管理システムにおけるトランザクションと同様に，データベース全体の一貫性を保証するためには，協調作業データベースシステムにもデータベースの一貫性を保持するような，開始と終了の概念をもつ作業の単位が必要である．そこで，そのような作業単位としてジョブを定義する．協調作業環境における一つの作業は，従来のデータベース管理システムにおける短時間トランザクションと異なる性質をもつため，これを区別するためにトランザクションではなくジョブと表現する．

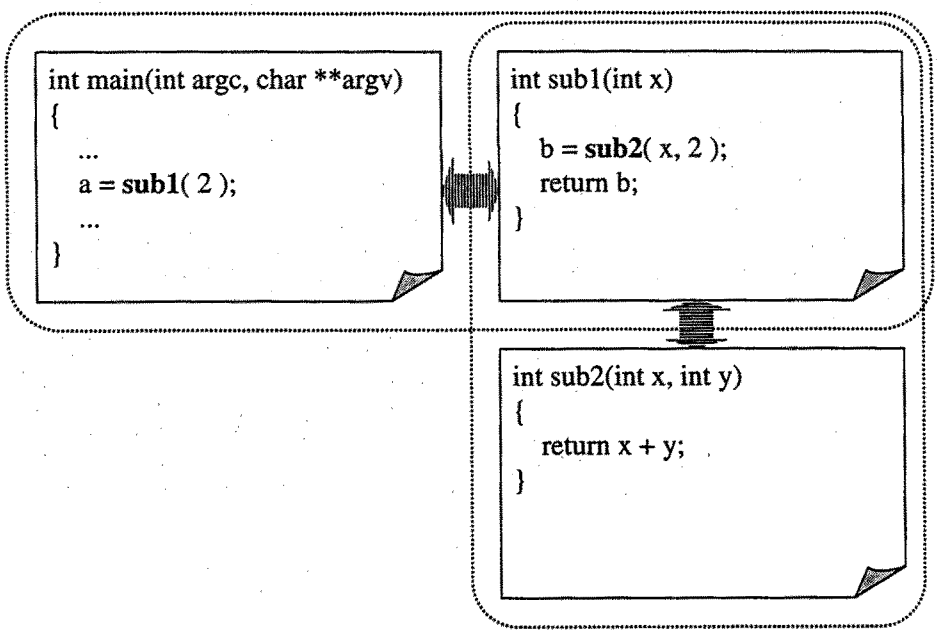



図 2.1: 領域一貫性制約の例

あるデータ項目の書き込み操作が行なわれた場合，領域一貫性制約を満たすために，他のデータ項目も書き換えなければならないことがある．協調作業では，そのようなデータ項目をジョブの実行者が直接書き換えることが困難であることが多い．例えば，図 2.1の例で，ある作業者が関数 `sub2(int x, int y)` の定義を `sub2(int x, int y, int z)` に変更した場合は，関数 `sub1` 内の `sub2` を呼び出している部分を書き換えなければならないが，`sub1` と `sub2` の開発担当者が異なれば，この書き換えは `sub1` の担当者でなければ困難である．そこでサブジョブの概念を導入し，複数の作業者がひとつのジョブに参加できる枠組を提供する．

**定義 2** ジョブ $J$ が読み出し・書き込みを行なうデータ項目の集合を$\xi_J$とする．ジョブ $J$ は，$\xi_J$に属するデータ項目 $x$ を読み出す操作 $r(x)$，$x$ を書き込む操作 $w(x)$，サブジョブを生成する操作 $S$ を実行することができる．ジョブ $J$ はその操作の系列 $OP_J$によって定義される．

ただし，ジョブがある時点までに読み込んだデータ項目の集合，書き込みを行なっ

たデータ項目の集合を，それぞれ $\Xi^r, \Xi^w$ とすると，

$$\Xi^r \supseteq \Xi^w$$

の関係が，任意の時点で成り立つものとする．

ジョブ $J$ はデータベースの一貫性を保証する最小の実行単位であり，次の二つの性質が満たされると仮定する．

- ジョブが単独で実行された場合，データベースに課せられているすべての領域一貫性制約が満たされた状態が保持される．
- ジョブ内で実行されるどの操作を取り除いても，すべての領域一貫性制約を満たすことができなくなる． □

## 2.5 ジョブの実行制御モデル

一つの協調作業は，複数の作業者がジョブを実行することによって進行する．領域一貫性制約に基づくデータベースモデルにおいては，一貫性制約条件が課せられたそれぞれの領域ごとに直列可能性を保証することにより，データベースの一貫性を保つことができる．このような正当性の基準は制約条件充足直列可能性 (predicatewise serializability)[KS88]，あるいは，独立化可能性 [XFS94] として提案されており，この基準に合致したスケジュールはデータベースの首尾一貫性を保持することが示されている．

この正当性の基準を用いた並行処理制御を行なうことにより，直列可能性という長時間トランザクションにとっては厳しい正当性の基準を緩和することができる．しかし，実際の制御方法については，文献 [KS88]，文献 [XFS94] では示されておらず，また，そのまま協調作業環境に適用した場合，領域が重なる二つのトランザクション（ジョブ）については，やはり長時間の待ちが生じるなどの問題がある．

本節では，複数のジョブを同時に実行する際の制御モデルについて述べる．本モデルでは，ジョブの協調実行の概念を導入することにより，長時間の待ちを生じさせずに作業を行なうことを可能とする．さらに領域予約操作を導入し，領域一貫性

制約を考慮することにより，作業者間で協議が必要であることをシステム側で判定する方法について示す．協議が必要な場合は，作業者へ通知を行なうことにより，作業者は互いの影響を強く意識することなく作業を進めることが可能となる．

### 2.5.1 ジョブの協調実行の方法

#### 2.5.1.1 ジョブ間の協調実行

従来のデータベースにおけるトランザクションと同様に，データベースが一貫性を保っている状態でジョブを単独で実行すれば，データベースの一貫性は保たれる．したがって，データ項目の読み出しについて次の二つの条件を満たすことが要求される．

**条件 A** ジョブがデータ項目 $x$ を読み出す時，$x$ が含まれるすべての領域 $\delta_l$ において，課せられている一貫性制約条件 $\pi_l$ が満たされている．

**条件 B** ジョブが読み出したデータ項目 $x$ は，そのジョブが書き込みを行なうまで，あるいはジョブが終了するまで内容が変化しない．

ジョブの性質により，ジョブ $J$ からどの一つのデータ項目に対する書き込み操作 $w(x)$ を除いても，データ項目 $x$ の属する領域 $\delta_l$ 内の一貫性制約条件 $\pi_l$ は満たされないことになる．したがって，条件 A から，ジョブ $J$ 内における最初の書き込み操作と最後の書き込み操作の間は，$J$ が書き込みを行なったデータ項目と同じ領域に含まれるデータ項目を，他のジョブが読み出すことができなくなってしまう．

この待ち時間をなくすためには，ジョブの実行途中における中間結果を共有データベース空間に書き込まないようにすればよい．そこで，共有データベースとは別に，各ジョブに対してそのジョブ専用の作業空間を割り当てる．そして，ジョブ内で行なわれる書き込み操作はジョブに対して割り当てられた作業空間に対して行ない，ジョブがコミットの要求を出した時に一括して書き込む，チェックイン・チェックアウトモデルを導入する．これにより，共有データベースへの書き込み操作がジョ

ブの完了時まで遅延されるため，他のジョブがデータの読み出しを待つことはなくなる．

共有データベースに対してデータの書き込みが行なわれると，他の作業者に影響が及ぶため，作業者間の協議が必要となる．ここで必要となる協議は，データ項目の変更内容に関する報告である．

したがって，データ項目の書き込みの要求を次のように処理する．

**方法 1** データ項目の書き込みの要求に対して，システムはジョブに割り当てた作業空間に対して書き込みを行なう．ジョブから明示的にコミットの要求が出されたときに，システムはすべての領域一貫性制約を満たすことができることを確認し，共有データベースへの書き込みを行なう．そして，書き込みを行なったデータ項目 $x$ を読み出して作業しているジョブの実行者に対して，$x$ がどのように変更されたかについて通知を行なったのち，ジョブを完了する．領域一貫性制約がいずれか一つでも満たせない場合は，コミットできないこと，および，なぜコミットできないか，つまりどの一貫性制約条件を犯しているかをジョブの実行者に通知し，ジョブの実行を続ける． □

あるジョブがコミットしたとき，そのジョブが共有データベースに対して書き込みを行なったそれぞれのデータ項目 $x$ について，$x$ を読み出して作業している他のジョブの実行者へ通知が行なわれる．これにより，すべてのジョブは仮想的にデータ項目 $x$ を最後の書き込み以降に読み出したことになり，条件 B を満たすことが可能となる．したがって，条件 A, B が共に満たされるので，データベースは一貫性を保つことが可能となる．

#### 2.5.1.2 ジョブとサブジョブの協調実行

ジョブの実行者が直接変更することが困難であるデータ項目の変更は，サブジョブとして他の作業者に作業を委任することにより行なう．したがって，ジョブは実行が完了したときにすべての領域一貫性制約を満たすことが可能となる．

一方サブジョブは，ジョブの一部として実行されるジョブである．サブジョブを生成したジョブのことをそのサブジョブの親ジョブと呼ぶ．親ジョブのコミットは

すべてのサブジョブが終了するまで遅延される．サブジョブは，作業空間を親ジョブと共有し，サブジョブのコミット時には共有データベースへのデータ項目の書き込みや領域一貫性制約を満たすことについての確認は行なわない点で，ジョブと大きく異なる．

**方法 2** サブジョブによるデータ項目 $x$ の読み出しの要求に対して，システムは，親ジョブの作業空間に $x$ が存在すればそれを読み出し，存在しなければデータベースから読み出しを行なう．サブジョブによるデータ項目 $x$ の書き込みの要求に対して，システムは親ジョブの作業空間に対して書き込みを行ない，$x$ を読み出している親ジョブとそのサブジョブの実行者に対して，$x$ が変更されたことと，どのように変更されたかについて通知する． □

### 2.5.2 領域予約操作による予告通知

ジョブの協調実行によって，複数のジョブを並行して実行した場合でも，領域一貫性制約を満たすような結果を得ることが可能となる．しかし，実際の作業において，データベース内のデータ項目が変更されてから通知が行なわれるのでは，作業効率が著しく低下する場合がある．例えば，2 つのジョブ $J_u, J_v$ があり，ジョブ $J_v$ があるデータ項目 $x$ を参照して作業を行なっているとする．このとき，$J_u$ が $x$ を変更し，書き込み操作を行なうと，$J_v$ は $x$ の変更点をふまえて，再度作業を行なう必要が出ることがある．つまり，$J_u$ の作業中に行なった作業が無駄になることがあり得る．

このような問題は，あるデータ項目 $x$ を読み出しているジョブの実行者に対して，あらかじめ変更が行なわれることを予告することによりある程度回避することができると考えられる．この予告を行なうために，領域予約という方法を用い，変更を行なう領域を予め宣言することにする．

**定義 3** データベース $D$ 内に存在するデータ項目のうち，書き込みを行なうデータ項目の集合にマークをつける操作を，領域予約という． □

領域予約の操作は，ジョブの中で実行される．したがって，ジョブの定義を次のように拡張する．

**定義 4** ジョブは $r(x), w(x), S$ の 3 つの操作に加え，$x$ を領域予約する操作 $R(x)$ を実行することができ，ジョブ $J$ はその操作の系列 $OP_J$ によって定義される．

ただし，ジョブがある時点までに領域予約したデータ項目の集合を $\Xi^R$ とすると，

$$\Xi^r \supseteq \Xi^R \supseteq \Xi^w$$

の関係が，任意の時点で成り立つものとする． □

ジョブからの領域予約の要求は，次のように処理する．

**方法 3** データ項目 $x$ の領域予約の要求に対して，システムは，データ項目 $x$ を読み出しているジョブの実行者に対して，$x$ が変更される予定であることを即座に通知する．また，領域予約が行なわれた後，別のジョブからデータ項目 $x$ の読み出しの要求が出されれば，そのジョブの実行者に対して，読み出しと同時に，$x$ が変更される予定であることを通知する．領域予約は，領域予約を行なったジョブが終了するまで保持する． □

領域予約は，従来の並行処理制御方式でとられていた施錠とは異なり，あるジョブによって領域予約がなされているデータ項目 $x$ に対して，他のジョブは $x$ の読み出しも書き込みも制限なく行なうことが可能である．領域予約は，データ項目の変更を行なうことで他の作業者に影響が出る場合に，変更を行なう前に作業者間で協議を行なうことを促すための方法である．上記のジョブの定義では，領域予約を行なったデータ項目に対してのみ書き込みが行なえることになっているが，それは領域予約を最大限に利用するためである．

### 2.5.3 領域一貫性制約の考慮による通知対象の拡大

協調作業で共有されているデータ項目のうち，他のデータ項目が存在してはじめて有効となるデータ項目が存在する．また，実際の作業においては，他のデータ項

目が存在するばかりでなく，それがある決められた値をもっていてはじめて有効となるデータ項目が存在する．例えば，文書作成を行なっている場合，他の作業者の記述や図表を引用することがある．このような場合，引用した記述や図表が存在するだけではなく，引用できるだけの意味がある内容である必要がある．

**定義 5** データ項目 $x$ が別のデータ項目 $y$ が存在し，それがある定められた値であることによって有効となる場合，$x$ は $y$ に依存しているといい，$x \gg y$ と表す．またこのような $x$ と $y$ の関係を依存関係とよぶ． □

このような依存関係は領域一貫性制約から導出することができる．例えば，二つのデータ項目 $x, y$ からなる領域に対して「データ項目 $x$ を使用するためには，データ項目 $y$ が存在し，それがある定められた値でなければならない」という一貫性制約条件が課せられている場合，この条件から "$x \gg y$" という依存関係を導出することができる．また,「データ項目 $x$ と $y$ に対して，$x = y$ でなければならない」という一貫性制約条件が課せられている場合，この条件から "$x \gg y$" と "$y \gg x$" の二つの依存関係を導出することができる．

**例 2** 例2.1において，関数 main の中で関数 sub1 を用いるためには，関数 sub1 が定義されている必要がある．すなわち，main は sub1 に依存している．同様に，sub1 は sub2 に依存している．したがって，main ≫ sub1，sub1 ≫ sub2 の関係が存在する． □

このデータ項目間の依存関係を考慮して，領域予約の要求，およびデータ項目の書き込みの要求に対する処理を，次のように拡張する．

**方法 4** 一貫性制約条件から $x \gg y$ なる関係を導出することができるとき，$y$ を含む領域に対する領域予約の要求に対して，システムは，データ項目 $x(\in \delta_l - \{y\})$ を生成した作業者に対しても，$y$ が変更される予定であることを通知する．また，$y$ の書き込みをデータベースに対して行なった場合は，$x$ を生成した作業者に対しても，$y$ がどのように変更されたかについて通知を行なう． □

この拡張により，実際には一貫性制約条件として定義が難しい意味的な制約についても，通知を行なうことにより協調を促し，一貫性を保つことが可能となる．

### 2.5.4 特定の領域に対する操作の取り消し

ここまで述べてきたジョブの実行制御モデルは，すべての作業者が同時に作業を行なっており，システムからの通知に対して即座に対処できると仮定している．しかし，特に文書作成に代表される協調作業では，作業者間の関係が疎であるために，作業者同士で協議した上でデータ項目の変更を行なわなければならない場合などに，すぐに協議ができるとは限らず，その場合は待ち状態に入るか暫定的な作業を進めるしかない．暫定的に作業を進行させた場合，その作業は後の協議によって認められる場合もあれば，その作業を取り消さなければならない場合もある．したがって，一連の作業を一貫性を保ったまま取り消すことができることは非常に重要なことである．ここでは，取り消しを行ないたい領域を与えることにより，一貫性を保ったまま取り消しを行なう方法について述べる．

ジョブの性質より，ジョブはすべての領域一貫性制約を満たすような最小限の作業であることから，あるデータ項目に対して行なわれた書き込み操作を取り消す場合には，その操作を行なったジョブを取り消す必要がある．あるジョブ $J_u$ を取り消す場合，$J_u$ が書き込みを行なったデータ項目が，他のジョブ $J_v$ によって変更されているならば，ジョブ $J_u$ だけを取り消しただけではデータベース内の一貫性は保つことができず，ジョブ $J_v$ の取り消しが必要である．そこで，まずこのようなジョブの依存関係について定義を行なう．

**定義 6** 2つのジョブ $J_u, J_v (u \neq v)$ の間には，ジョブの完了時刻に関して，順序関係が存在する．$J_u$ が $J_v$ に先行すると仮定するとき，$w(x) \in OP_{J_u} \wedge w(x) \in OP_{J_v}$ を満たす $x$ が存在するならば，$J_v$ は $J_u$ に依存しているといい，この関係を $J_u \gg J_v$ と表す．さらに，もし $J_u \gg J_v \gg \cdots \gg J_w$ の関係が成り立っているならば，$J_w$ は $J_u$ に依存しているといい，この関係を $J_u \gg^+ J_w$ と表す． □

この依存関係を用いて，あるデータ項目の集合 $X$ に対する書き込み操作の取り消しの要求を以下の方法で処理する．

**方法 5** ジョブ $J$ は，ジョブ内で書き込み操作を行なったすべてのデータ項目につ

いて，書き込みを行なう前の状態を保存しており，その書き込み前の状態を書き込む操作系列を$\overline{OP_J}$と表すことにする．

取り消し要求がジョブ$J_u$から出されたとき，次の処理を行なう．

1. データ項目の集合$X$と共通部分をもつデータ項目の集合$X'$に対して最後に書き込みを行なったジョブ$J_v$を探す．
2. ジョブ$J_v$に依存しているジョブの集合$JS = \{J_w | J_v \gg^+ J_w\}$を求める．
3. $JS$に含まれるジョブ$J_w$について，$J_w$の完了時刻が新しいものから順に，$J_w$を取り消す操作の集合$\overline{OP_{J_w}}$をジョブ$J_u$の操作として実行する．
4. 取り消し操作によって変更されたデータ項目と，どのように変更されたかを$J_u$の実行者に通知する． □

ジョブの取り消しは，あるジョブの中の操作列として実行される．通常取り消し操作は多段階にわたって利用できる方が望ましい．これはどのジョブ$J_w$まで取り消しを行なったかを記録するポインタを，取り消しを実行するジョブ$J_u$ごとに用意することにより容易に実現できる [PK92]．

**例 3** 図2.2(1)(2)(3)は，9個のデータ項目からなるデータベースにおいて，それぞれジョブ1，ジョブ2，ジョブ3が順に完了した状態を表している．ここでデータ項目dで行なわれた操作を取り消す場合，ジョブ1を取り消す必要がある．ジョブ1を取り消すためにはデータ項目hとiについてジョブ1に依存しているジョブ3を取り消す必要がある．したがって，まずジョブ3を取り消し（同図(4)），次にジョブ1を取り消すことにより，ジョブ2だけが実行された状態が得られる（同図(5)）． □

## 2.6 COCOA System の実装

本節では，前節までで述べた協調作業支援のためのデータ管理手法をテキストファイルを対象とする協調作業支援に適用し，設計・実装を行なった試作システムであ

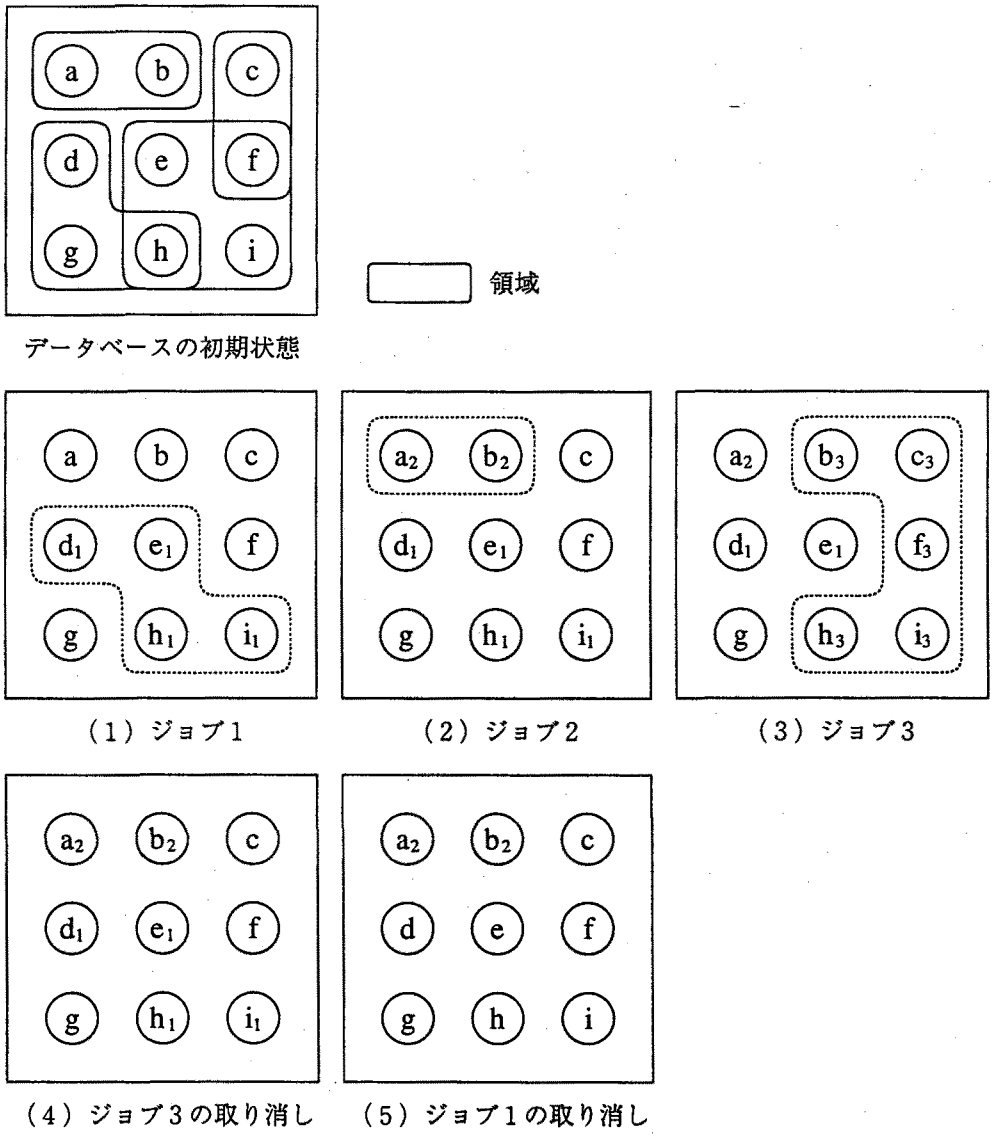



図 2.2: ジョブの実行と取り消し

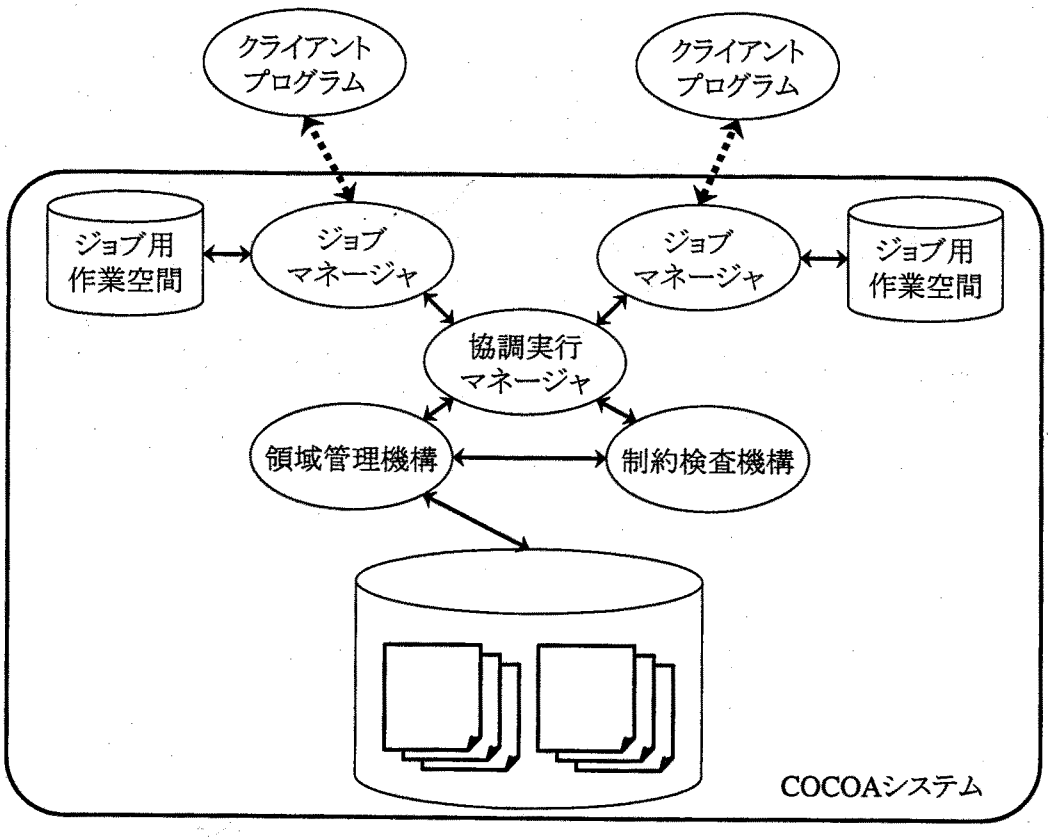



図 2.3: COCOA System の構成

る，COCOA システム (COmmon COoperative work Assistant System) について述べる．

実装した COCOA システムの構成を図 2.3に示す．このシステムは，ジョブマネージャ，協調実行マネージャ，領域管理機構，制約管理機構の 4 つの部分からなる．それぞれの役割は次の通りである．

### 2.6.1 ジョブマネージャ

COCOA システムでは，計算機ネットワークを介して開始を要求された個々のジョブに対して，ジョブマネージャとジョブ用作業空間を割り当てる．ジョブマネージャは，ジョブが生成されてから終了するまでの間，共有データベースおよびジョブ用

に割り当てられた作業空間へのアクセスを管理し，サブジョブの生成およびその管理を行なう．具体的には，ジョブマネージャは以下の機能を提供する．

- ジョブが要求する書き込み操作を，ジョブのコミット時にまとめて実行する．

  前節までで述べたように，ジョブがデータベースの一貫性を保持するための一つの条件は，ジョブが読み出すデータが一貫性を満たした状態であることである．ジョブはデータベースの一貫性を満たす最小限の作業単位であるため，ジョブが書き込み操作を共有データベースに対して行なうと，そのジョブがすべての書き込み操作を終えるまで他のジョブがデータの読み出しを待たされることがある．ジョブマネージャは，ジョブが要求する書き込み操作を共有データベースに対して行なうのではなく，そのジョブ用の作業空間に対して行ない，そのジョブのコミット時にまとめて共有データベースに対して書き込み操作を行なうことにより，この待ち時間をなくす．

- 作業空間を共有するサブジョブの実行を提供する．

  データ項目の書き込みによって，領域一貫性制約を満たすために他のデータ項目の書き換えが必要になることがある．この書き換えを担当する他の作業者に委任する方法としてサブジョブの実行を提供し，複数の作業者の参加による協調的な一貫性制約の保持を可能にする．

### 2.6.2 協調実行マネージャ

協調実行マネージャは，一般的な協調作業の進行過程に見られるようなメッセージ交換と協議を支援する部分である．協調実行マネージャは，ジョブマネージャから送られる共有データベースへのアクセス要求，および，領域予約の要求を以下のように処理する．

- データ項目の書き込み要求に対して，そのデータ項目を読み出しているジョブを担当している作業者，および，そのデータ項目に依存するデータ項目を作成した作業者に対して，データ項目がどのように更新されたかを通知する．

協調作業においてデータを共有する場合，共有されるデータ項目間に参照関係があることが多い．例えば他の作業者が作成した図や表を引用して文章を書く場合などである．このようなデータ項目の依存関係は，次に述べる領域管理機構や制約検査機構で導出することができるが，その内容が意味的に一貫しているかどうかはシステム側で判定することは当然できない．したがって，システム側では関係する作業者に対して変更通知を行なうことにより，データ項目間の関係が意味的に一貫したものになるよう支援する．

- データ項目に対する領域予約の要求に対して，そのデータ項目を読み出している作業者，および，そのデータ項目に依存するデータ項目を作成した作業者に対して，領域予約が行なわれたことを通知する．

  協調作業では，他の作業者に影響を及ぼすような書き込み操作に対して，書き込み後に通知するばかりでなく，書き込み前にも影響が出ることを予告することが望ましい．協調実行マネージャは，各データ項目がどのジョブで読み出されているかを管理し，次にのべる領域管理機構および制約検査機構からの情報を利用してどの作業者に通知を行なうかを決定する．

- データ項目の読み出し要求に対して，そのデータが領域予約されていれば，それを読み出し要求を出したジョブを担当する作業者に通知する．

### 2.6.3 領域管理機構

テキストを扱う協調作業において，一般にアクセスは文書内のある範囲（文字単位あるいは行単位）に対して行なわれる．この点を考慮し，領域管理機構では以下の管理を行なう．

- テキスト内のある範囲をデータ項目として定義する．

  一般には，アクセスの単位としてファイルを考えることが多い．しかし，例えば対象が文書ファイルである場合，それぞれの章や節，図，表がデータ項目にあたり，ファイル全体はそれらのデータ項目の並びであると考えることができ

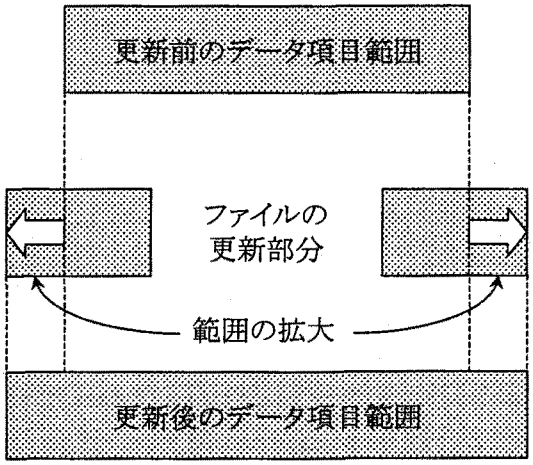



図 2.4: 境界編集時のデータ項目範囲の更新

る [OY92]. 領域管理機構では，このようなデータ項目単位でのファイルへのアクセスを可能にするために，データ項目のファイル内の位置および大きさを管理する.

データ項目に対応するファイル内の位置と大きさの情報は，そのファイルに対する書き込み前と書き込み後の差分情報を用いて更新する．ただし，そのデータ項目に対応する範囲の境界部分が編集された場合，新しい境界の判定が不可能になるため，図2.4に示すように，少なくとも元の範囲を含むような形で範囲の拡大を行なう.

- 書き込み操作を行なう際，その書き込み前の状態をジョブと結び付けて保存する.

  これにより，2.5.4節で述べた特定領域に対する操作の取り消しを実現する.

### 2.6.4 制約検査機構

制約検査機構は，各領域に課せられた一貫性制約条件を検査する機構である．具体的には以下の処理を行なう.

- 領域間の参照関係を記録し，参照に関する一貫性制約を検査する.

領域一貫性制約は，参照関係にあるデータ項目間に課せられた「参照先が存在しなければならない」という制約と，それ以外の制約に大別することができる．参照関係に関する制約は，その関係をシステムで管理しておくことにより検査できる．

- 参照関係以外の領域一貫性制約について，その制約を検査するプログラムを起動する．

  参照関係以外の制約は，テキスト処理に強力な能力をもつプログラミング言語が存在することを考慮して，外部プログラムで検査を行なう．制約の検査に必要な情報，例えばデータ項目の内容は，外部プログラムに引数として渡す．また，外部プログラムでは，作業者へのメッセージの送信やデータベースからのデータ項目の読み出し，領域一貫性制約の追加・削除・変更を行なう必要が出ることがある．制約検査機構は，このような処理を行なうためのインタフェースを提供する．

### 2.6.5 実際の協調作業への応用例

COCOAシステムは，制約検査機構における外部プログラムを適切に設計することにより，さまざまな協調作業に応用できる．ここでは，共同文書作成と共同プログラム開発の例を示す．

#### 2.6.5.1 共同文書作成

複数の執筆者が協調して文書を作成する環境では，以下のような制約が考えられる．

1. ページ数に関する制約

   各執筆者ごとに，ページ数の下限・上限が決められている．

2. 用語統一に関する制約

   文書全体で用いる用語は統一されなければならない．また，キーワードの初回出現部分では太字で記述しなければならない．

3. 図表の参照関係に関する制約

本文からの図表の参照は，図表の本体が存在しなければ行なえない．

以上のような制約は，以下のように管理することができる．

1. ページ数に関する制約

各執筆者が担当する部分のページ数を計算し，制約条件に合致していなければ担当執筆者に警告の通知を行なうような外部プログラムを作成する．例えば，プレーンテキスト形式の文書であれば行数からページ数を得ることができる．また，LaTeX 文書であれば，LaTeX 処理系で処理した結果得られるログ情報からページ数を得ることができる．

2. 用語統一に関する制約

統一すべきキーワード（推奨用語と類似用語）と初回出現位置を管理するファイルを（データ項目の一つとして）用意する．用語統一に関する制約を検査する外部プログラムでは，ある執筆者が担当部分の編集しコミットしようとしたとき，推奨用語と類似した用語が使用されていないか，および，文書全体の中での各キーワードの初回出現位置が変わっていないかを検査し，そのような場合には担当の作業者に通知を行なう．

3. 図表の参照関係に関する制約

図表の参照関係を制約検査機構に登録することにより，参照関係に関する制約を検査することができる．

#### 2.6.5.2 共同プログラム開発

共同プログラム開発では，プログラムの部品（モジュール）をプログラマが分担して開発していくことが一般的である．共同プログラム開発では，以下のような制約が考えられる．

1. 関数の整合性に関する制約

関数呼び出しにおいては，引数の個数や型がその関数の定義と一致していなければならない．

2. ドキュメントの整合性に関する制約

関数の仕様を記述したドキュメントと，実際の関数の仕様が一致していなければならない．

3. 実行可能性に関する制約

作成されたソースコードからは，実行可能なオブジェクトが得られなければならない．

以上のような制約は，以下のように管理することができる．

1. 関数の整合性に関する制約

外部プログラムにより，実際にコンパイルを行ないコンパイラからの出力結果より検査する．また，関数の定義部分が変更されたときに，その関数を利用しているモジュールの開発担当者に通知できるように，呼び出し部分から定義部分への参照関係を制約検査機構に登録する．

2. ドキュメントの整合性に関する制約

関数の仕様を記述したドキュメントから，実際の関数定義部分への参照関係を制約検査機構に登録することにより，関数定義が変更された際に対応するドキュメントの更新を忘れることを防ぐ．

3. 実行可能性に関する制約

外部プログラムにより実際にコンパイルを行ない，コンパイラからの出力結果を用いて正しくコンパイルされるかどうかを検査する．

## 2.7 むすび

本章では，協調作業のためのデータ管理手法を提案した．まず，共有データベースに課せられている一貫性制約を明示的に定義できるように，データベースに領域一貫性制約の概念を導入し，このデータベース上で作業者が行なう作業の単位をジョブとして定義した．ジョブを同時に実行するために，ジョブの協調実行の概念と領域予約の操作を導入し，領域一貫性制約を考慮することにより，作業者間で協議が必要となる場合をシステムで判定できるようにした．以上により，作業者は協議が必要な時をシステムからの通知で知ることができるようになり，その結果，作業による作業者間の影響を強く意識することなく，作業を進めることが可能となった．

また，共同作業者が作業中でないために，協議が即座に行なえず作業が中断しうる問題を重視し，協議が行なえなくとも作業を続けることができるような制御方法を提案した．特に，協議が行なわれないままで継続された作業結果は，相手の作業者に対して望ましくない結果であることが考えられる．したがって，データベースの一貫性を保ちながら，特定の領域に対する操作を取り消すための枠組みを与えることにより，作業者が望ましくない結果を取り消すことができるようになり，その結果，中断することなく作業を進めることが可能となった．

さらに，提案したモデルを用いた協調作業支援システムの実装例としてCOCOAシステムを実装することで，本モデルの有用性を示した．このCOCOAシステムでは，領域管理機構を導入することによって，テキストファイルを領域単位でアクセスすることが可能となった．このように，提案したデータベース管理手法を実際の協調作業へ応用する際には，ファイルなどの物理的なアクセス単位と，データベース管理システムにおける制御単位となるデータ項目というアクセス単位との間のマッピングを行なう機構が必要となる．

計算機を利用した協調作業は，世界的な計算機ネットワーク環境の普及によってますます重要な研究分野となっている．本章で提案した協調作業支援のためのデータベース管理手法は，さまざまな形態の協調作業支援システムにおけるデータベース管理手法の基盤として応用できると考えている．

# 第 3 章

# 通信アプリケーションのためのデータベースシステム

本章では，データベースシステムの構成要素の一つであるデータベースアプリケーションに焦点を当て，特に通信アプリケーションからのデータベースの利用を最大限に考慮したデータベース管理システムの設計手法について述べる．

データベースアプリケーションは，通常はデータベース管理システムを中心とし，そこで管理されているデータベースに格納されたデータを処理するプログラムである．一方，通信アプリケーションでは，中心はあくまでも通信機能であり，データベース管理システムはデータ通信によって提供あるいは収集されるデータを格納する補助的な役割を果たすものとなる．つまり，通信アプリケーションのためのデータベース管理システムを設計する際には，アプリケーションからの要求に応えることを第一に考えなければならない．

本章では上記の点を重視したデータベース管理システムの設計手法として，特に通信アプリケーションがデータ構造記述およびデータ表現形式として利用する ASN.1 (Abstract Syntax Notation One: 抽象構文記法 1 ) を取り上げ，通信アプリケーションからのデータベース機能の利用を容易にするデータベースプログラミング言語，および，通信アプリケーション支援に適したデータ格納部の設計・構築手法を提案する．

## 3.1 まえがき

世界規模の計算機ネットワークの発展によって，データの管理は従来の集中型管理から分散型管理へ移行し，種々のデータをその管理母体に応じて分散配置することが可能になった．このような環境では，クライアントからのデータの検索要求や更新要求を処理するために，分散されたサーバ間，および，サーバとクライアント間で，要求・応答メッセージやデータの交換が必要になる．現在では，マルチメディアデータをはじめとする多種多様なデータが計算機システム間で頻繁に交換されるようになっている．

データ管理の分散化に伴い，データの安全な共有を可能にするデータベース管理システムが果たす役割は，従来にも増して重要なものとなってきている．特に，従来の分散データベースシステムのようにデータベースシステムが計算機ネットワーク環境を利用するという形態ではなく，計算機ネットワーク環境の上で動作する通信アプリケーションに対してデータベースシステムが支援するという形態でのデータベースシステムの役割が重要となってきている．このような支援を行なうデータベースシステムを設計する際には，通信アプリケーションからの要求に応えることをまず念頭におかなければならない．

計算機ネットワーク上にデータが分散化されることによって生じる大きな問題の一つは，それらのデータを管理している計算機システムの異種性である．つまり，ある計算機で管理されているデータは，通常，そのシステムでのデータ表現を理解できる計算機以外では意味をなさないということである．したがって，異種のシステムに分散したデータをやりとりする通信アプリケーションでは，当然この問題に対処しなければならない．その方法の一つが，ネットワーク上でのデータ表現を，計算機システムのアーキテクチャに依存しない形で統一することである．

ネットワーク上でのデータ表現形式として代表的なものに，RPC (Remote Procedure Call, 遠隔手続き呼び出し) のために開発されたデータ表現形式である XDR (eXternal Data Representation) や，国際標準化機構 (International Organization for Standardization, ISO) で定められた ASN.1[ISO87a, ISO87b] がある．この二つの

ツールでは，どちらもネットワーク上でやりとりするデータの構造を専用のデータ構造記述言語で記述する．そして，それを処理系で処理することによって，計算機内部でのデータ構造定義（通常，C 言語や C++言語のデータ構造定義），および，計算機内部でのデータ表現とネットワーク上でのデータ表現との間の相互変換ルーチンがソースコードの形式で生成される．一般に，XDR や ASN.1 を利用する通信アプリケーションは，このような方法によって生成されたソースコードを利用して構築されている．

本章では，特に ASN.1 に焦点をあて，ASN.1 を利用する通信アプリケーションを支援するデータベースシステム（以下，ASN.1 データベースシステムと呼ぶ）の設計・構築手法に関して論じる．

以下，3.2節では ASN.1 の概要について説明する．3.3節では，ASN.1 データベースシステムに求められる要件に関して議論する．3.4節では，ASN.1 データベースシステムにおいてアプリケーションの記述を容易にするデータベースプログラミング言語 ASN.1/PL について述べる．3.5節では，オブジェクト指向データベース管理システム (Object-Oriented Database Management System, OODBMS) を用いた ASN.1 データ格納部の実現方法について述べる．3.6節では，さらに通信アプリケーションに要求されるパフォーマンスを実現するためのデータ格納方式について議論する．最後に 3.7節で本章のまとめとする．

## 3.2 ASN.1 の概要

本節では，国際標準化機構で定められた国際標準である ASN.1 について概説する．ASN.1 は，OSI (Open Systems Interconnection) の基本参照モデルにおけるアプリケーション層の間で送受されるデータの構造を計算機のアーキテクチャに依存しない形で定義するためのデータ構造記述言語 [ISO87a]，および，定義されたデータ構造をもつデータのネットワーク上での表現形式を規定する基本符号化規則 [ISO87b] の二つの国際標準により規定されている．

### 3.2.1 ASN.1の利用目的

異なるアーキテクチャの計算機システム間では，例えば整数値を表現するためのバイト数やワード内でのバイトオーダ，さらには構造体や配列などの構造をもったデータの内部表現が異なるため，内部表現形式のままでは正しくデータを送受することができない．そこで，まず計算機システム間で送受するデータの構造をASN.1のデータ構造記述言語を用いて定義する．実際にデータ送受信を行なう際にはASN.1の基本符号化規則を適用し，送信側では送信側の計算機の内部表現形式からASN.1のデータ表現形式へと符号化してデータを送信し，受信側では受信したASN.1のデータ表現形式でのデータを受信側の計算機での内部表現形式に復号化する．このような方法によって，マシンのアーキテクチャの違いを吸収し，曖昧さのないデータの交換を行なうことが可能になる．

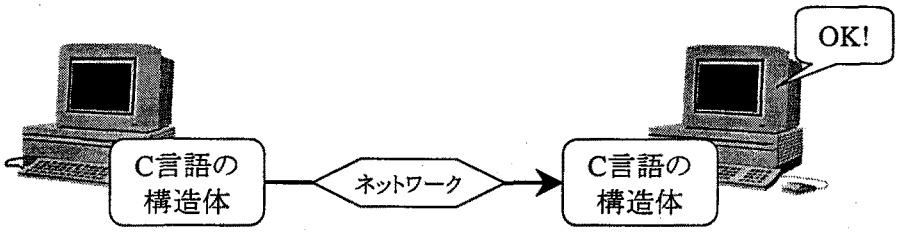
図3.1は，ASN.1の利用形態を示したものである．同図(a)は同じアーキテクチャをもつ計算機間でのASN.1を利用しないデータの送受を表したものであり，受信側は正しくデータを解釈することができる．同図(b)は異なるアーキテクチャをもつ計算機間でのデータの送受を表したものであり，この場合にはデータの内部表現の違いにより受信側は正しくデータを解釈することができない．同図(c)はこの問題を解決するためにASN.1を利用した場合を表したものである．この図のように，ASN.1を利用したデータ送受信では，計算機ネットワークに送出されるデータはアーキテクチャに依存しない形式となるため，どのようなアーキテクチャの計算機でもASN.1による復号化さえ行なえば受信したデータを正しく解釈することができる．

### 3.2.2 ASN.1によるデータ構造定義

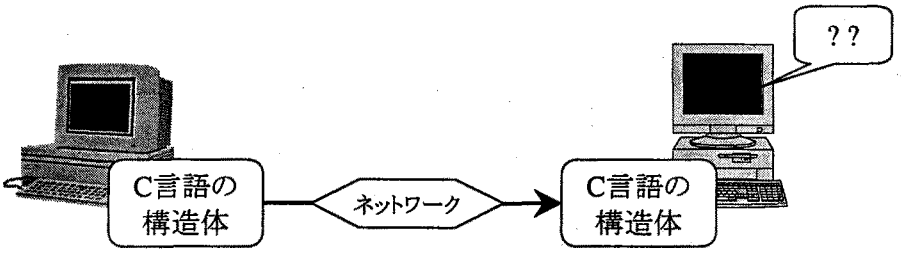
#### 3.2.2.1 モジュール

ASN.1におけるモジュールとは，通信アプリケーションで交換されるデータの型定義，ならびに，共通して利用される定数の定義を記述したものである．一つのモジュールは，次のような記述によって定義される．

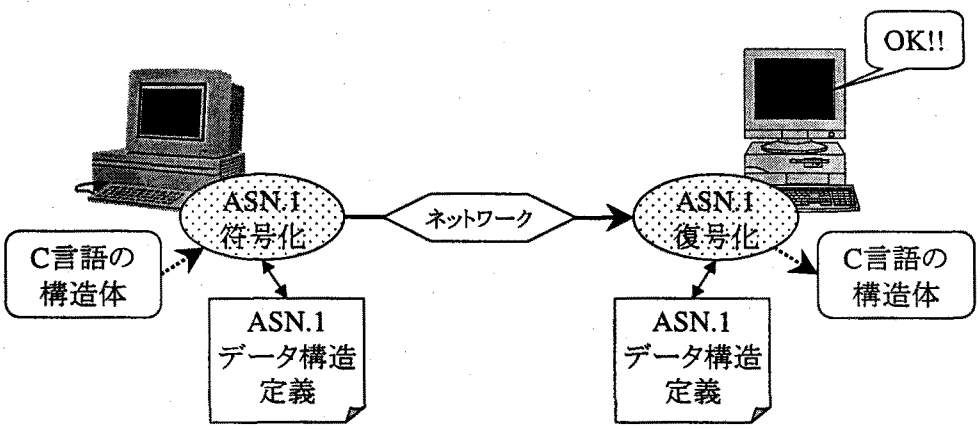
```
<<module>> DEFINITIONS ::=
```




(a) 同機種間でのデータ送受信 （成功）




(b) 異機種間でのデータ送受信 （失敗）




(c) 異機種間での ASN.1 を利用したデータ送受信 （成功）

図 3.1: ASN.1 の利用形態

```
BEGIN
<<linkage>>
<<declarations>>
END
```

<<module>>の項には，このモジュールに付けられた名前を記述する．単なるテキスト表現であるモジュール名に加え，モジュールを一意に識別するためのオブジェクト識別子をオプションとして指定することができる．

<<linkage>>の項には，他のモジュールとの関係を記述する．これは，このモジュール内で定義されるオブジェクト（データ型や定数）の中で他のモジュールからの利用を許可するものを宣言する部分（EXPORTS 節）と，他のモジュールで定義されているオブジェクトの利用を宣言する部分（IMPORTS 節）からなる．

<<declarations>>の項は，実際の ASN.1 定義を記述する部分である．ここに，データ型や定数の定義を記述する．

図 3.2に，モジュール定義例を示す．このモジュールは，人に関する情報を表すデータ構造を定義するものであり，6 つのデータ型を定義している．

#### 3.2.2.2 データ型の定義

この節では，ASN.1 が提供するデータ型について詳しく述べる．

**単純型**

単純型は，ASN.1 において扱うことが可能な基本的なデータ型である．ASN.1 では多くの単純型が定義されているが，その一部を以下に列挙する．

- BOOLEAN 型： TRUE あるいは FALSE のいずれかの値をとるデータ型．
- INTEGER 型： 整数を値とするデータ型．その大きさには制限がない．
- ENUMERATED 型： 与えられた値の集合から 1 つをとるデータ型．
- REAL 型： 任意の精度の実数を値とするデータ型．

```
PersonalRecords DEFINITIONS ::=
BEGIN
Person ::= SEQUENCE {
        name        PrintableString,
        age         INTEGER,
        sex         ENUMERATED { male(0), female(1) },
        belong      Belong }
MarriedPerson ::= SEQUENCE {
        COMPONENTS OF Person,
        spouse      PrintableString,
        children    Children OPTIONAL }
Children ::= SET OF PrintableString
Belong ::= CHOICE {
        univ    [0] University,
        comp    [1] Company,
        null    [2] NULL }
University ::= SEQUENCE {
        name        PrintableString,
        faculty     PrintableString OPTIONAL }
Company ::= SEQUENCE {
        name        PrintableString,
        dept        PrintableString OPTIONAL }
END
```

図 3.2: ASN.1 によるモジュール定義の例

- BIT STRING 型： 0個以上のビットを値とするデータ型.
- OCTET STRING 型： 0個以上のオクテットを値とするデータ型.
- OBJECT IDENTIFIER 型： 世界のあらゆるものを曖昧性なく識別するための統一的な手法が定められており [ISO91]，それによって与えられた名前（オブジェクト識別子）に対する参照を表す型.

**構造型**

構造型は，単純型を結合して複雑なデータ型を構成するものである．また，これを再帰的に繰り返し，任意の深さの入れ子構造を構造型として構成することが可能である．構造型として，以下のデータ型構成子が提供されている.

- SEQUENCE 型: 0個以上のフィールド（データ型の要素）のリストを表すためのデータ型. 各フィールドには，値をもたなくてもよいことを表すOPTIONAL指定や，値が省略されたときの既定値を表すDEFAULT指定を付加することができる.
- SEQUENCE OF 型： 0個以上の同じデータ型の要素のリストを表すためのデータ型.
- SET 型： 0個以上のフィールド（データ型の要素）の集合を表すためのデータ型. 各フィールドには，値をもたなくてもよいことを表すOPTIONAL指定や，値が省略されたときの既定値を表すDEFAULT指定を付加することができる.
- SET OF 型： 0個以上の同じデータ型の要素の集合を表すためのデータ型.

**メタ型**

メタ型は，単純型でもなく構造型でもない「メタ」なデータ型であり，以下の二つがある.

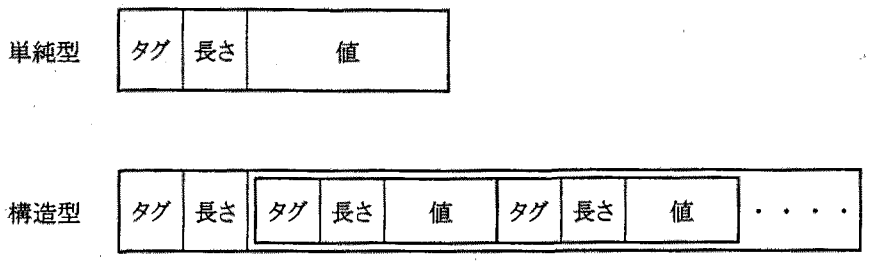



図 3.3: ASN.1 による符号化

- CHOICE 型： 一つ以上のデータ型の中から，一つのデータ型を選択してその型とするデータ型.

- ANY 型： 任意の ASN.1 のデータ型をとることができるデータ型.

### 3.2.3 ASN.1 の符号化規則

3.2.2.2節で述べた ASN.1 のデータ型をもつ値（以下，ASN.1 データと呼ぶ）を，計算機ネットワークを介して送受するデータ表現に符号化するための規則が，基本符号化規則として定められている．この規則による符号化は TLV (Tag, Length, Value) 符号化と呼ばれ，その構造は図 3.3 のようになる．符号化の原理は次のようなものである．

1. タグフィールドには，データ型を識別するための値（タグ）を入れる．

2. 長さフィールドには，値フィールドの長さをオクテット単位で入れる．

3. 値フィールドには，実際に符号化した値を入れる．

4. 構造型の場合，値フィールドはさらにタグ，長さ，値の組が入る．これを再帰的に繰り返すことにより，任意の入れ子構造をもつデータを符号化することができる．

## 3.3 ASN.1データベースシステムの要件

本節では，ASN.1を利用する蓄積型通信アプリケーションの開発環境である，ASN.1データベースシステムに要求される要件について議論する．

図3.4は，従来のASN.1アプリケーションの開発手法を蓄積型ASN.1アプリケーションの開発に適用した場合の概念図である．この図は，特にデータ表現に着目してデータの流れを表したものである．上位層は計算機ネットワークを表しており，ここではASN.1の符号化規則によるデータ表現が用いられる．中間層はプログラミング言語でのデータ表現であり，ここでは計算機内部でのデータ表現が用いられる．下位層はデータベース管理システムを表しており，ここではデータベース管理システム内でのデータ表現が用いられる．

一般にASN.1アプリケーションの開発では，ASN.1の符号化規則によるデータ表現と，計算機内部でのデータ表現の二種類のデータ表現をプログラム中で扱わなければならない．さらに，データの蓄積・管理のために関係データベース管理システムやオブジェクト指向データベース管理システムなどの既存のデータベース管理システムを利用した場合，上記の二種類のデータ表現に，データベース管理システムが扱えるデータ構造も加わるため，全部で三種類のデータ表現を扱う必要がある．

このような開発手法では，二つの重要な問題が生ずる．第一の問題は，スキーマ変換に関する問題である．文献 [Masu93a, Masu93b] では，ASN.1アプリケーションの一つであるODA (Open Document Architecture) に基づくマルチメディア文書をオブジェクト指向データベースに格納する方法について議論しているが，ここでの議論が暗示していることは，ASN.1で構造が定義されたODA文書を既存のデータベース管理システムを利用してデータベース化するためには，ASN.1を用いて定義されたODA文書のデータ構造をデータベース管理システムが扱えるデータ構造に変換する必要があるということである．さらに，このような変換が必要なために，格納されたODA文書を操作するアプリケーションを記述する際には，それがどのように変換されたかを常に考えておかなければならない．つまり，次のような問題の考慮が必要となる．

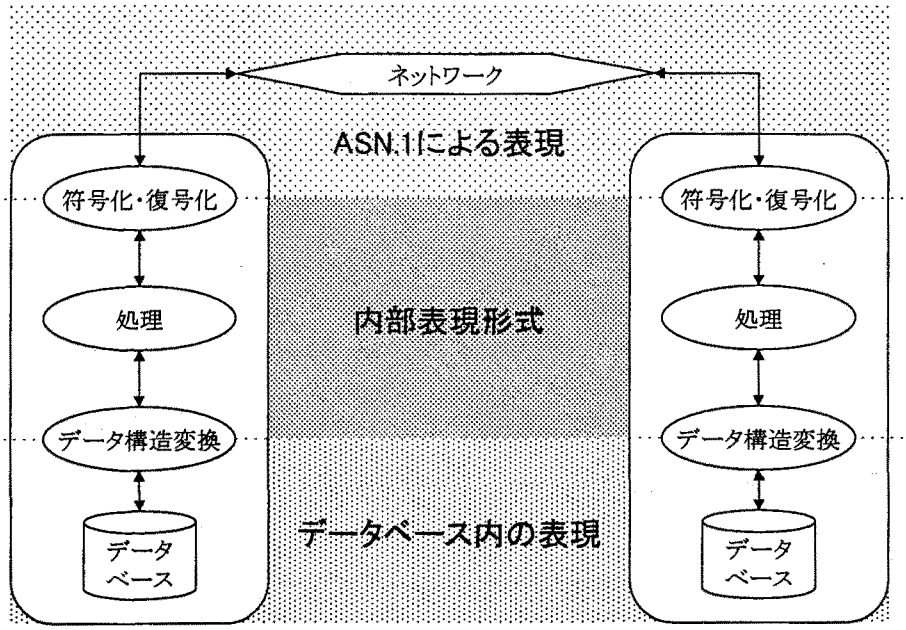



図 3.4: 従来の蓄積型 ASN.1 アプリケーションの構造

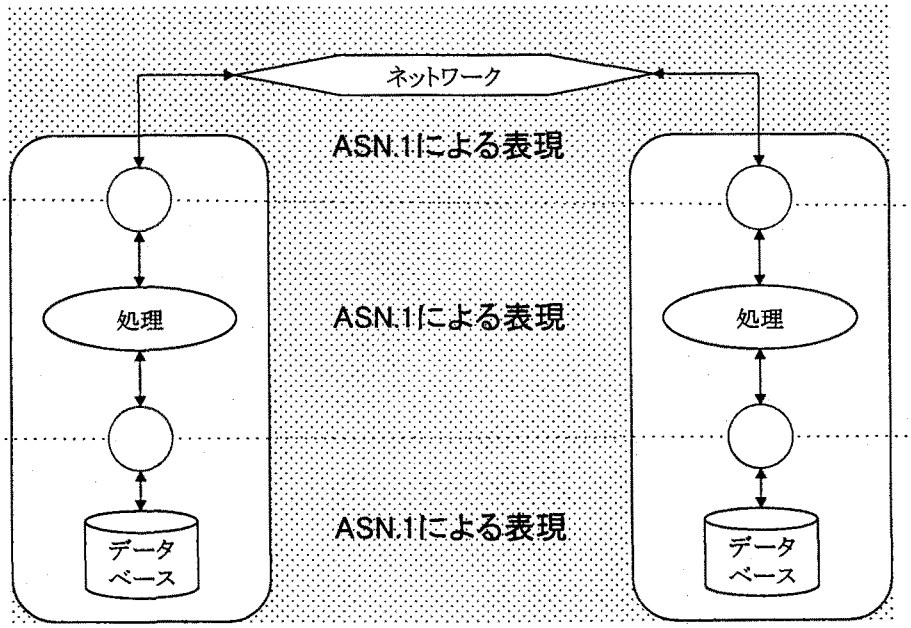



図 3.5: 提案する ASN.1 データベースシステムの概念

1. ASN.1 で記述されたデータ構造定義を，使用するデータベース管理システムにおけるスキーマ定義に変換しなければならない．また，データ構造定義能力の違いから，その変換が複雑になる．

2. ASN.1 データ格納する際には，使用するデータベース管理システムのスキーマ上でのデータ構造に変換しなければならない．逆に，データベースから検索した ASN.1 データを処理する際には，内部表現形式に変換されたデータに対する操作を記述しなければならない．

第二の問題は，インピーダンスミスマッチに関する問題である．データベースアプリケーションにおいてはデータ項目集合を対象とする操作を行なうことが多いのに対し，C 言語などの汎用プログラミング言語では集合を直観的に表現し，操作することが困難である．また，特に ASN.1 データのように複雑な構造をもつデータを扱う場合，汎用プログラミング言語ではデータ構造のナビゲーションなどの操作を行なう複雑なアプリケーションを柔軟に記述するのが困難である．さらに，このようなデータ構造の違いから，汎用プログラミング言語にデータベース操作を埋め込んでデータベースアプリケーションを記述した場合，十分な型検査が不可能であり，アプリケーションの実行時にさまざまな型誤りを引き起こす原因となっている．これらの問題は，インピーダンスミスマッチの問題と呼ばれ，従来からデータベースプログラミングにおける重要な問題として認識されている．

ASN.1 アプリケーションにおいては，データ構造がまず ASN.1 のデータ構造記述言語で定義される．したがって，上記の二つの問題を解決するためには，図 3.5に示すように，プログラミング言語のレベルで ASN.1 のデータ構造を直接扱えることが望ましい．さらに，データベース管理システムでも ASN.1 のデータ構造を直接扱えれば，データ構造の変換の必要性がなくなりプログラミングが容易になる．また，インピーダンスミスマッチの問題も解消される．

さらに，一般に蓄積型 ASN.1 アプリケーションでは，外部からのデータ検索やデータ更新の要求メッセージに対してできるだけ高速に応答を返す必要がある．これに対し，ASN.1 の符号化規則は汎用性を重視しているため，符号化・復号化処理

は時間のかかる処理となる．そこで，符号化・復号化処理をできるだけ省いて高速にデータの検索・更新およびデータの送信を行なえることが望まれる．

以上の議論をまとめると，ASN.1 データベースシステムの要件は以下のようになる．

**要件 1** ASN.1 のデータ構造定義をそのままデータベースにおけるスキーマ定義とする．

**要件 2** ASN.1 データを操作対象とする型検査が可能なデータベースプログラミング言語を用いて ASN.1 データを操作するアプリケーションを記述することによって，複雑なデータ操作の記述を容易にし，データベースプログラミングにおける型誤りを排除する．さらに，アプリケーションのアーキテクチャ非依存性を実現する．

**要件 3** ASN.1 で符号化されたデータをそのままデータベースに格納する．

以下の節では，これらの要件を満たすための ASN.1 データベースシステムの実現手法について論じる．

## 3.4 ASN.1/PL: ASN.1 データベースプログラミング言語

本節では，ASN.1 のデータ構造を直接操作できるデータベースプログラミング言語として設計した ASN.1/PL について述べる．

### 3.4.1 ASN.1/PL の設計

前節までで述べたように，ASN.1/PL の設計に当たっては，ASN.1 のデータ構造を直接扱えるようにすることによってデータ構造の変換を意識することなくプログラムを記述できるようにし，これによってインピーダンスミスマッチの問題を解決することを第一に考慮した．

さらに，プログラミング言語において，多相性をもつ関数が有効であることは様々な場面で実証されている．例えばオブジェクト指向プログラミング言語においては，同名メソッドがそれを受けたオブジェクトのクラスによって異なる動作をとるというメソッドの多相性が利用されている．データベースプログラミングにおいては，対象となるデータの構造が複雑になるほど多相性の概念が特に有効かつ重要であり，また，本質的であることが示されている [AB87]．ODA など多くのアプリケーションにおいて，ASN.1 で記述されたデータ構造は非常に複雑になっていることから，ASN.1/PL に多相性の概念を採り入れるのは有用であると考える．

この考えに基づく言語として，多相性をもつ関数型言語である ML をデータベース操作に適するように拡張した汎用データベースプログラミング言語 Machiavelli が提案されている [BO96, OBB89]．また，Machiavelli は ML からの拡張において型推論アルゴリズムもデータベース操作に適するように拡張しており，これにより前述のインピーダンスミスマッチの問題も解決している．ASN.1/PL の設計では，この言語の考え方を採り入れている．

### 3.4.2 ASN.1/PL の構文要素

ASN.1/PL のプログラムは，関数宣言および変数宣言の列からなる．

#### 3.4.2.1 型

ASN.1/PL では，ASN.1 によって提供されている組み込み型，および，ASN.1 のモジュール内で定義されたユーザ定義型を扱う．また，プログラム内で新たな型を生成することもできる．プログラム中で型を指定する場合は，ASN.1 の型名を記述する．

#### 3.4.2.2 値

ASN.1/PL での値は，すべて ASN.1 の型をもつ．値の記述は，ASN.1 の国際標準で定義されている値記法で記述する．

#### 3.4.2.3 変数

変数は，あるデータを参照するために使用するものであり，参照しているデータの型をもつ．変数の型はその宣言時に決定され，それと異なる型の値を代入することはできない．

変数宣言および変数が参照している値の更新は，それぞれ次のように記述する．

宣言: val *var_name* [: *type*] = *expression*

更新: *var_name* := *expression*

変数の型 *type* は，右辺の式から推論できるため省略することができ，指定された場合には型の整合性が検査される．変数は，一般の汎用言語と異なり，すべてある値への参照を表す．したがって，変数が永続的な値を参照している場合には，その変数の値の更新はデータベース内に格納されている値を更新することになる．

また，特別に ASN.1 の型名を変数として記述することができる．これはデータベースに格納されているその型の値の集合（SET OF 型をとる）を参照するために用いる．

#### 3.4.2.4 関数

関数は，ASN.1/PL のプログラムの主体となるものであり，アプリケーション開発者は関数の組み合わせとしてプログラムを記述する．

関数宣言および関数適用は，それぞれ次のように記述する．

宣言: fun *fun_name* ([*arg1* [: *type1*], ...]) [: *type*] = *expression*

適用: *fun_name* ([*arg1*, ...])

引数の型および返り値の型は，右辺の式から推論可能な場合は省略することができる．これによって，多相性をもつ関数の宣言が可能になる．例えば，関数 nameof が

```
fun nameof(x) = x.name
```

と宣言されている場合，この関数は name を要素としてもつ任意の SEQUENCE 型，SET 型，CHOICE 型に対して適用可能である．

また，ASN.1/PL は，ユーザとのデータの入出力を行なう関数や，ネットワークを介したデータの送受信を行なう関数などの基本的な関数を，システム関数として提供する．

#### 3.4.2.5 式

値，変数，関数適用は式である．また，さまざまな計算を以下に挙げる式で表現する．すべての式は，何らかの型をもつ．

**演算子**

すべての型に対して，その値の等価性を判定する演算子を使用できる．また，ASN.1 の単純型に対しては基本的な演算子を使用できる．SEQUENCE OF 型と SET OF 型をもつ式に対しては，ある値がその集合値に含まれるかどうかを判定する in 演算子を使用できる．CHOICE 型の値に対しては，その選択を判定する isa 演算子を使用できる．

**航行式**

ASN.1 の型は複雑な入れ子構造をもつことがある．この構造をたどるためには，ドット記法を用いて次のように記述する．

*expression.field_name*

これは，*expression* で表される SEQUENCE 型，SET 型，CHOICE 型の値から，その構造内の *field_name* で表される要素を取り出す．

**条件式**

条件式として，次の構文で示される if 式と case 式が使用できる．

- if *expression1* then *expression2*
  else *expression3*
- case *expression1* of
  [*rel_op*] *expression2* => *expression3*,

⋮

[else => *expression4*]

end

case 式における *rel_op* は in や isa を含む比較演算子であり，省略された場合には "=" として扱う．

### 反復式

ASN.1/PL は次の二種類の反復式を提供する．これらの反復式は特別な型 unit をもつ．

- while *expression1* do *expression2*
- foreach (*l_var*, *expression1*) *expression2*

while 式は，BOOLEAN 型をもつ *expression1* が TRUE である間，*expression2* の計算を行なう．foreach 式は，SEQUENCE OF 型あるいは SET OF 型をもつ *expression1* の中から要素を一つずつ取り出して *l_var* がそれを参照し，*expression2* の評価を行なう．

### 構造化問合せ式

SQL の宣言的な問合せ記述をとり入れ，ASN.1/PL では構造化問合せ式を次のように記述する．

select *expression* [, ...]
from *var_name* <- *expression* [, ...]
where *expression*

from 節の *expression* は SEQUENCE OF 型あるいは SET OF 型をもつ式であり，問合せの対象となる集合を指定する．where 節の *expression* は BOOLEAN 型をもつ式であり，この式の値が TRUE となる要素が選択される．select 節の *expression* では，from 節に記述された変数名を含む任意の式を記述することができる．

### 3.4.3 プログラム例

図3.2のモジュールで定義されたPerson型のデータが格納されたデータベースに対する，ASN.1/PLで記述されたプログラム例を図3.6に示す．このプログラムは，ユーザからコマンドを文字列として受けとり，それに従って次の動作を行なう．

`list` データベースからすべてのPerson型のデータを取り出し，その名前のリストを表示する．

`univ` データベースから所属が大学であるPerson型のデータを取り出し，その名前のリストを表示する．

`comp` データベースから所属が企業であるPerson型のデータを取り出し，その名前のリストを表示する．

`quit` プログラムを終了する．

**その他** コマンドをPerson名とみなし，入力された文字列に一致する名前をもつPerson型のデータの内容をすべて表示する．

### 3.4.4 ASN.1/PL処理系

ASN.1/PLで記述されたプログラムの処理は二段階からなる．まず，プログラム全体に対して型の整合性検査を行なう．これは，データを蓄積するために実際に用いるデータベースシステムに依存しない部分である．次に，実行可能な形式にプログラムの変換を行なう．この変換処理は，使用するデータベース管理システムに依存する．そこで，プログラムの変換処理については，UNIX上のC++ベースのOODBMSであるObjectStoreを用いて実装したASN.1データベースシステム（3.5節参照）でのASN.1/PL処理系について述べる．

```
fun univPersons() =
    select  x
    from    x <- Person
    where   x.belong isa University;
fun compPersons() =
    select  x
    from    x <- Person
    where   x.belong isa Company;
fun namedPersons(name) =
    select  x
    from    x <- Person
    where   x.name == name;
fun main() = {
    val command = inputOcts();
    while not command == "quit" do {
        case command of
        "list" => foreach(p, Person)
                      output(p.name);
        "univ" => foreach(p, univPersons())
                      output(p.name);
        "comp" => foreach(p, compPersons())
                      output(p.name);
        else   => foreach(p, namedPersons(command))
                      output(p);
        endcase;
        command := inputOcts();
    }
}
```

図 3.6: ASN.1/PL によるプログラム例

#### 3.4.4.1 ASN.1/PL の型推論と型検査

ASN.1/PL では，プログラミングの簡便化のために，変数や関数の宣言時に必ずしも型を宣言する必要がないように設計した．したがって，それらの変数や関数の型を決定するために，型推論が必要である．また，ASN.1/PL で記述されたプログラムの処理においては，対象となる ASN.1 のデータ構造が複雑になり得ることから，型誤りの排除のためには記述されたコードの型検査を前もって行なうことが重要である．

この節では，ASN.1/PL の型推論および型検査を行なう機構として，汎用データベースプログラミング言語 Machiavelli[BO96] の型推論アルゴリズムが利用可能であることを示す．

**言語 Machiavelli の型と式**

Machiavelli は，ML[Paul91] をデータベースプログラミングのために拡張した型推論システムをもつ言語であり，プログラム内の様々な式に対して型推論および型検査を行なう機構をもつ．

Machiavelli で扱うことができる型は，データベースシステムが扱う基本型に加え，様々な型を扱うことができる．以下に，その一部を挙げる．

- $[\, l_1 : \tau_1, ..., l_n : \tau_n \,]$: フィールド $l_1, ..., l_n$ （それぞれの型は$\tau_1, ..., \tau_n$）からなるレコード型．
- $< l_1 : \tau_1, ..., l_n : \tau_n >$: フィールド $l_1, ..., l_n$ （それぞれの型は$\tau_1, ..., \tau_n$）からなる選択型．
- $\{\tau\}$: 型$\tau$の集合型

これらの型をもつ値に対して，以下に挙げるような式を扱うことができる．

- $[\, l_1 = e_1, ..., l_n = e_n \,]$: フィールド $l_1, ..., l_n$からなるレコード．
- $< l = e >$: 選択型からの一つの選択．

- $\{e_1, e_2, ...\}$: ある型の値の集合.
- $e.l$: レコード $e$ からのフィールド $l$ の選択.

Machiavelli ではこのほかにも様々な式を扱うことができ，それらに対して型付けを行なう型推論アルゴリズムが定義されている.

**Machiavelli の型推論の利用可能性**

ASN.1/PL で扱う ASN.1 の型は，Machiavelli で扱うことができる型に次のように対応づけることができる.

- 単純型: 基本型
- SEQUENCE 型，SET 型: レコード型
- SEQUENCE OF 型，SET OF 型: 集合型
- CHOICE 型: 選択型

また，変数，関数，航行式，構造化問合せ式などを組み合わせた ASN.1/PL の任意の式は，Machiavelli で扱うことができる式の表現に対応づけることができる．したがって，Machiavelli がもつ型推論アルゴリズムを ASN.1/PL の型推論機構で利用することが可能である.

#### 3.4.4.2 プログラムの変換

OODBMS では，それがサポートするプログラミング言語で記述されたプログラムしか扱うことができない．したがって，ASN.1/PL 処理系は ASN.1/PL で記述されたプログラムを C++のプログラムに変換することになる.

ASN.1/PL から C++への変換は，符号化や復号化の処理も考慮して次のように行なう.

- 変数宣言: 変数はすべて参照型であるので，ポインタ型の変数宣言に変換する.

- 関数宣言: 基本的には一つの関数定義に変換する．C++では多相的な関数を一つの関数定義で表現できないため，多相的な関数は引数と返り値の型の可能な組合せについて一つずつ関数定義に変換する．
- 関数適用: 関数呼び出しに変換する．
- 演算子: C++が提供する演算子，あるいは，ASN.1コンパイラがクラスメソッドとして出力する演算子用メソッドの呼び出しに変換する．
- 航行式: $e.l$ は，式 $e$ のフィールド $l$ が復号化されているかどうかをフラグによって判断し，復号化されていればその値を返し，復号化されていなければ復号化を行なってからその値を返すようなコードに変換する．
- 条件式: if式，case式ともにC++のif文に変換する．
- 反復式: for文に変換する．
- 構造化問合せ式: from節で指定された要素をfor文で一つずつ取り出しながらwhere節で指定された条件を評価し，それが真である場合にはselect節で指定された式を評価して結果の集合に加えるようなコードに変換する．

## 3.5 OODBMSを用いたASN.1データベース

これまでに開発されてきたASN.1アプリケーションでは，データの蓄積・管理機能はそれぞれ独自の方法で実現されてきた．具体的には，主要な方法として，(1) データを単に平坦なファイルとして管理し，その管理ルーチンをC言語などの汎用プログラミング言語で記述する方法，(2) 関係データベース管理システムを利用してデータを蓄積・管理し，ASN.1記述に基づいてスキーマを設計する方法，の二つの方法がある．いずれの方法においても，アプリケーション開発者はアプリケーションごとに異なる方法でデータの蓄積・管理部を設計する必要があり，汎用性がない．これが開発効率を下げる要因となっている．

本節では，ASN.1 で定義されたデータ構造をもつデータ（以下，ASN.1 データと記述）の蓄積・管理を行なうデータベース（以下，ASN.1 データベースと記述）を実現する基盤として，OODBMS を用いた方法を提案し，ASN.1 記述から OODBMS におけるスキーマ定義への変換，および，問合せや永続データの操作を容易に記述できるようにするためのメソッド定義の出力を自動的に行なう ASN.1 コンパイラの実現方法について論じる．この ASN.1 コンパイラを用いることにより，アプリケーション開発者はデータの蓄積・管理部をほぼ自動的に生成できるようになり，これを用いてデータの問合せなどのデータベースアクセス処理を効率良く記述できるようになる．

以下，3.5.1節で，汎用的な ASN.1 データベースを実現する際の基本方針について述べる．3.5.2節では，OODBMS を利用した ASN.1 データベースを実現するための ASN.1 コンパイラの実装について述べる．3.5.3節では，実現した ASN.1 コンパイラを用いたディレクトリ情報ベース (Directory Information Base, DIB) の実現およびその性能評価について述べ，実現した ASN.1 コンパイラの有用性を実証する．

### 3.5.1 ASN.1 データベースの実現のための基本方針

これまで，ASN.1 データを格納するためのデータベースに関するいくつかの実装例が報告されている．例えば，Hart ら [HSO94] は，ASN.1 でデータ構造が定義されたゲノムデータベースを関係データベースシステムを用いて実現する方法について示している．しかし，関係データモデルは平坦な表形式のデータ構造しか許さないため，複雑な入れ子構造をもつ元の ASN.1 データの構造は，一般に，複数の関係に分割して表現する必要があり，問合せの記述の容易さや問合せ処理速度の点で問題がある．また，OSI ディレクトリ [ISO88] における DIB の実現 [NK+91, NO+93] や OSI 管理情報ベース (Management Information Base, MIB) の実現 [NH+93, YF92] など，アプリケーションに特化したデータベースの実現についても報告されている．しかし，これらの実装はアプリケーションに依存した形で行なわれており，他のアプリケーションの開発への応用が困難である．さらに，データの蓄積および管理を独自に設計・実装しており，開発効率の点で問題がある．

今後のASN.1の利用分野の拡大を考慮した場合，ASN.1アプリケーションに対する汎用的なASN.1データベースの実現のための枠組を提供することがシステム開発効率の観点から重要である．複雑な入れ子構造を含むASN.1の型記述能力を考慮すると，そのプラットフォームとしてはOODBMSが適していると考えられる．しかし，ASN.1は，CHOICE型やOPTIONAL指定，DEFAULT指定など，オブジェクト指向データモデルでも直接は表現できない型表現能力をもつ．したがって，ASN.1データベースをOODBMSを用いて実現するためには，ASN.1で定義されたデータ型を，これらのASN.1の特徴を考慮しながらOODBMSにおけるスキーマ定義に変換する必要がある．

また，データベース機能を使用する通信アプリケーションの開発を支援するには，アプリケーションプログラムを作成する際にデータベースシステムがもつ機能を容易に利用できることが重要である．この点を重視し，ASN.1アプリケーションの開発支援に広く用いられているASN.1コンパイラ [NM+90, Samp93] の考え方をベースに，OODBMSを用いたASN.1データベースの実現の核となるASN.1コンパイラの実装に際して，以下の基本方針を採用した．

通信アプリケーションは通常C言語やC++言語によって記述されることを考慮し，専用のデータベース言語をもつOODBMSではなく，C++言語のクラス定義がほぼそのままの形でスキーマ定義となるような，C++言語をベースとするOODBMSを対象とする．したがって，実現するASN.1コンパイラは，ASN.1による型定義の集合（ASN.1モジュール）を入力とし，OODBMSを用いたデータ蓄積を考慮したC++言語のクラス定義を生成する．また，生成されたクラス定義は，OODBMSに対するスキーマ定義となるとともに，アプリケーションプログラム作成のベースともなるため，単にデータ蓄積を考慮するだけでなく，生成されたクラスを用いたアプリケーションプログラムの記述を容易にするために有用なメソッド定義を同時に出力することとする．

この基本方針のもとで，データの永続性を考慮すると，次の点が問題となる．

- ASN.1モジュールの中では複数のデータ型が定義されるが，永続性を必要とするデータ型（実データを表す型）と永続性を必要としないデータ型（プロト

コルに関連するパラメータを表す型など）がある．永続性を必要とするデータ型については ASN.1 コンパイラによってそれを考慮したクラス定義を出力する必要があるが，永続性を必要とするデータ型かどうかは，通常の ASN.1 モジュールの記述からは判断できない．

- インデックスの生成は検索処理の高速化に重要である．C++言語に基づく OODBMS では，インデックス機構を利用するためにあらかじめクラス定義の中にインデックスの生成を指示するコードを記述しておく必要がある．したがって，ASN.1 コンパイラがインデックスの生成を指示するコードを出力することが望ましいが，通常の ASN.1 モジュールの記述からはどのデータ型にインデックスを付加すればよいかは判断できない．

これらの問題に対し，次のような方法をとる．

1. 永続性を必要とするデータ型に対して，アプリケーション開発者が明示的に "--DB extent--" というコンパイラ指令を記述するものとする．例えば，図 3.2 の ASN.1 モジュール例における Person 型を根とするデータ型のデータベースへの蓄積および操作を可能にするには，次のようなコンパイラ指令を追加すればよいものとする．

```
Person ::= --DB extent--
        SEQUENCE { ... }
```

2. 入力する ASN.1 モジュール内にアプリケーション開発者が明示的に "--DB index on *index_path*" というコンパイラ指令を記述することによってインデクシング指定ができるようする．例えば，図 3.2の ASN.1 モジュールにおいて次のようなコンパイラ指令を記述することにより，Person 型のフィールド name をキーとしたインデックスが生成されるようにする．

```
--DB index on Person.name
```

この ASN.1 コンパイラを使用した場合の通信アプリケーションの開発の流れを図 3.7に示す．

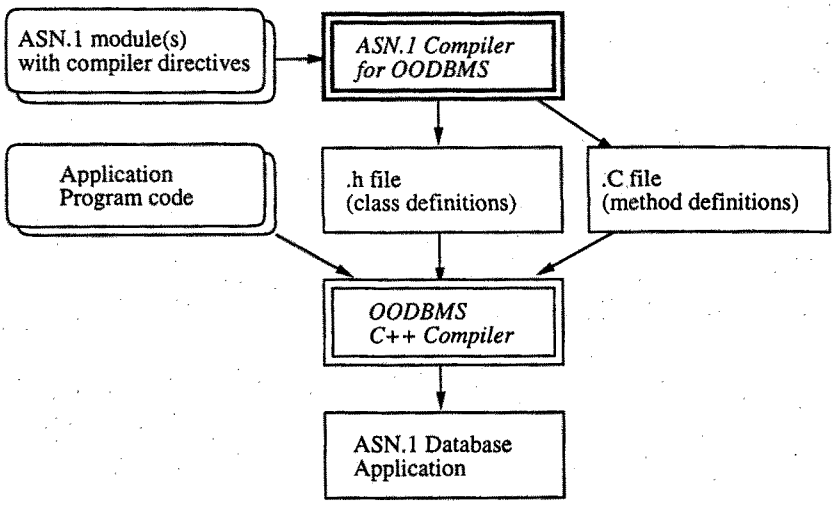



図 3.7: ASN.1 コンパイラを用いた ASN.1 データベースアプリケーション開発の流れ

### 3.5.2 OODBMS を対象とした ASN.1 コンパイラの実現方法

ASN.1 データ構造定義を C++クラス定義に変換する従来の ASN.1 コンパイラでは，ASN.1 の基本型に対応する C++クラスライブラリを提供し，入力される ASN.1 モジュール内で定義される各データ型を，基本的に C++の一つのクラス定義に変換して出力する．このように出力されるクラス定義においては，データの蓄積に関しては当然考慮されていない．しかし，この変換方法をはじめとする従来の ASN.1 コンパイラの基本的な機能は OODBMS へのスキーマ定義の出力のために応用が可能である．

#### 3.5.2.1 データ構造の変換

ASN.1 コンパイラによって出力される C++クラス定義が，ASN.1 の特徴を考慮した OODBMS のスキーマ定義となるように，構造型に関しては次のような変換を行なう．

- SEQUENCE 型や SET 型から変換されたクラスにおいて，OPTIONAL や DEFAULT が指定されていないフィールドに対応するデータメンバは埋め込み型として宣言し，OPTIONAL あるいは DEFAULT が指定されたフィールドに

対応するデータメンバは，そのデータメンバが値をもたないことや規定値をもつことを NULL ポインタによって示せるようにポインタ型として宣言する．

- SEQUENCE OF 型や SET OF 型は，OODBMS が提供するリストクラスや集合クラスのテンプレートを使用するように変換する．これにより，これらのリスト値や集合値の要素に対する問合せが可能となる．

- CHOICE 型は，その選択フィールドをメンバとする共用体と，どのデータメンバが選択されているかを示す選択識別子の二つのデータメンバに変換する．共用体内の各メンバは，格納領域の無駄を少なくするためにすべてポインタ型として宣言する．

#### 3.5.2.2 メソッド定義の出力

問合せや永続データの操作を考慮し，通常の ASN.1 コンパイラの機能である符号化メソッド，復号化メソッドの出力に加え，データの永続性を考慮した以下のメソッドを各クラスの定義において出力する．

- ==演算子: インスタンスの値の等価性を判定する演算子．データベースに対する問合せにおいては，データ値の等価性の評価が重要である．通常，クラスのインスタンスに対する==演算子は，そのクラスのメンバがポインタ型である場合そのポインタ値の比較を行なうため，OPTIONAL フィールドに対応するポインタ型のデータメンバが指すインスタンスの値の等価性を判定することができない．この==演算子の多重定義により，OPTIONAL フィールドに対応するデータメンバに対しても値の等価性を調べることができるようになる．また，DEFAULT 指定されたフィールドについては，それに対応するデータメンバのポインタ値が NULL である場合には規定値との比較を行なう必要があるため，次に述べるメソッドをあらかじめ適用しておくことによって規定値を設定し，値の等価性を判定できるようにした．これを用いれば，構造をもつ任意のデータ値の等価性を検査することができる．

- Set_*field_name*_To_Default メソッド: *field_name* で示されるフィールドに対応するデータメンバの値を規定値に設定するメソッド.

- PersistentCopy メソッド: 一時的インスタンスの値を永続的インスタンスにコピーし, コピーされた永続的インスタンスへのポインタを返すメソッド. 例えば, ネットワークを介して受信したデータをデータベースに格納する場合などに用いる.

- TransientCopy メソッド: 永続的インスタンスの値を一時的インスタンスにコピーし, コピーされた一時的インスタンスへのポインタを返すメソッド. 例えば, データベースから取り出した値を変更して送信する場合などに用いる.

#### 3.5.2.3 コンパイラ指令に基づくコードの出力

1. コンパイラ指令 "--DB extent--" が記述されたデータ型に対して, ASN.1 コンパイラはその外延集合をエントリポイントとして定義するコードをクラス定義内に出力すると同時に, その外延集合を自動的に管理するため, その ASN.1 型から変換されたクラスのコンストラクタとデストラクタにおいて, 永続データの生成や削除にともなう外延集合の自動的な更新を行なうコードを出力する.

2. コンパイラ指令 "--DB index on *index_path*" に対して, ASN.1 コンパイラは記述されたインデックスパスを解析し, そのパスに含まれる各データメンバに対する索引付け宣言をクラス定義中に出力する. また, 実際にインデックスを生成するための関数を出力する. これにより, アプリケーション開発者がこの関数を main 関数から呼び出すようなコードを記述するだけでインデックスの生成が行なえるようにする.

#### 3.5.2.4 ObjectStore を対象とした実装

前節で述べた方法に基づき, C++言語をベースとする OODBMS の一つである ObjectStore を対象とした ASN.1 コンパイラの実装を行なった. また, 実装にあたっては, ASN.1 データ構造定義を C++クラス定義に変換するフリーソフトウェアの

```
University ::= SEQUENCE {
  name    PrintableString,
  faculty PrintableString OPTIONAL }
```

↓

```
class University {
public:
  PrintableString  name;
  PrintableString* faculty;
  University() { faculty = NULL; }
  ...
};
```

(a) SEQUENCE 型

```
Children ::= SET OF PrintableString
```

↓

```
class Children {
public:
  os_Set<PrintableString*> set;
  ...
};
```

(b) SET OF 型

```
Belong ::= CHOICE {
  univ [0] University,
  comp [1] Company,
  null [2] NULL }
```

↓

```
class Belong {
public:
  enum {
    univCid = 1,
    compCid = 2,
    nullCid = 3 } choiceId;
  union {
    University* univ;
    Company*    comp;
    AsnNull*    null;
  };
  int discriminant() { return choiceId; }
  ...
};
```

(c) CHOICE 型

図 3.8: ASN.1 構造型から C++クラス定義への変換例

```
Person ::= --DB extent--
  SEQUENCE {
    ... }

            ↓

class Person {
public:
  persistent<db>
      os_Set<Person*>* extent;
  ...
  Person() {
    if (objectstore::is_persistent(this))
      extent->insert(this);
  }
  ~Person() {
    if (objectstore::is_persistent(this))
      extent->remove(this);
  }
  ...
};
```

図 3.9: 外延集合定義指定を含む型の変換例

ASN.1 コンパイラである snacc[Samp93] を機能拡張する形で実現した．今回実装した ASN.1 コンパイラのデータ構造変換例を図 3.8に，外延集合の生成を指示されたデータ型の変換例を図 3.9に示す．実装においては ObjectStore を対象としたが，C++言語をベースとする他の OODBMS を対象とする ASN.1 コンパイラも，少ない変更で実現できるものと考える．

ObjectStore の提供するインデックス機構においては，インデックスを生成するパスの末端の要素の型が整数型やポインタ型などの基本型ではなくユーザ定義クラスである場合，そのクラスの要素の順序付けを行なうための比較関数（ランク関数）およびハッシュ関数を定義しなければならない．ASN.1 コンパイラにおいてこのような関数を自動的に生成することは困難であるので，関数の枠組となるコードのみ

を出力し，これらの関数の定義はアプリケーション開発者が記述することとした．

したがって，アプリケーション開発者の作業手順をまとめると，以下のようになる．

1. アプリケーションの仕様で与えられた ASN.1 モジュールの中に，蓄積すべきデータ型に関するコンパイラ指令およびインデクシングを行なうパスに関するコンパイラ指令の記述を追加し，ASN.1 コンパイラに入力する．これにより，OODBMS におけるスキーマ定義となる C++言語のクラス定義を生成する．

2. 指定したそれぞれのインデックスに対して，比較関数およびハッシュ関数の定義を行なう．また，main 関数の中から，ASN.1 コンパイラが自動的に生成したインデックス生成のための関数を呼び出すようにする．

3. ASN.1 コンパイラが生成したクラス定義に基づき，アプリケーションプログラミングを行なう．

4. OODBMS が提供する C++コンパイラによりコンパイルすることにより，実行形式のアプリケーションを作成する．

このように，データの蓄積およびインデクシングに関してアプリケーション開発者が行なわなければならない作業は，コンパイラ指令の記述といくつかの簡単な関数の定義のみである．

### 3.5.3 DIB の構築によるシステム評価

OODBMS を対象とした ASN.1 コンパイラを評価するために，ASN.1 を利用した通信アプリケーションの一つである OSI ディレクトリ [ISO88] における DIB の構築を行ない，その性能評価を行なった．本節では，DIB の構築に際して必要となる DIB のデータ構造の定義と，それを用いて実現した DIB の性能評価について述べる．

#### 3.5.3.1 DIB の概要

OSI ディレクトリは ISO で規定された国際標準であり，さまざまな情報を利用者に提供するための基本的なアプリケーションである．提供される情報は DIB とよ

表 3.1: DIB に対する検索操作群

| 操作名 | 処理内容 |
|---|---|
| Read | 与えられた識別名に対し，そのエントリ情報を返す. |
| Compare | 与えられた識別名および一つの属性値アサーションに対し，DIB の内容と一致するかどうかを返す. |
| List | 与えられた識別名に対し，そのエントリの直接下位エントリの相対識別名の集合を返す. |
| Search | 与えられた識別名に対し，それに対応するエントリ，そのエントリの直接下位エントリ，そのエントリのすべての下位エントリのいずれかに関して，与えられた選択条件を満たすエントリ情報の集合を返す. |

ばれるデータベースに蓄積管理される．DIB に蓄積される情報の基本単位はエントリとよばれ，各エントリはそれを識別するための識別名および属性値の集合によって定義される．エントリに付与される識別名は相対識別名のリストとして定義され，相対識別名は属性型と属性値の組の集合によって定義される．この識別名によって，ファイルシステムにおけるディレクトリ構造と同様に，各エントリは階層構造の中に配置される．この階層構造は DIT (Directory Information Tree) とよばれる．OSI ディレクトリにおいて規定された DIB に対する検索操作群を，表 3.1に示す.

OSI ディレクトリの国際標準で規定されたエントリ情報を表すための ASN.1 によるデータ構造定義を図 3.10に示す．このように，エントリ情報は，従来のデータベースアプリケーションと比べてネストの深い複雑な構造をもつ．例えば識別名の型を表す DistinguishedName 型は，SEQUENCE OF SET OF SEQUENCE {AttributeType, AttributeValue} という型をもつ.

#### 3.5.3.2 DIB 構造の定義

表 3.1の操作群からもわかるように，DIB に対する検索では，まず，検索の対象となるエントリを識別名をキーとして識別する必要がある．これを実現するためには，図 3.10に示した EntryInformation 型をそのまま DIB のエントリ構造とすればよい．しかし，この方法では DIT がもつ階層構造が直接表現されないために，List

```
EntryInformation ::= SEQUENCE {
            DistinguishedName,
  fromEntry BOOLEAN DEFAULT TRUE,
            SET OF CHOICE {
              AttributeType,
              Attribute } OPTIONAL }
Attribute ::= SEQUENCE {
  type  AttributeType,
  value SET OF AttributeValue }
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY
AttributeValueAssertion ::= SEQUENCE {
  AttributeType,
  AttributeValue }
RDNSequence ::=
  SEQUENCE OF RelativeDistinguishedName
DistinguishedName ::= RDNSequence
RelativeDistinguishedName ::=
  SET OF AttributeValueAssertion
```

図 3.10: OSI ディレクトリの EntryInformation 型の定義

```
Entry ::= --DB extent--
  SEQUENCE {
    entryInfo EntryInformation,
    children  SET OF Entry }
--DB index on Entry.entryInfo.distinguishedName
```

図 3.11: DIB エントリの定義

操作や Search 操作など，識別したエントリの下位エントリをさらに取り出す必要がある検索操作に時間がかかる．そこで，DIT 構造を DIB に反映させるために，EntryInformation 型の要素とその下位エントリの情報をもつ Entry 型を，DIB 構造の根となるデータ型として新たに定義した．また，DIB に蓄積されているエントリ情報の集合に対する検索を可能とするために，Entry 型に対して外延集合の定義を指示するコンパイラ指令を記述し，さらに，その検索が識別名によって行なわれることを考慮し，識別名をキーとするインデックスの生成を指示するコンパイラ指令を加えた．国際標準によって与えられた ASN.1 モジュールに対して追加した記述を図 3.11に示す．

#### 3.5.3.3 検索プログラムの記述

前節のように定義した DIB 構造を，実現した ASN.1 コンパイラに入力して C++ のクラス定義を生成し，その上で検索プログラムの記述を行なった．

DIB に対する Read 操作や List 操作などの検索を行なう関数は，OODBMS が提供するデータ操作言語を用いて記述した．Read 操作を処理する関数の一部を図 3.12 に示す．この図に示した部分は，引数として与えられた識別名をもつエントリを DIB 内で特定するための部分であり，List 操作など他の操作の処理を行なう関数でも同様の問合せ文で対象エントリを特定できる．このように，比較的簡単な記述でエントリの特定を行なうことができる．また，識別名をキーとするインデックスの生成を指示しているため，この検索文は実行時に最適化される．このようにして特定したエントリの属性値集合から必要なデータを取り出して，その処理結果を構成する．

```
ReadResult* read(DistinguishedName& dn)
{
  ...
  Entry* entry = (*Entry::extent)
     [% entryInfo.distinguishedName == dn %];
  ...
}
```

図 3.12: Read 操作を実現する関数の一部

但し，今回の評価は別名エントリを含まない場合のみを扱っているため，この検索文によってエントリを特定できない場合にはエラーとすればよいが，別名エントリを含む場合には，名前解析処理において別名の展開を行なう必要がある．このような名前解析処理には，例えば，西山らの提案する方法 [NY+94] を利用することが考えられる．

#### 3.5.3.4 実現した DIB の評価

実現した DIB に対する Read 操作（全属性値の読み出しおよび属性型のみの読み出し）と List 操作に関する性能評価を行なった．その性能評価環境と性能評価結果を表 3.2に示す．この性能評価結果は，操作対象エントリの識別名をランダムに生成しそのエントリに対する操作要求を ASN.1 の基本符号化規則で符号化された符号列の形式で与え，その処理結果を符号列の形式で得るまでにかかる時間を 1,000 回繰り返し計測した平均値である．したがって，計測した時間には，符号列で与えられた操作要求の復号化にかかる時間，問合せ文によって対象となるエントリをデータベース内で特定するのにかかる時間，操作要求の種類やパラメータにしたがってそのエントリから必要な情報を取り出すのにかかる時間，その結果を符号化するまでの時間が含まれる．また，トランザクションの開始および終了にかかる時間も含まれる．

この結果から，どの操作に対しても 50ms 程度で処理できることがわかる．DIB の実装に関する文献 [NK+91, NO+93] で報告されている評価値と比較しても，評価

表 3.2: 構築した DIB の性能評価

| 性能評価環境 | |
|---|---|
| H/W | Sun Microsystems 社製<br>SPARCstation 10 |
| DIT 構造 | 国，組織，組織単位，組織人の 4 レベルの階層構造 (10 分木)<br>別名エントリは含まない |
| 格納エントリ数 | ルート: 1 エントリ<br>国 : 10 エントリ<br>組織: 100 エントリ<br>組織単位: 1,000 エントリ<br>組織人: 10,000 エントリ<br>合計: 11,111 エントリ |
| 格納属性 | 国 : 国名<br>組織: 組織名<br>組織単位: 組織単位名<br>組織人: 一般名，姓，電話番号，<br>郵便番号，郵便アドレス |
| **性能評価結果** | |
| Read 操作<br>全属性値/型のみ | 国 : 平均 29.6ms / 41.0ms<br>組織: 平均 33.8ms / 43.8ms<br>組織単位: 平均 44.1ms / 44.3ms<br>組織人: 平均 52.1ms / 50.0ms<br>総平均: 39.9ms / 44.8ms |
| List 操作 | 国 : 平均 46.2ms<br>組織: 平均 47.5ms<br>組織単位: 平均 53.5ms<br>組織人: 平均 51.5ms<br>総平均: 49.7ms |

環境は異なるものの遜色のない数値である．また，アソシエーションの確立などのOSI 通信処理には通常この数倍から数十倍の時間がかかるため [NK+91, NO+93]，実現したDIB は十分実用的であるといえる．また，Compare 操作と Search 操作に関してはここでは評価を行なわなかったが，これらの操作においてもエントリを特定してそこから必要なデータを抽出して結果を得るという手順は Read 操作や List 操作と同じであり，同様の性能評価値が得られるものと考えられる．

また，DIB の実現にあたり，文献 [NK+91, NO+93] では専用のデータベースシステムの設計から行なっているのに対し，本実装では ASN.1 コンパイラを用いることにより容易に実用レベルの DIB を構築できた点も重要である．

## 3.6 ASN.1 データの格納方式

3.5節で述べた，OODBMS を用いた ASN.1 データベースの実現方法では，既存の商用 DBMS を利用するため，一般に次のような問題がある．

1. 検索時間に関する問題

   データ検索の高速化に必要なインデックス機能やクラスタリング機能が，使用する DBMS で提供されているものをそのまま利用することになるため，ASN.1 データの構造を考慮した適切な高速化手法をとることが不可能である．さらに，ASN.1 データベースシステムとしては不要な機能によるオーバヘッドが存在する．

2. 符号化・復号化の問題

   ASN.1 データは符号列として送受信されるが，データの格納を復号化したあとの内部データ表現で行なっているため，データの送信の際にコストのかかる符号化処理を行なう必要がある．また，データの格納においては，直接の検索キーとならないデータに対しても復号化して格納する必要がある．

これらの問題は，通信アプリケーションに要求されるパフォーマンスに重大な影響を与えるものであるが，商用 DBMS を利用する限り，これらの問題をすべて解決

することはできない．そこで，本節ではより一般的な立場から，ASN.1データを格納する方式について議論する．

一般に，ASN.1データの格納方式として，以下の二つの方式が考えられる．

1. 計算機での内部表現形式での格納方式（図3.13）

   データの送信時および受信時に必ず符号化・復号化処理が必要になる．

2. ASN.1の符号化規則に基づく表現形式での格納方式（図3.14）

   受信した符号列をそのままデータベースに格納することにより，復号化処理を省略することができる．また，データの送信時にも符号化処理を省略できる．しかし，データを検索した後に何らかの処理を行なってから送信する場合，および，データを受信した後に何らかの処理を行なってからデータベースに格納する場合には，符号化・復号化処理を行なう必要がある．

上記の二つの格納方式のうち，どちらの方式が効率よくデータを提供できるかという問題は，データ検索や通信の頻度など，ASN.1データベースを利用するアプリケーションの特性に依存すると考えられる．例えば，検索キーとなるデータ項目が固定されており，検索されたデータをそのまま送信すればよい場合には，検索キー以外のデータ項目の符号化・復号化処理は無駄な処理となるため，ASN.1の符号列のままデータを格納した方が効率的である．一方，検索されたデータに対してさまざまな処理を行なった後にデータを送信するようなアプリケーションの場合には，計算機の内部表現形式でデータを格納しておいた方が効率的である．

ここでは，OSIディレクトリにおけるDIBやネットワーク管理のためのMIBに代表される通信アプリケーションを主に対象として考える．このようなアプリケーションでは，検索キーが固定されていることから，ASN.1の符号列で格納した方が効率的である．

以下，ASN.1データを符号化規則に基づく符号列で格納する方式を前提とし，この場合にどのようなクラスタリングおよびインデクシングが適しているかを議論する．

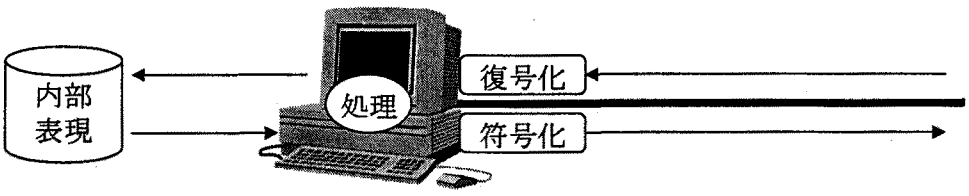



図 3.13: 計算機の内部表現形式で格納した場合の符号化・復号化処理

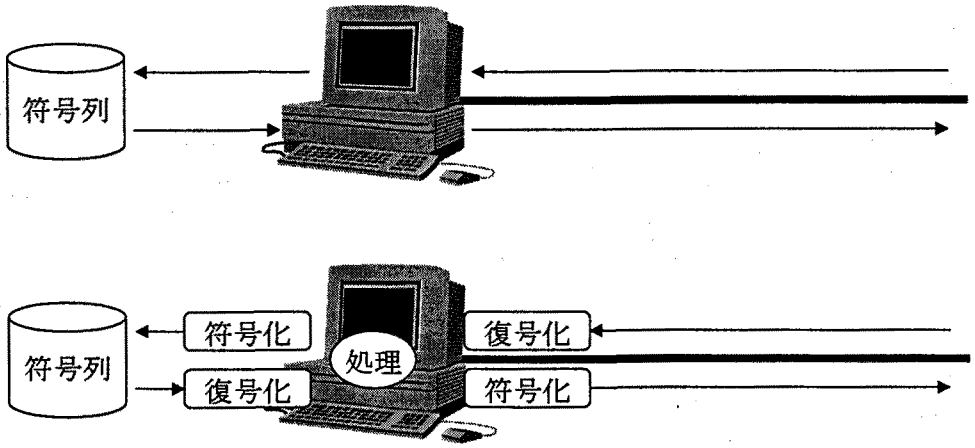



図 3.14: ASN.1 の符号列で格納した場合の符号化・復号化処理

### 3.6.1 ASN.1データのクラスタリング

一般に，ASN.1で定義されるデータ構造をもつデータは，内部に入れ子構造を含んだ複雑な構造をもつ．このような複雑な構造をもつデータ全体を一つの格納単位として二次記憶に格納すると，そのデータの一部しか必要ない場合でも全体を主記憶に取り出すことになり，さらにその中から必要なデータを取り出すために復号化処理を行なわなければならない．したがって，検索効率を向上させるためには，論理的に一つのデータを分解して二次記憶に格納し，部分的な復号化処理を可能にする方法が有効である．このようにデータ構造を論理的に分割して格納する手法を，文献[Kato91]では論理クラスタリングと呼んでいる．

ここでは，複雑なデータ構造をもつASN.1データに対してどのような論理クラスタリングを行なえば検索効率を向上させることができるかを，ASN.1データの符号化の問題と合わせて考察する．

ASN.1データに対する符号化規則では，大きく分けて単純型に対する符号化規則と構造型に対する符号化規則がある（図3.3参照）．

#### 3.6.1.1 単純型符号化によるクラスタリング

クラスタリングの第一の手法は，構造をもたない単純型をベースとしたクラスタリングである．この手法の概念図を図3.15に示す．この手法では，まず格納すべき型に対してそれぞれページを割り当てる．そして，単純型のデータのみを符号列として格納し，構造型のデータに関しては，内部要素として実データを格納するかわりに，分割格納された各要素に対して付与された識別子(ID)を格納する．

この手法には以下のような特徴がある．

- 符号化処理においてもっともコストのかかる単純型の符号化を行なう必要がない．構造型の符号列を得るためには，符号化された状態の単純型のデータを集め，長さを計算しタグを付加するだけでよい．

- 値を更新する場合，単純型のデータを更新するだけでよい．

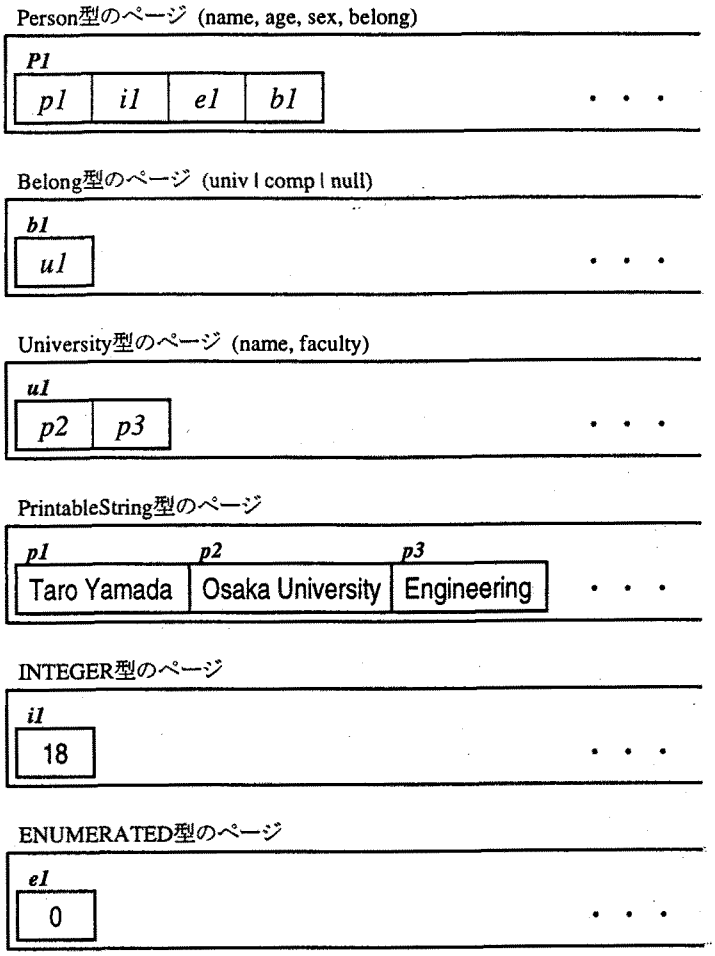



図 3.15: 単純型符号化によるクラスタリング

- 構造型全体を取り出す場合，データが多くのページに分散して格納されるため多くのページへのアクセスが必要となる．

#### 3.6.1.2 構造型符号化によるクラスタリング

クラスタリングの第二の手法は，構造型の符号列をベースとしたクラスタリングである．この手法の概念図を図 3.16に示す．この手法では，構造型のデータを，各構造型に対して割り当てられたページに符号列としてほぼそのままの形で格納する．ただし，その中に構造型である要素がある場合は，その要素を別のページに格納し，実際のデータのかわりにその要素の ID を格納する．また，構造型全体の符号列を

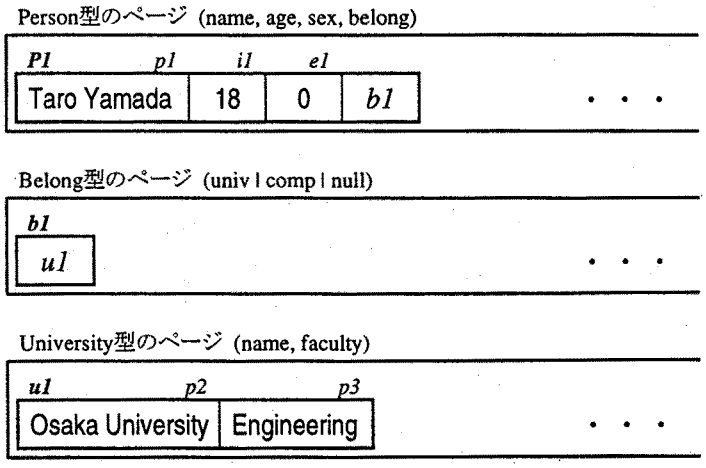



図 3.16: 構造型符号化によるクラスタリング

得る際にその先頭に付加すべきタグと長さは，データベースには格納せず，データベースから読み出した後に計算し付加する．

この手法には以下のような特徴がある．

- 単純型符号化によるクラスタリングよりもデータの分割が少ないため，構造型全体を取り出す際のアクセスページ数が少なくなる．
- 構造型に含まれる要素の値を更新する場合，更新前と更新後で符号長が変化する可能性があるため，再符号化を行なう必要がある．
- 構造型の中のある要素を取り出す必要がある場合に，符号化された状態の構造型のデータを復号化する必要がある．

#### 3.6.1.3 ユーザ指定によるクラスタリング

単純型符号化によるクラスタリングおよび構造型符号化によるクラスタリングは，与えられたデータ構造定義に基づいてシステムが自動的にクラスタリングを行なうものである．しかし，このようなクラスタリングは，通信アプリケーションのデータアクセス特性に必ずしも適合しているとは限らない．

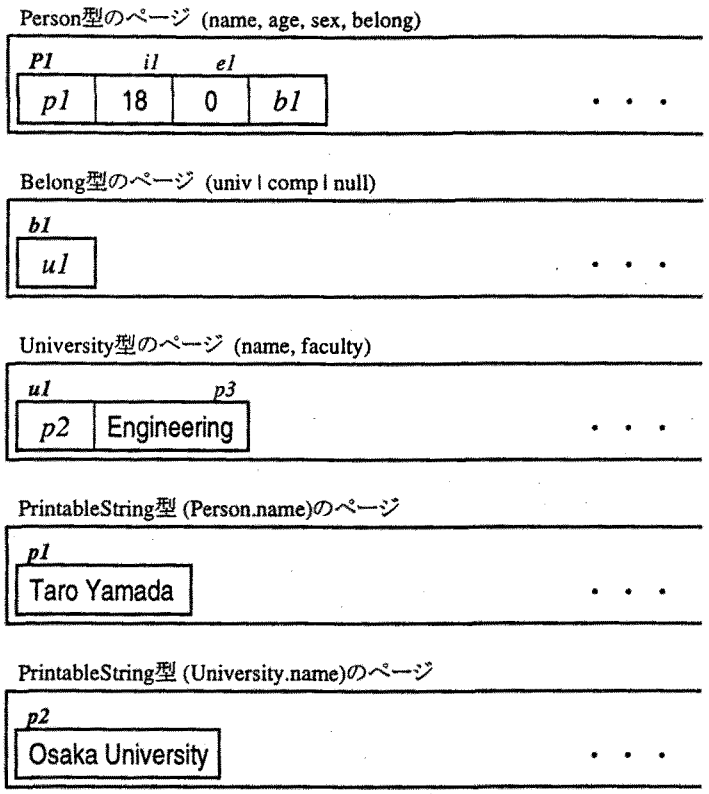



図 3.17: ユーザ指定によるクラスタリング

クラスタリングの第三の手法は，ユーザ指定に基づくクラスタリングである．この手法の概念図を図 3.17に示す．この手法は，基本的には構造型符号化によるクラスタリングを行なうが，構造型の中で検索や更新が頻繁に行なわれるフィールドをユーザが指定することにより，指定されたフィールドのデータをそれぞれ別ページにクラスタリングする．図 3.17の例は，ユーザが Person 型の構造型の中の name フィールド，および，University 型の構造体の中の name フィールドを検索対象として指定した例を表している．

この手法は，アクセスページ数が減少するという構造型符号化によるクラスタリングの長所と，構造型に含まれる要素に対する検索・更新操作の処理が容易になるという単純型符号化によるクラスタリングの長所を合わせもつ．通信アプリケーションではデータベースへのアクセスパターンが前もってわかることが多いことから，この手法が ASN.1 データのクラスタリング手法として適していると考えられる．

### 3.6.2 ASN.1データに対するインデクシング

ASN.1データベースシステムは通信アプリケーションを対象としていることから，クライアントからのデータの検索・転送要求に対して高速に応答しなければならない．したがって，ASN.1データベースシステムにおけるインデックス機構の設計は，そのパフォーマンスに大きな影響を与える．さらに，ASN.1データが符号列の形式でデータベースに格納されることも考慮しなければならない．

以下，ASN.1データベースシステムにおけるインデクシング手法に関して，考えられる手法を比較・検討する．

#### 3.6.2.1 パスインデックス

パスインデックスは，入れ子構造をもつデータに対してインデックスのキーとなるデータ項目までのパス情報を管理する手法である．そのエントリは，キー値とパス内に存在する要素のIDから構成される．例えば，図3.2のASN.1モジュールで定義されるPerson型において，

Person.belong.univ.name

Person.belong.univ.faculty

の二つのパスに対してインデックスを付加する場合，この手法では図3.18のようなパスインデックスが作成される．(図中，斜字体で示した要素は，図3.15，図3.16，図3.17での各要素のIDを表す.)

この手法では，検索を行なう際にインデックスへのアクセスが一度で済むため検索効率がよい．また，パス内に存在するすべての要素のIDが得られるため，そのIDを用いてパスの途中からデータを取り出すこともできる．しかし，図3.18から明らかなように，共通部分をもつ複数のパスに対してインデックスを生成した場合，共通部分の要素のIDを重複して記録しなければならないため，インデックスの格納に必要な物理記憶領域が増大する．また，複数のデータから同一のデータ要素を参照しているという多重参照（図3.21参照）が存在するようなデータベースでは，そのデータ要素を参照しているすべてのデータに関して，そのデータ要素以下のIDをや

はり重複して記録しなければならないため，物理記憶領域の増大だけでなく，データの更新操作の処理が複雑になる．

#### 3.6.2.2 マルチインデックス

マルチインデックスは，パスを組に分割し，その組ごとのインデックスを作成してパスインデックスと同等の情報を保持する手法である．例えば，図3.18と同等の情報をマルチインデックスで管理する場合，図3.19に示したように5つのインデックスに分割して格納される．

この手法では，パスインデックスとは対照的に，共通部分をもつ複数のパスに対するインデックスを生成した場合でもIDを重複して記録する必要はなくなるため，データの更新・削除操作に対するインデックスの更新操作が容易になる．同様に，データ要素の多重参照がある場合でもIDの重複が防げる，しかし，特にパスの共通部分が短い場合には，パスを分割するための物理記憶領域のオーバヘッドが増大する．また，検索を行なう際には複数のインデックスへのアクセスが伴うことになり，特にパス長が長い場合には検索速度が低下する．

#### 3.6.2.3 BP インデックス

ASN.1を利用する通信アプリケーションでは，一般に扱わなければならないデータ構造が非常に複雑であり，深い入れ子構造をもつものが多い．

そのため，ASN.1データベースシステムにおいてパスインデックスを使用すると，複数のパスにインデックスを付加した場合に，インデックスに格納するIDの重複がかなり大きくなることが予想され，その問題点が大きく影響することが考えられる．一方，マルチインデックスを使用すると，検索を行なう際にアクセスすべきインデックスの数が増大し，検索速度の低下の問題が顕著となる．

そこで，パスインデックスとマルチインデックスの中間的なインデクシング手法として，BP (Branch Point) インデックスを提案する．BP インデックスでは，複数の共通部分をもつパスに対してインデックスを生成する場合，パスを組単位で分割するのではなく，その共通部分でまとめて分割する．例えば，図3.18，図3.19と同

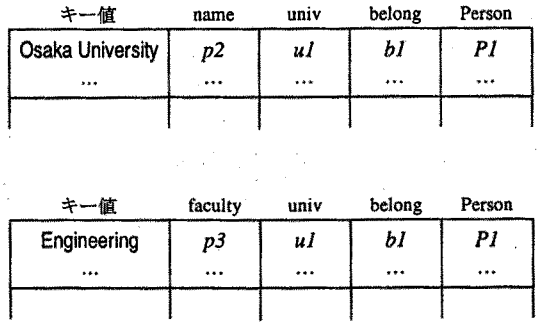



図 3.18: パスインデックス

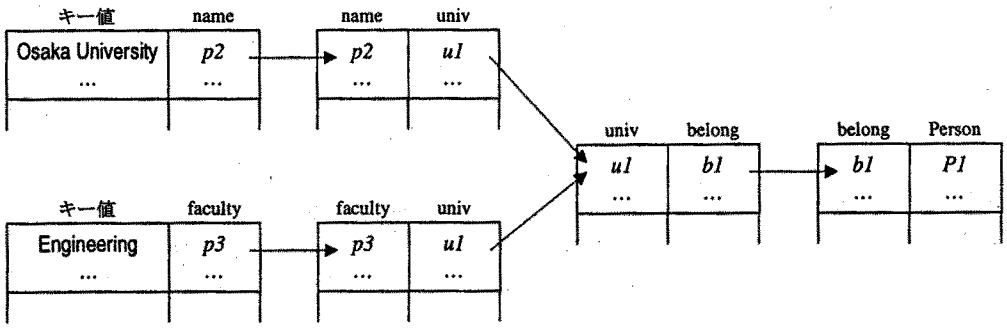



図 3.19: マルチインデックス

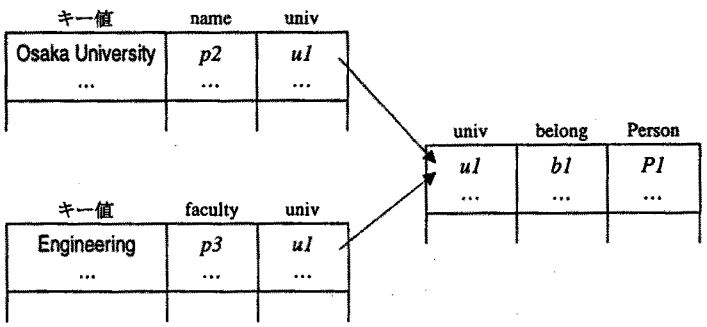



図 3.20: BP インデックス

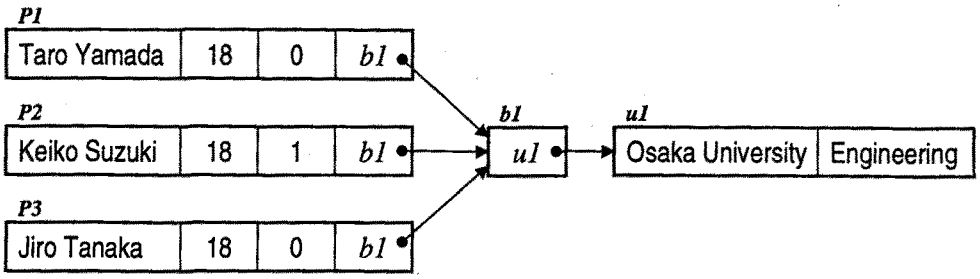



図 3.21: データ要素の多重参照

等のインデックスは，BP インデックスでは図 3.20のように格納される．

この手法では，パスインデックスの問題点である ID 情報の重複が解決され，マルチインデックスの問題点であるアクセスすべきインデックス数の増大も抑制されると考えられる．

#### 3.6.2.4 比較評価

三つのインデクシング手法を比較評価するため，それぞれのインデクシング手法を実装し，インデックスの格納に必要な物理記憶領域，および，データの検索，追加，更新の各操作の処理にかかる時間を測定した．測定にあたっては，二つのパスに対してインデックスを付加し，そのパス長および共通部分の長さを変化させた．データ操作にかかる時間は，データベースに格納されているレコード数が 100 件から 20,000 件の間で変化させて測定した．各インデクシング手法は，一般にインデックスの実現方法として用いられている B 木に基づいて実装した．

レコード数が 10,000 件の場合における各インデクシング手法における占有物理記憶領域を表 3.3に示す．ここで，キーとなるデータ項目が占める物理記憶領域は，どの手法も 194,780 バイトであった．この表からわかるように，占有する物理記憶領域は BP インデックスが最小である．この評価結果は共通部分をもつ二つのパスに対してインデックスを付加した場合の結果であるが，より多くのパスに対してインデックスを付加する場合には BP インデックスと他の二手法でさらに差が大きくなると考えられる．

また，データ検索処理時間の測定結果を図 3.22に，データ追加処理時間の測定結

表 3.3: 占有物理記憶領域の比較

| インデックス手法 | サイズ (byte) | | |
|---|---|---|---|
| | パス長: 7<br>共通部分長: 6 | パス長: 7<br>共通部分長: 1 | パス長: 3<br>共通部分長: 2 |
| パスインデックス | 834,780 | 834,780 | 514,780 |
| マルチインデックス | 834,780 | 1234,780 | 514,780 |
| BP インデックス | 634,780 | 834,780 | 474,780 |

果を図 3.23に，データ更新処理時間の測定結果を図 3.24にそれぞれ示す．ただし，データ更新処理時間はパスの二番目の要素以下に対する更新操作を行なった場合の平均所要時間である．

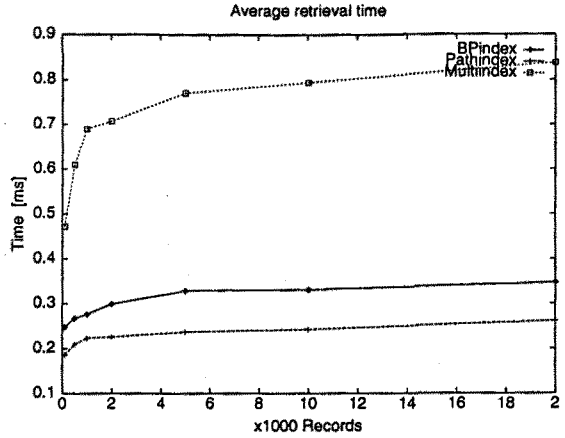
これらの図から，マルチインデックスはデータ操作に対して複数のインデックスへのアクセスが必要となるために生じる処理時間の増大が顕著であることがわかる．また，BP インデックスはパスインデックスよりも一般に処理時間がかかるものの，その差はあまり大きくなく，追加操作や更新操作においてはより高速に処理できる場合もあることがわかる．

一方，データ要素の多重参照が存在する場合を考えると，マルチインデックスは他の二手法よりも占有物理記憶領域が小さくなるものと考えられる．また，追加操作や更新操作も他の二手法より短時間で処理できると考えられる．したがって，データ要素の多重参照が多い環境ではマルチインデックスが適していると考えられる．

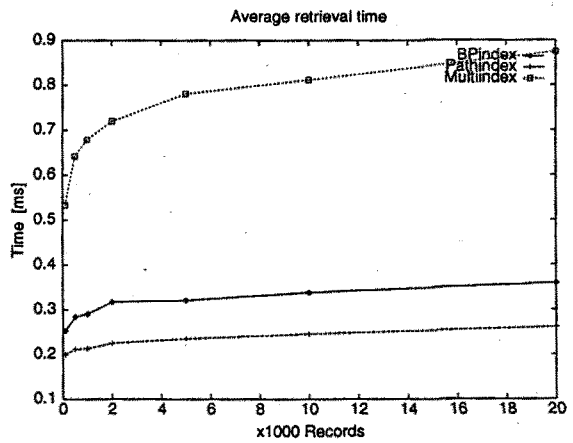
ASN.1 を利用する通信アプリケーションでは，インデックスを付加すべきパスや多重参照の有無などが前もってわかっている場合が多い．したがって，BP インデックスを拡張し，多重参照が行なわれる部分でインデックスを分割する機能を付加することにより，BP インデックスがより有効に機能すると考えられる．

#### 3.6.2.5 インデックスのキー

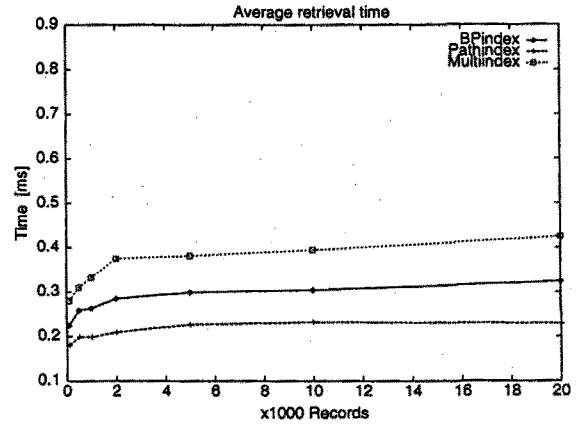
ASN.1 データベースシステムでは，インデックスのキーは ASN.1 の符号列の形式で管理する．符号列をキーとすることの利点は，以下の二点である．




(a) パス長: 7, 共通部分長: 6

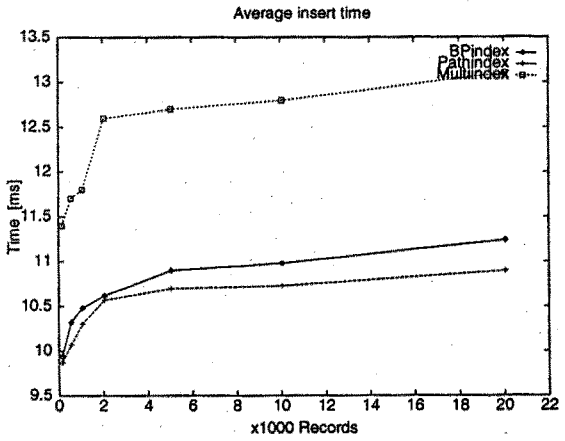



(b) パス長: 7, 共通部分長: 1

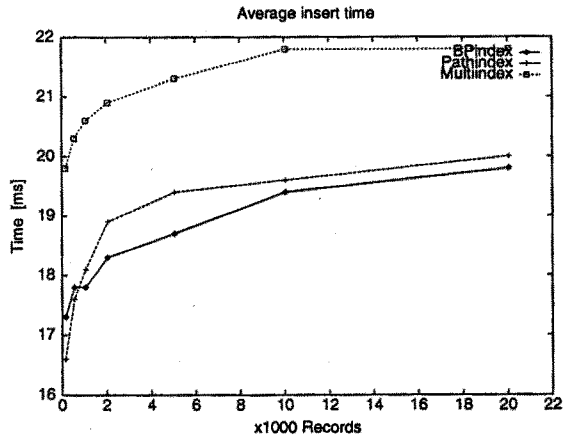



(c) パス長: 3, 共通部分長: 2

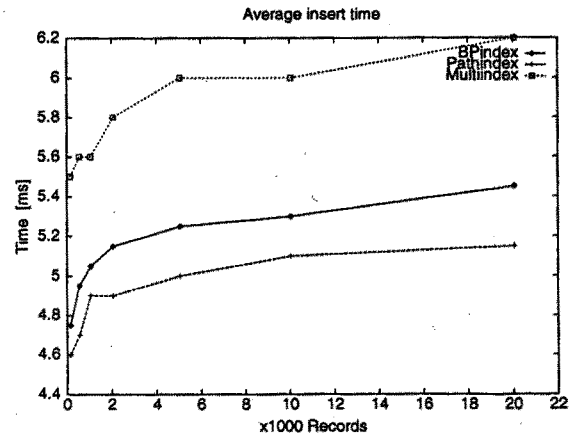
図 3.22: データ検索処理時間の比較




(a) パス長: 7, 共通部分長: 6

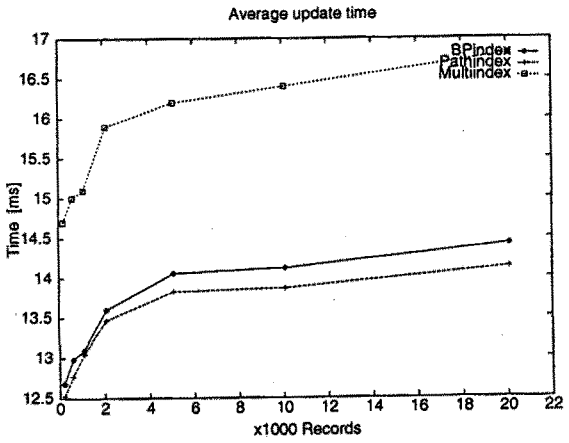



(b) パス長: 7, 共通部分長: 1

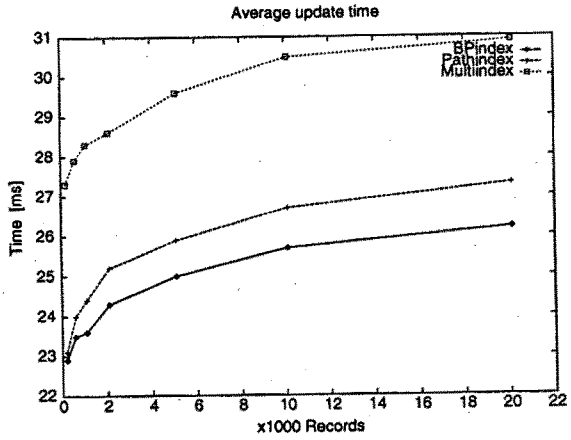



(c) パス長: 3, 共通部分長: 2

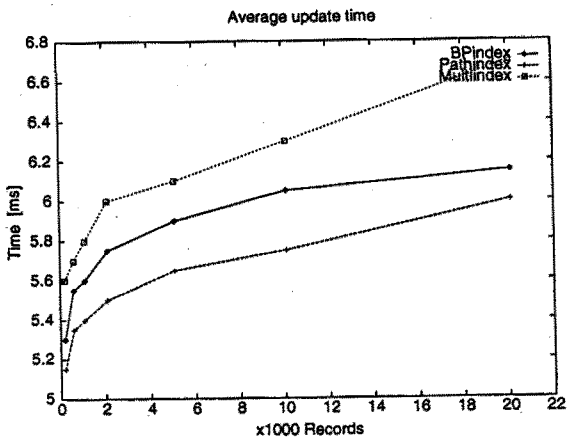
図 3.23: データ追加処理時間の比較




(a) パス長: 7, 共通部分長: 6




(b) パス長: 7, 共通部分長: 1




(c) パス長: 3, 共通部分長: 2

図 3.24: データ更新処理時間の比較

- 符号列は単なるオクテット列であるため，キー値の比較の際にデータ型を考慮する必要がない．
- 構造型のデータでも符号化すればインデックスのキーとすることができる．

ただし，符号列をキーとした場合，ASN.1の基本符号化規則の性質から，同じデータ値を表す符号列が一意に定まらないという問題が生じる．例えば，整数値を表す符号列のオクテット数は任意である．また，符号列の長さを表すのに固定長形式（実際の長さを長さフィールドに符号化する）と不定長形式（長さフィールドには不定長形式であることが示され，符号列の末尾に特別な印を付加する）の二つの形式が許されている．しかし，符号化規則の柔軟性に基づくこれらの問題は，符号化方法を統一することによって解消できる．

もう一つの問題は，SET OF型のデータ項目をキーとする場合におこる．SET OF型のデータは集合を表すため，SET OF型のデータ値の比較を行なうのは困難である．この解決策としては以下のようなものが考えられる．

1. 符号化の際，まずSET OF型のデータの各要素を符号化し，それらを辞書順でソートしたものを連結することにより，一意な符号化を実現する．
2. SET OF型のデータをキーとするインデックスを作成することは許すが，その扱いはSEQUENCE OF型と同じとし，要素の順序はアプリケーション側の責任で管理する．

1.の手法ではその処理のオーバヘッドにより検索時間が遅くなることが予想される．特に，SET OF型が入れ子になっているようなデータ構造を考えると非常に非効率的である．したがって，アプリケーション作成者の負担は大きくなるものの，一般には2.の手法がよい性能を示すと考えられる．

## 3.7 むすび

本章では，ASN.1を利用する通信アプリケーションに対してデータ蓄積・管理機能を提供するASN.1データベースシステムの設計・構築手法に関して論じた．まず，

ASN.1 データベースシステムに求められる要件について述べ，特にプログラミングにおけるインピーダンスミスマッチによる問題点，および，高速なデータベース処理が要求されることを述べた．そして，それらの問題点を解決する手法として，新たに ASN.1 データベースシステムのためのプログラミング言語 ASN.1/PL を提案した．また，ASN.1 によるモジュール定義をそのままスキーマ定義とできるような OODBMS をデータ格納部として利用する ASN.1 データベースシステムの実現手法について述べた．さらに，通信アプリケーションの特性を考慮した上でさらに高速なデータベース処理を実現するためのデータ格納方式について，クラスタリングとインデクシングの両面から議論した．

以上の ASN.1 データベースシステムの設計・構築手法を用いれば，さまざまな特性をもつ通信アプリケーションにおける汎用的かつ高速なデータ蓄積・管理部を構築でき，またアプリケーション開発効率も向上すると考える．現在および将来のインターネットの利用の拡大を考えた際，異機種間のデータ通信はますます重要になると考えられ，本章で論じた設計・構築手法はさまざまなアプリケーションを開発する上で重要な開発基盤となることを確信する．

# 第 4 章

# 広帯域ネットワークにおける分散データベース処理手法

近年，ATM 交換方式などの発展により，計算機ネットワークの帯域幅の拡大が進んでいる．このような状況下で，従来はデータ伝送量を削減させることが性能向上の要因であったのに対し，これからは計算機ネットワークの帯域幅をいかに有効利用して性能向上を図るかが大きな課題となる．特に，広帯域ネットワークでは大量のデータの転送を短時間で行なえることから，計算機ネットワークを介したデータベース移動をデータベース処理に利用することも現実的に可能である．そこで，本稿では，ATM ネットワークの仮想 LAN 環境を想定し，従来のトランザクション処理手法とは異なるデータベース移動を用いたトランザクション処理手法を提案する．また，シミュレーションによって，提案した手法と従来の二相コミットプロトコルを用いたデータベース固定型のトランザクション処理手法との性能比較を行なう．

## 4.1 まえがき

近年，ATM 方式を中心として，計算機ネットワーク技術が急速に発展している．特に著しいのが帯域幅の拡大であり，数百 Mbps や数 Gbps の帯域幅をもつ計算機ネットワークも構築されつつある．帯域幅の拡大は，大量のデータの短時間での転送を可能にし，クライアント・サーバ型のシステムでの応答時間の短縮をもたらすものと期待される．特に，現在盛んに研究が行なわれているサイエンティフィックデータベースや CALS などでは，マルチメディアデータを多く扱うことから，ATM における帯域幅，QoS (Quality of Service) の保証といった性質が有効になる．移動体計算環境においても，バックボーンとして ATM などの広帯域ネットワークを用いることにより，柔軟性に富んだ高度な通信環境を提供することが可能となる．

一方，帯域幅の拡大は，分散処理において通信コストがボトルネックであったこれまでの状況が一変することを意味している．つまり，計算機ネットワーク上の分散処理に関する研究は，伝送データ量の削減による性能向上から，帯域幅の有効利用による性能向上を目的とした技術の開発へと移行する必要がある．データベース分野においても，従来はデータベースは特定のサイトに固定され，それらに対する処理はメッセージによる処理依頼で行なうなど，伝送データ量を削減する手法を開発することが一般的であった．しかし，広帯域ネットワークでは，データベース自体を計算機ネットワークを介して短時間で移動させること（以下，データベース移動と呼ぶ）も現実的に可能である．

ここで，データベース移動の対象となるのは，大型計算機センターなどで集中管理しているような，位置依存性が高く，しかもギガバイト，テラバイトオーダにも至るような超大規模なデータベースではなく，計算機ネットワークで相互接続された分散システムにおける，位置依存性がそれ程高くない百メガバイトオーダまでのデータベースである．例えば，全世界規模のイントラネットを構築している企業で社内情報を共有するような場合がこれに当たり，このような分散システムは計算機ネットワーク技術の発展に伴い，これから益々盛んに構築されていくものと考えられる．

データベース移動を用いることにより，これまでの分散データベース技術とは異なる新たな可能性が出てくる．例えば，従来のように各データベースをいずれかのサイトに固定して配置しておき，処理依頼および二相コミットプロトコル (two-phase commit protocol, 2PC) によってトランザクション処理を行なった場合，各サイトに分散するデータベースに対する処理依頼メッセージとその応答メッセージの交換が複数回行なわれた後，二相コミットのためのメッセージ交換を二往復する必要がある．一方，データベース移動を用いてアクセス対象となるデータベースをトランザクション発生サイトに移動してしまえば，その後のメッセージ交換の必要がない．広帯域ネットワークでは，大量のデータの転送は短時間で行なえるが，伝搬遅延は従来のネットワークとほとんど差がないことから，データベース移動を利用したトランザクション処理時間の短縮が期待できる．

一方，データベース移動を現実的なものにするためには，データベースアクセスの高速化やデータベースに付加しているインデックスの移動などの課題がある．低価格化，高性能化が急速に進むメモリ技術によって，近い将来には数ギガバイトの主記憶を持つマシンが一般的になると考えられることから，前者は，主記憶データベース技術を用いることで解消できる．後者に関しては，文献 [SA+94] などにおいてデータ項目の移動を利用したシステムで採用されている二次インデックスの移動技術が利用できる．

このように，データベース移動のための技術課題が着実に解消されていき，更にネットワークの帯域幅が急速に拡大しているのとは対照的に，通信の伝搬遅延に関しては光の伝搬速度が決まっているために大幅な改善は不可能である．したがって，ネットワークの広域化，データベース処理の多様化が進んでいくにつれて，データベース移動が強力で有効なデータベース技術の一つになり，様々な用途に有効に活用されるものと考えられる．

本章では，データベース移動を利用したトランザクション処理手法を提案する．一般に，トランザクションには数レコードを対象とした簡単な書込み操作のみのものや複数の結合操作などを含む複雑な処理を要するものなど様々な形態があり，すべてにおいてデータベース移動による処理の方が 2PC を用いたデータベース固定型の

処理 (以後，固定処理と呼ぶ) よりも効率的であるとは限らない．したがって，トランザクションの複雑性やアクセスパタンに応じて固定処理とデータベース移動の二手法を選択する適応型の処理手法を考える．また，シミュレーションにより，提案する手法の性能評価を行なう．

以下，4.2節で想定する分散環境を示し，4.3節で提案する手法について説明する．4.4節では，性能評価のためのシミュレーションの結果を示し，その結果に対する考察を行なう．4.5節で関連研究との比較を行ない，最後に4.6節で本章のまとめを行なう．

## 4.2 ATM 環境のモデル

ATM は，基本的にはコネクション型の交換方式であり，データ交換を行なう前に通信したいサイトとのコネクション（ATM では仮想チャネルという）を確立する必要がある．コネクションの種類としては，ATM 交換機においてあらかじめルートを設定しておく PVC (Permanent Virtual Channel) と，サイトからの要求によってルートを設定する SVC (Switched Virtual Channel) があり，それぞれに一対一通信のためのポイント・ツー・ポイントコネクションと一対多通信のためのポイント・ツー・マルチポイントコネクションがある．PVC を利用する場合は，コネクションの設定時間を省略することができる．

ATM ネットワークでは，あるサイトから複数のサイトへあらかじめポイント・ツー・マルチポイントコネクションの PVC を設定しておくことにより，そのサイトをマルチキャストサーバ (MCS) として特定のグループへのマルチキャストが可能となる．つまり，物理的な接続形態とは異なる仮想的な LAN の構築が可能となり，仮想 LAN 内のデータのブロードキャストは，MCS に対してデータを送信することで実現できる（図4.1）．これは，基本的にコネクション型の ATM ネットワークにおいて，仮想 LAN 内の全サイトへの処理メッセージの送信が容易に行なえることを示している．

本論文では，仮想 LAN 内の分散データベースに対する分散処理を想定する．し

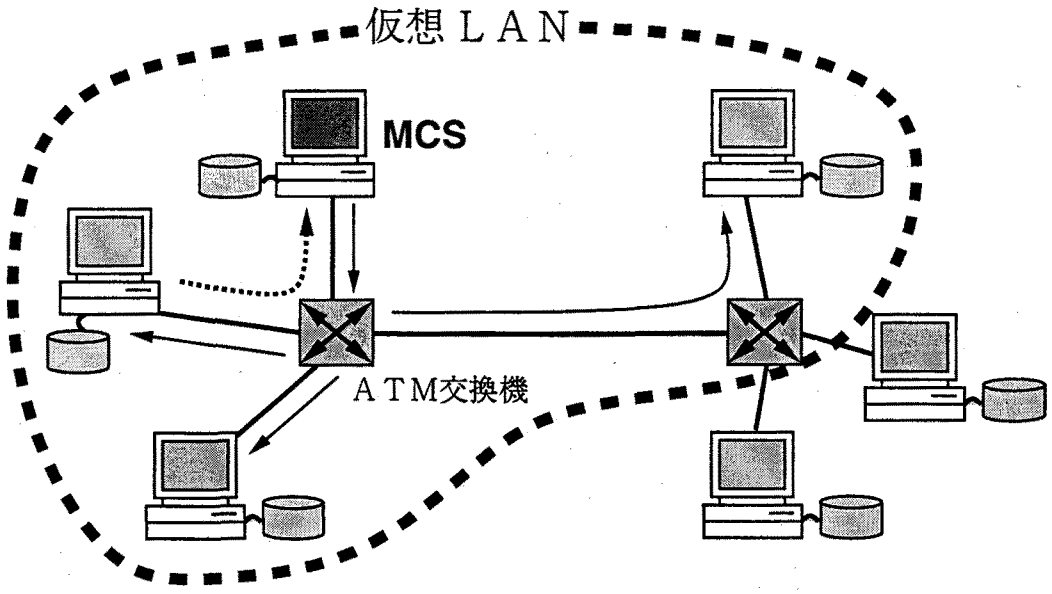



図 4.1: 仮想 LAN とメッセージのブロードキャスト

たがって，トランザクション発生サイトからのメッセージのブロードキャストには，MCS から仮想 LAN 内の各サイトに対して設定されている PVC を利用する．その他のメッセージやデータの交換に用いる一対一通信に関しては，必要時に SVC によるコネクションを確立する．ATM では，輻輳が起こった場合や通常 20 分程度の長い時間その通信路を使用しなかった場合以外はコネクションが解放されることはないため，一度確立された SVC は一つのトランザクション内で再設定する必要がないものとする．また，簡単のために，仮想 LAN 内の任意の二サイト間の平均伝搬遅延を定数 $d_m$ で表し，任意のサイトから MCS までの平均伝搬遅延を定数 $d_{MCS}$ で表す．なお，高速なデータベースアクセスを実現するために，仮想 LAN 内のデータベースはすべて主記憶データベースとし，各々に一意な識別子 DB-id ($D_j$) を付与する．

## 4.3 トランザクション処理手法

本節では，まず固定処理とデータベース移動を用いた処理の通信所用時間の定式化を行なう．その結果をもとに，データベース移動に基づく適応型トランザクショ

ン手法を提案する．

### 4.3.1 処理の処理時間の定式化

後述の適応型トランザクション処理手法での選択基準として利用する，固定処理とデータベース移動を用いた処理の通信所用時間を定式化する．ただし，この計算を後述の手法内で動的に行なうことを考慮し，システムに対して大きな負担とならない，簡単な式で近似的に表す．

#### 4.3.1.1 固定処理

固定処理における通信手順を図4.2(a)に示す．まず，トランザクション発生サイトからトランザクションの処理に必要なデータベースを所持する各サイト（サイト数$k$）へのコネクションを設定しなければならない．これに要する時間を$C(k)$と表す．コネクション設定後は，処理操作などに必要な通信が$n$回行なわれ，これに$nd_m$の時間を要する．最後に，2PCのためのメッセージ交換が二往復行なわれるため，これに$4d_m$の時間を要する．ただし，広帯域ネットワークの特性を考慮すると，処理依頼メッセージや処理結果（中間結果を含む）のサイズは通信帯域幅に比べて小さいことから，各メッセージの転送遅延は0に近似している．また，2PCのための遅延は，トランザクション発生サイトとトランザクションに関係した複数（もしくは一つ）のサイト間の伝搬遅延のうち最大のものになるため，正確には$d_m$より大きくなるが，ここでは簡単のために$d_m$としている．以上のことから，固定処理における通信所要時間$T_{fix}$は次のように表される．

$$T_{fix} = (n+4) \cdot d_m + C(k) \tag{4.1}$$

#### 4.3.1.2 データベース移動を用いた処理

図4.2(b)に示すように，データベース移動を用いた処理は，データベース移動要求，データベース移動，移動完了通知の3回の通信で実現される．ここで，トラン

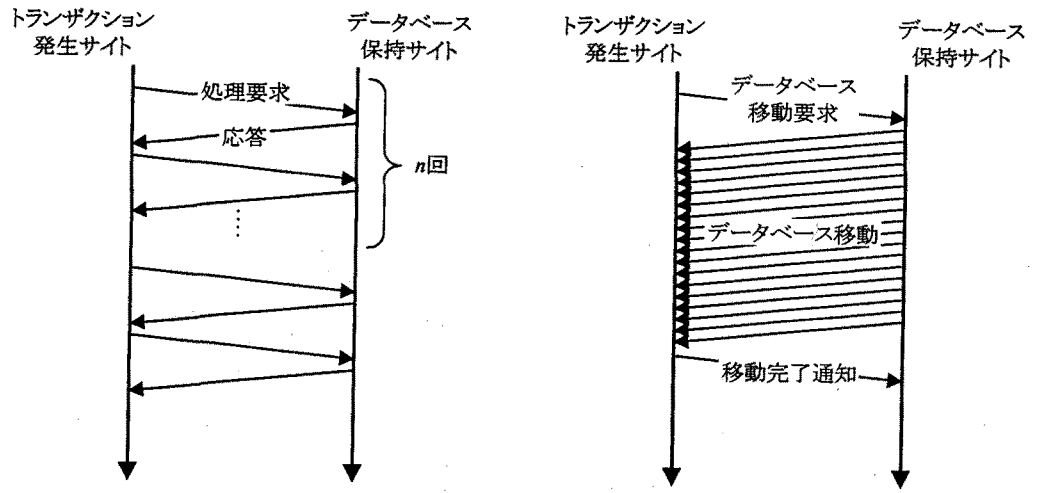



図 4.2: 各手法の通信手順

ザクションの開始時にはトランザクションの処理に必要なデータベースが同定できるものと仮定する．

まず，データベース移動要求のためのメッセージをブロードキャストする．そのために，最初に MCS までメッセージを送信し，その後 MCS によって全サイトへと転送されるため，$d_{MCS} + d_m$の時間を要する．ただし，MCS から各サイトへの実質的な伝搬遅延は，必要なデータベースを所持しないサイトもあり，見積もることが困難であるため，簡単のために定数 $d_m$としている．また，転送遅延は固定処理の場合と同様に 0 に近似している．ここで，データベース移動要求のメッセージをブロードキャストしているのは，全サイトが各データベースの正確な位置情報を保持することを可能にするためである．

次に，データベース移動を行なうために，トランザクション発生サイトとデータベースを所持しているサイト（サイト数 $k$）との間に SVC コネクションを設定しなければならない．これには固定処理と同じく $C(k)$ の時間を要する．コネクション設定後，データベース移動が実行される．データベース $D_j$のサイズを $Size(D_j)$，トランザクションの処理に必要でかつトランザクション発生サイトが所持していないデータベースの集合を$\boldsymbol{D_N}$，データベース移動のための予約帯域幅を $B_M$とす

ると，データベース移動では転送遅延を無視できないため，これに要する時間が$d_m + \sum_{D_j \in \boldsymbol{D}_N} Size(D_j)/B_M$となる．ここでも実際の伝搬遅延は，2PC の場合と同様の理由で$d_m$より大きくなるが，簡単のため$d_m$としている．また，一般的には$B_M$は分散システムの特性，つまり，データベースのサイズやトランザクションのアクセスパタンなどからシステム管理者が適当な固定値を割り当てるものである．ただし，システム側でネットワークの負荷状態をある程度把握することができる場合には，各サイトで動的に$B_M$の値を決定することが望ましい．

最後に，移動完了通知がトランザクション発生サイトからブロードキャストされ，これに$d_{MCS} + d_m$の時間がかかる．このメッセージをブロードキャストすることにより，全サイトは移動が完了したことを知り，データベースを転送したサイトは移動失敗時に備えて保持しておいたコピーを消去することができる．

最終的に，データベース移動を用いた処理の通信所要時間$T_{DB}$は次のようになる．

$$\begin{aligned} T_{DB} &= 3d_m + 2d_{MCS} + C(k) \\ &\quad + \sum_{D_j \in \boldsymbol{D}_N} Size(D_j)/B_M \end{aligned} \tag{4.2}$$

### 4.3.2 データベース移動に基づくトランザクション処理手法

前節の定式化の結果から，固定処理では処理に必要な通信回数 ($n$) が，データベース移動を用いた処理では移動するデータベースの総サイズが，処理時間に大きく影響することが分かる．

ここでは，4.3.1節の解析に基づいて，トランザクション実行時に固定処理とデータベース移動を用いた処理の効率的な方を選択する適応型の処理手法を提案する．提案する手法は，単純手法と履歴統計手法の二つである．両手法とも，次の3ステップで構成される．

ステップ 1: トランザクション解析

処理に必要なデータベース，固定処理でのメッセージ交換回数などを同定する．

**ステップ 2:** 手法選択

固定処理もしくはデータベース移動を用いた処理のどちらを実行するか選択する．選択の方法については，4.3.2.1節および4.3.2.2節で述べる．

**ステップ 3:** トランザクション実行

ステップ 2 で選択された手法を用いてトランザクションを実行する．

#### 4.3.2.1 単純手法

変数 $t_1$ を次のように定義する．

$$t_1 = T_{DB} - T_{fix} \tag{4.3}$$

このとき，単純手法のステップ 2 は次のように表される．

if $t_1 < 0$ then データベース移動を用いた処理
else 固定処理

ここで，$T_{DB}$ を計算するためには，移動しなければならないデータベースの総サイズ $\sum_{D_j \in \boldsymbol{D}_N} Size(D_j)$ を知る必要があるため，仮想 LAN 内の各サイトで，DB-id とその DB-id をもつデータベースのサイズ情報の表を保持するようにする．これにより，ステップ 1 で処理に必要なデータベースが同定されれば，$\sum_{D_j \in \boldsymbol{D}_N} Size(D_j)$ の値を計算することができる．また，各サイトにおいて，自分のサイトにあるデータベースのサイズが変動するたびに全サイトに知らせるのは非効率的であるため，サイズ情報の表で保持している値と実際の値の差が一定値を越えた場合にのみその値をブロードキャストする．

#### 4.3.2.2 履歴統計手法

単純手法では，一つのトランザクションを処理する上で固定処理とデータベース移動を用いた処理のうち効率的な手法を選択した．しかし，一つのトランザクションに関しては効率的である選択が，長期的には非効率的であるといったことが生じ

| DB-id | Size | Current Loc | Cont | Usage-log | | | | |
|---|---|---|---|---|---|---|---|---|
| $D_1$ | 50 | $S_1$ | 1 | $S_1$ | | $\cdots$ | $S_2$ | $S_1$ |
| $D_2$ | 35 | $S_2$ | 0 | $S_1$ | $S_4$ | $\cdots$ | | |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | | | $\vdots$ | | |
| $D_m$ | 40 | $S_4$ | 1 | | | $\cdots$ | $S_2$ | $S_1$ |

図 4.3: DB 情報表

得る．例えば，現トランザクションだけで判断すると固定処理の方が効率的な場合でも，連続して同じサイトから同一のデータベースを用いるトランザクションが発生する場合には，先にデータベース移動で処理をしておいた方が効率的である．

そこで，ここで提案する履歴統計手法では，次の二点に注目する．

- 各データベースは，その利用頻度が (特に最近) 高いサイトに位置するべきである．
- サイト $S_I$が特定のデータベース $D_j$を連続使用することが分かっている場合，サイト $S_I$が $D_j$をもつ優先度を上げるべきである．

つまり，履歴統計手法では，単純に式 (4.3) によって手法を選択するのではなく，履歴情報や連続使用の情報を保持することにより，上述のことを考慮した選択を行なう．履歴統計手法で保持する DB 情報表を図 4.3に示す．図中の表の属性は，次の記法を用いている．

**DB-id:** データベース識別子（DB-id）
**Size:** DB-id で示されるデータベースのサイズ
**Current Loc:** データベースの現在位置（現サイト）
**Cont:** 現サイトが連続使用を指定しているか
0: 連続使用しない，1: 連続使用する
**Usage-log:** 利用したサイトの履歴
—列が一つのトランザクションを示す

Cont と Usage-log はトランザクション毎に更新されるため，更新情報をトランザクション毎にブロードキャストする必要がある．データベース移動で処理する場合には，Usage-log は，データベース移動要求，移動完了通知メッセージの内容から直接決定される．Cont の更新情報もこれらのメッセージに付加して転送することが可能である．一方，固定処理において，これらの更新をデータベース移動を用いた処理と同様の方法で全サイトに伝えるためには，コミット準備，コミット/アボートのメッセージを処理に関わったサイトだけでなく，すべてのサイトにブロードキャストする必要がある．したがって，履歴統計手法では固定処理の通信所要時間が式 (4.1) より $2d_{MCS}$だけ大きくなる．

ここで，データベース $D_j$がサイト $S_I$に存在する有効性を示す目的関数 $f(S_I, D_j)$ を次のように定義する．

$$f(S_I, D_j) = \alpha \cdot P \cdot |L| + \sum_{l=1}^{|L|} \{k_l \cdot (|L| + 1 - l)\} \tag{4.4}$$

$$\alpha = \begin{cases} 1: & S_I \text{が} D_j \text{の連続使用を宣言している} \\ & \text{あるいは 現トランザクションで} D_j \\ & \text{の連続使用を宣言する．} \\ 0: & \text{その他．} \end{cases}$$

$P$: 連続使用優先係数

$|L|$: データベース利用履歴のサイズ

（過去 $L$ 個のトランザクションの履歴を残す）

$$k_l = \begin{cases} 1: & l \text{回前のトランザクションでサイト} \\ & S_I \text{が} D_j \text{を利用．} \\ 0: & \text{その他．} \end{cases}$$

$f(S_I, D_j)$ は，前述の履歴統計手法が注目している二点のうち第一項で第二点（連続使用宣言しているサイトの優先）を，第二項で第一点（利用頻度が高いサイトの優先）を反映している．ここで，連続使用優先係数 $P$ は，連続使用宣言に対しての優先度の度合を表すものである．

次に，データベース $D_j$ を現在位置するサイト $S_C$ からサイト $S_I$ に移動する有効性を示す評価関数として，移動評価関数 $G(S_I, D_j)$ を次のように定義する．

$$G(S_I, D_j) = f(S_I, D_j) - f(S_C, D_j) \tag{4.5}$$

サイト $S_I$ が，他サイトにあるデータベース集合 $\boldsymbol{D}_N$ を必要とするトランザクションを発生するとき，評価値 $t_2$ を $G(S_I, D_j)$ の定義に基づいて各 $D_j \in \boldsymbol{D}_N$ の $G(S_I, D_j)$ の平均値として次のように定義する．この $t_2$ は，現トランザクションをデータベース移動を用いて処理することの有効性を表している．

$$t_2 = \frac{1}{|\boldsymbol{D}_N|} \sum_{D_j \in \boldsymbol{D}_N} G(S_I, D_j) \tag{4.6}$$

最後に，固定処理とデータベース移動を用いた処理のどちらを選択するかを示す値として $t_{sel}$ を次のように定義する．

$$t_{sel} = t_1 - K \cdot t_2 \tag{4.7}$$

$K$ は $t_2$ の係数であり，この $K$ の値の設定がシステム全体の性能に大きく影響する．したがって，システム管理者などの専門家が，ネットワークやデータベースの規模，アプリケーションの性質に応じてこの値を決定する．以下では $K$ を履歴依存係数と呼ぶ．

履歴統計手法のステップ 2 は次のようになる．

if $t_{sel} < 0$ then データベース移動を用いた処理
else 固定処理

## 4.4 シミュレーション

本節では，提案した手法の性能評価ために行なったシミュレーションの結果を示す．更に，履歴統計手法における連続使用優先係数 $P$，履歴依存係数 $K$ の影響についても，シミュレーションによって示す．

表 4.1: シミュレーションのためのパラメータの設定

| パラメータ | パラメータの意味 | 値 |
|---|---|---|
| $\|S\|$ | 仮想 LAN 内のサイト数 | 20（サイト識別子 $S_1, \cdots, S_{20}$） |
| $\|D\|$ | データベースの総数 | 20（DB-id $D_1, \cdots, D_{20}$） |
| $Size(D_j)$ | 各データベースのサイズ | 30+1.5$i$ [Mbytes] ($i$=0,$\cdots$,19) |
| $p_1$ | 最初の 100 回のトランザクション発生確率 | 0.025 ($S_1, ..., S_{10}$)<br>0.05 ($S_{11}, ..., S_{18}$)<br>0.15 ($S_{19}$)<br>0.3 ($S_{20}$) |
| $p_2$ | 後の 100 回のトランザクション発生確率 | 0.025 ($S_{11}, ..., S_{20}$)<br>0.05 ($S_3, ..., S_{10}$)<br>0.15 ($S_2$)<br>0.3 ($S_1$) |
| $d_{MCS}$ | MCS への平均伝搬遅延 | 0.12 [s] |
| $d_m$ | その他の 2 サイト間の伝搬遅延 | 0.12 [s] |
| $B_M$ | データベース移動のための予約帯域幅 | 1 [Gbps] |
| $C(k)$ | コネクション設定時間 | $C(k) = 0.3k$ [s] |
| $n$ | 固定処理における通信回数 | $n'\|\boldsymbol{D}_N\|/\|\boldsymbol{D}_U\|$<br>($n'$は 1〜30 でランダムに決定) |
| $\|L\|$ | 利用履歴のサイズ | 20 |
| $P$ | 連続使用優先係数 | 8.00 |
| $K$ | 履歴依存係数 | 0.07 |

### 4.4.1 各手法の性能比較

提案した二つの適応型トランザクション処理手法を，シミュレーションによって性能評価し，固定処理およびデータベース移動のみを用いた処理と比較した．シミュレーションでは，200 回のトランザクションを連続的に発生させ，その処理に要する通信所要時間の総和を求めた．

用いたパラメータの値を表 4.1に示す．分散システム内のサイト数およびデータベース数はそれぞれ 20 とした．最初の 100 回のトランザクションと後の 100 回のトランザクションとで各サイトのトランザクション発生確率を変化させているが，これは提案した手法の環境変化への適応性を検証するためである．通信回数を 1〜30

でランダムに変化させることで，簡単な書込み操作のみのトランザクションや結合操作を複数含むような複雑なトランザクションなど様々な性質のトランザクションを表現できる．また，データ転送に関するパラメータは，近い将来に実現されると思われる全世界規模の分散システムを想定したものである．つまり伝搬遅延は，光ファイバケーブルを用いたATMネットワークで各サイト間の平均ホップ数（経由する交換機数）が10程度，各交換機でのルーチング遅延が市販のATM交換機同様に5ミリ秒程度と仮定して設定している．データベース移動のための帯域幅は，ネットワーク全体の帯域幅が数Gbps程度になることを予想して1 Gbpsと設定している．ここで，$B_M$として固定値を用いているが，ネットワーク全体の帯域幅は十分広く，$B_M$の確保に失敗することはないものと仮定している．また，データベースのサイズは，アプリケーションの性質などに依存し，この値の設定がデータベース移動を用いた処理の通信所要時間に大きく影響するが，今回は，最も顕著な例として，固定処理の通信所要時間とデータベース移動を用いた処理の通信所要時間がほぼ等しくなるように設定した．なお，連続使用優先係数$P$，履歴依存係数$K$は，4.4.2節で示すシミュレーションによって選択した最適値を用いている．

シミュレーションの手順は，次のステップの繰り返しである．

1. トランザクション発生確率に応じてトランザクションの発生サイトを決定する．

2. トランザクションの処理に必要なデータベースの総数$|\boldsymbol{D}_U|$を1から$|\boldsymbol{D}|$の範囲でランダムに決定する（$|\boldsymbol{D}_U| \geq |\boldsymbol{D}_N|$）．もし，トランザクション発生サイト$S_I$があるデータベースに対して連続使用宣言をしている場合，$|\boldsymbol{D}_U|$の値が連続使用宣言をしているデータベースの総数よりも小さければ，$|\boldsymbol{D}_U|$をその値に再設定する．

3. トランザクションの処理に必要なデータベースとして$|\boldsymbol{D}_U|$個のデータベースを選択する．ただし，連続使用宣言をしているデータベースは必ずこれに含まれなければならない．

4. トランザクションの連続度$\beta$を0,1,2の中からランダムに決定する．この値は，現トランザクション以後に$S_I$が何回連続してトランザクションを発生するか

を表している．もし，この値が0より大きい場合は，以後$\beta$回，$S_I$のトランザクション発生確率を1.00増加させる．

5. 連続使用するデータベースを3.で決定したデータベースの中からランダムに決定する．

6. （提案した手法のみ）固定処理もしくはデータベース移動を用いた処理を選択する．

7. （選択された手法に応じて）通信所要時間の総和を増加させる．ただし，各手法の通信所要時間は，式(4.1)，(4.2)より得られる計算値を実際の値と考える．履歴統計手法では式(4.1)より$2d_m$大きい値を使用する．

シミュレーションの結果を図4.4に示す．グラフにおいて横軸はトランザクション処理数，縦軸は通信所要時間の総和である．また，'DB-migration'はデータベース移動を用いた処理，'Fixed-processing'は固定処理，'Simple method'は単純手法，'Log-statistics method'は履歴統計手法をそれぞれ表している．

この結果より，提案した適応型手法の有効性が認められる．特に，履歴統計手法では，各サイトのデータベースの利用頻度やトランザクションの連続性を考慮して手法選択を行なうため，単純手法より良い結果を示している．

### 4.4.2 連続使用優先係数 $P$，履歴依存係数 $K$ の影響

本節では，連続使用優先係数$P$，履歴依存係数$K$が履歴統計手法の通信所要時間に与える影響をシミュレーション結果より考察する．

4.4.1節のシミュレーション環境において，$P$，$K$の値を$P$は0から10まで0.5ずつ，$K$は0から2.0まで0.05ずつ変動させて履歴統計手法の通信所要時間を計測した結果を図4.5に示す．グラフでは，x軸が$K$の値，y軸が$P$の値，z軸が通信所要時間を表している．図4.5の結果より，$P$，$K$の値によりトランザクション処理時間が大きく変動することが分かる．この環境では，$K = 0 \sim 0.10$の範囲（図の灰色の部分）で良い結果が観測される．

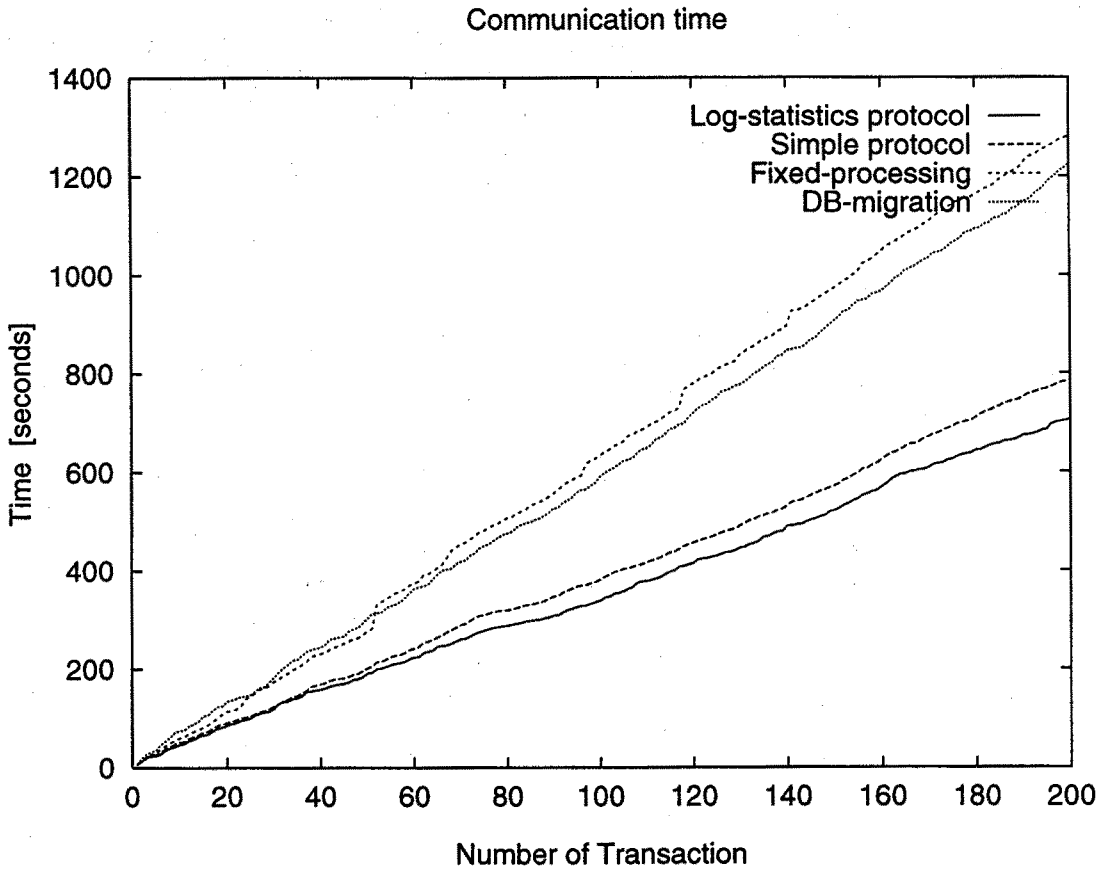



図 4.4: 各手法の通信所要時間の比較

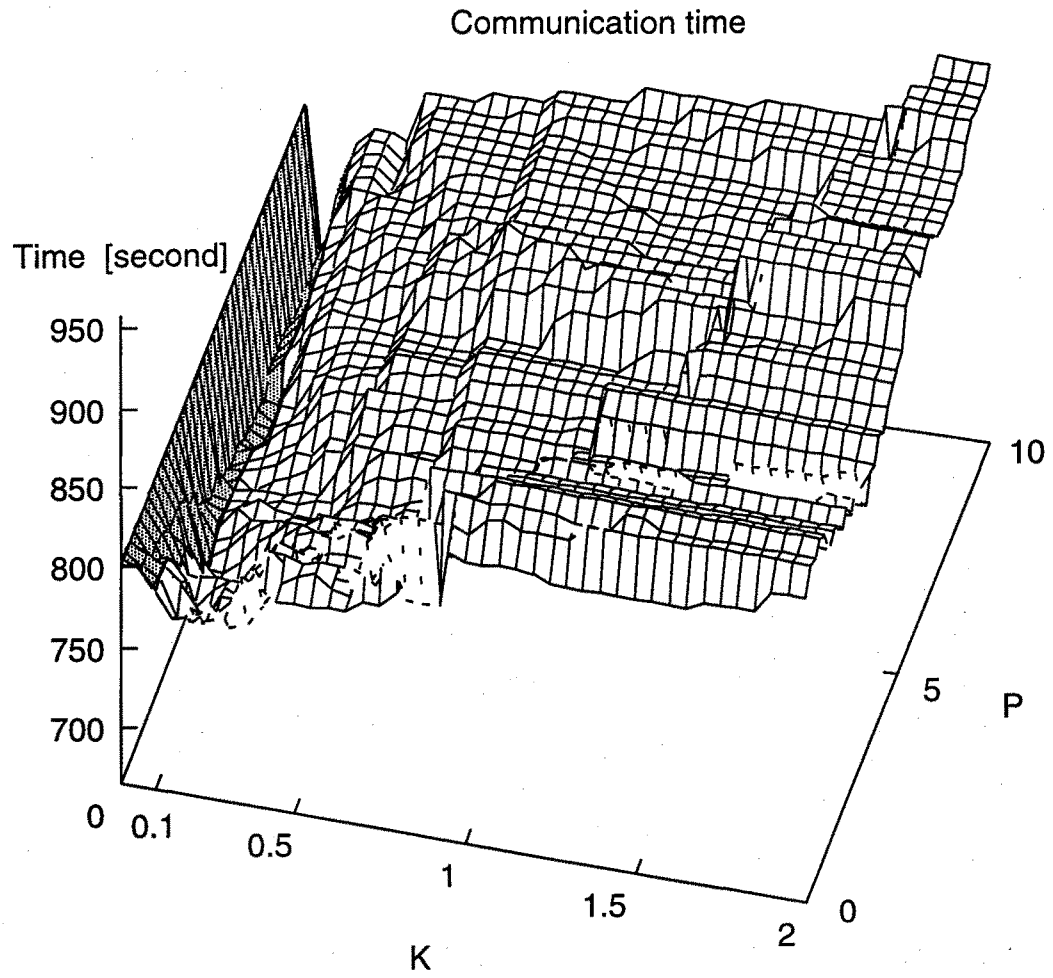



図 4.5: 履歴統計手法における通信所要時間に対する履歴依存係数 $K$ および連続使用優先係数 $P$ の影響

そこで，さらに詳しく観察するために，$K = 0.00, 0.01, 0.02, \cdots, 0.10$ の場合に $P$ を 0 から 15.0 まで 0.1 ずつ変動させ，同様の環境でトランザクション 200 回分の通信所要時間を計測した．$K = 0.00, \cdots, 0.05$ の結果を図 4.6に，$K = 0.06, \cdots, 0.10$ の結果を図 4.7にそれぞれ示す．図では，横軸が $P$ の値，縦軸が通信所要時間の総和を表している．なお，比較のために，単純手法を用いた場合の処理時間も記している．

この結果から，$(K, P) = (0.05, 10.1 \sim 10.9)$, $(0.06, 8.3 \sim 9.5)$, $(0.07, 7.4 \sim 8.5)$, $(0.08, 7.0 \sim 7.5)$ の範囲で通信所要時間は最小値 705.76 秒を示している．実際のシ

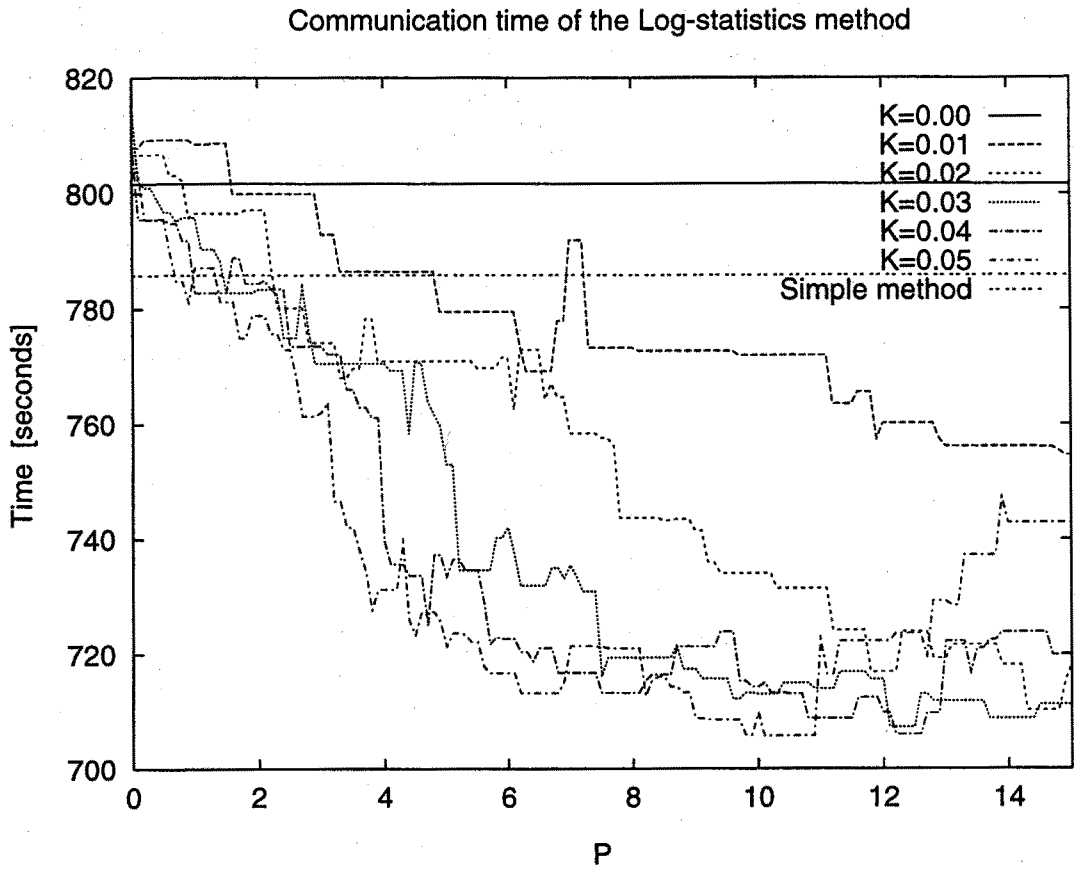



図 4.6: 履歴統計手法における通信所要時間に対する履歴依存係数 $K$ の影響 ($K = 0.00, \cdots, 0.05$)

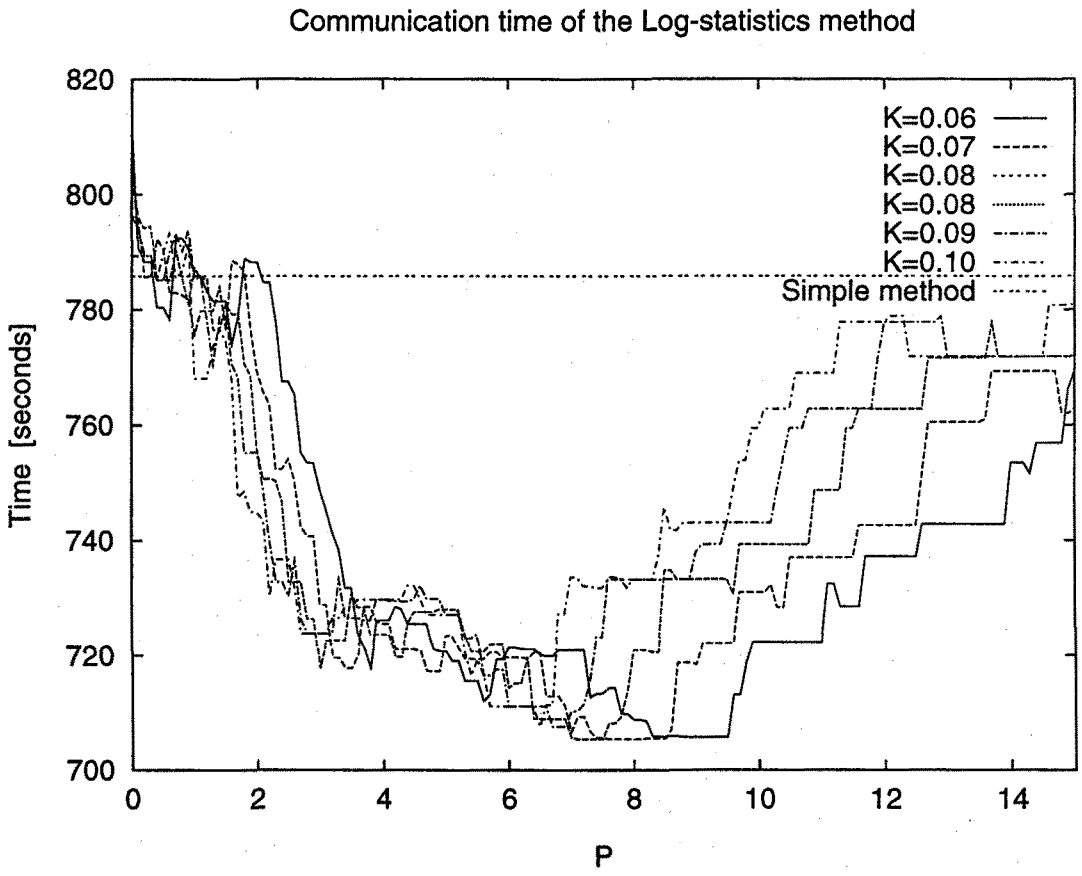



図 4.7: 履歴統計手法における通信所要時間に対する履歴依存係数 $K$ の影響 ($K = 0.06, \cdots, 0.10$)

ステムにおいて履歴適応手法を実装し運用するに当たっては，$P$，$K$ の値を詳細なシミュレーションによって決定する必要がある．ただし，シミュレーションで実際の環境を正確に表すことは困難であるため，突出的によい性能を示す $P$，$K$ 値は選択せず，広範囲でよい性能を示す $P$，$K$ 値を実システムに採用すべきである．例えば，4.4.1節でのシミュレーションにおける $P, K$ の値としては，8.00, 0.07 をそれぞれ用いた．

## 4.5 関連研究との比較

分散処理においてデータベースやデータ項目の移動を考えた研究はいくつか報告されている．文献 [Devi93],[JK93],[LNS93],[SPW90],[VBW94] などでは，相互接続されているサーバ間の負荷を均一化することを目的として，分散データの再配置を行なっている．しかし，これらの研究は広帯域ネットワークを想定しておらず，データの移動は大きなオーバヘッドとなると考えている．したがって，データの移動を頻繁には行なわず，トランザクション単位でデータベースを移動するものとは仮定が異なる．

広帯域ネットワークの特性をトランザクション処理に利用する研究としては，文献 [BLW93],[HG+87] などがある．文献 [HG+87] では，データベースを光ファイバのリング型ネットワーク上で定期的にブロードキャストする datacycle と呼ぶ手法を提案し，読出し操作のみのトランザクションが中心的な環境で高いスループットを実現している．datacycle は，データベースの内容がネットワーク上を流れるという点では本章で提案した手法と類似している．しかし，この手法では，書込み操作は特定の一つのサイトで行なっているので，本質的には分散データベースではない．また，リング型ネットワーク上でしか利用できない．

文献 [BLW93] では，トランザクションの処理に必要なデータ項目をトランザクション発生サイトに転送する send-on-demand という手法を提案している．さらに，読出し操作のみのトランザクションでは datacycle を用い，書込み操作を伴うトランザクションでは send-on-demand を用いるといった混合型の手法を提案している．

しかし，これらの手法は，datacycle を基盤としているため，リング型のネットワーク上でしか利用できない．また，すべての書込み操作をトランザクション発生サイトへデータ項目を移動することで処理しており，転送すべきデータ項目の総サイズが大きくなると処理効率が劣化する．

一方，本章で提案したトランザクション処理手法では ATM ネットワークを想定しているため，あらゆるトポロジのネットワークに適用可能である．更に，提案した手法では，固定処理とデータベース移動の予測処理時間やトランザクションのアクセスパターンを考慮して固定処理とデータベース移動を選択するため，変化のある環境下でより高い処理効率が得られるものと考えられる．

## 4.6 むすび

本章では，ATM ネットワークの特性を考慮した分散データベース処理手法として，データベース移動に基づくトランザクション処理手法を提案した．また，シミュレーションにより提案した手法の有効性を検証した．その結果より，適当な連続使用優先係数 $P$，履歴依存係数 $K$ の値を決定すれば，一般的な環境において，履歴適応手法が最も優れているということを示した．また，履歴適応手法における $P$，$K$ 値の影響，値の設定の重要性を示した．今後は，提案した手法に並行処理制御の機能を付加して実装し，プロトコルオーバヘッドを考慮した上で，提案した手法の有効性を検証する必要がある．

計算機ネットワークの帯域幅の拡大や計算機の処理能力の発展が進むにつれてデータベース移動が有効に機能する範囲が益々広がり，DB 移動は分散処理における最も強力な技術の一つになるものと考えられる．

# 第 5 章

# 結論

## 5.1 本論文のまとめ

本論文では，近年の計算機ネットワークの高度化に適応したデータベースシステム構築技術について，データベースシステムを構成する要素であるエンドユーザ，アプリケーションプログラム，データベース管理システムの三つの観点から議論した．

まず第 1 章では，計算機システムにおけるデータベースシステムが果たす役割の重要性について言及した．さらに，計算機ネットワーク環境の高度化によってデータベースシステムの応用分野や構築技術に大きな影響を与えていることを述べた上で，本論文の論点を明らかにした．

第 2 章では，データベースシステムのエンドユーザを中心に捉えた場合，世界規模の計算機ネットワークの発展によって計算機システムを利用した国際的な協調作業環境が実現可能になりつつあることを考え，エンドユーザの非同期蓄積型の協調作業を支援するデータベースシステムの構築手法について議論した．まず，従来のトランザクション処理における並行処理制御手法を協調作業環境にそのまま適用した場合の問題点について述べ，それを解決する方法として，協調作業において共有されるデータベースに課せられている一貫性制約を明示的に定義できる領域一貫性制約の概念を導入し，そのデータベース上で作業者が行なう作業の単位をジョブとして定義した．次に，協調作業環境においては，システム側で作業の進行を制御す

るだけでなく，作業者間の協議による作業の調整が可能かつ重要であると考え，作業者間の協議・協調を支援するようなジョブの協調実行制御モデルを示した．また，特に国際的な協調作業など，共同作業者が必ずしも同時に作業を進めているとは限らず，協議が必要になった際に即座に協議できない場合があることを重視し，そのような場合にでも作業を進行させることができるようなジョブの実行制御方法を示した．さらに，提案した手法がさまざまな協調作業に適用可能であることを示すために，共同文書作成や共同プログラム開発などに応用可能なCOCOAシステムの実装を行なった．

第3章では，データベースシステムを利用するアプリケーションプログラムを中心に捉えた場合，計算機ネットワークを介して情報提供を行なうような通信アプリケーションを支援するデータベースシステムが要求されていることを重視し，その構築手法について議論した．まず，通信アプリケーションにおいてはネットワーク上でやりとりされるデータの表現形式を統一する必要があることから，特にASN.1を利用する通信アプリケーションの支援を考え，ASN.1データベースシステムの要件について論じた．次に，アプリケーション開発におけるインピーダンスミスマッチの問題を解消するために，ASN.1データベースシステムにおけるプログラミング言語としてASN.1/PLを提案した．また，ASN.1データベースにおけるデータ格納部を既存のデータベース管理システムを利用して実現する手法として，特にASN.1データの蓄積に適していると考えられるOODBMSを利用することを考え，自動的にスキーマの定義を行なうASN.1コンパイラの実現方法について述べた．さらに，通信アプリケーションに要求される高速応答性を実現するためのデータ格納方式について，クラスタリングとインデクシングの両面から議論した．

ASN.1は，OSIディレクトリやネットワーク管理プロトコルであるSNMP (Simple Network Management Protocol) などで利用されているだけでなく，構造化文書の表現形式を規定したODA，マルチメディア情報の表現形式を規定したMHEG (Multimedia and Hypermedia coding Expert Group) などでも使用されている．さらに，遺伝子情報を蓄積するゲノムデータベースにおいても，遺伝子情報のデータ構造の定義およびデータ交換フォーマットとしてASN.1が使用されている．このように，

ASN.1 は複雑なデータ構造も簡単に記述できることから，今後もその利用は進むと考えられ，そのアプリケーション開発においては，本論文で提案した ASN.1 データベースシステムの構築手法が適用可能であると考える．

第 4 章では，データベースシステムの中核をなすデータベース管理システムにおけるトランザクション処理について考え，計算機ネットワークの広帯域化によって従来のトランザクション処理手法よりも高速な処理が可能になることを論じた．広帯域ネットワークでは，データベースのような大量のデータでも短時間で転送できることから，データベース移動という新たな技法を利用することによって，トランザクションをより高速に実行できる可能性があることを述べた．このような背景から，データベース移動を用いたトランザクション処理手法を提案し，シミュレーション実験によってその有効性を確認した．

また，第 4 章の中では述べなかったが，データベース移動をトランザクション処理に導入することによって発生する問題として，データベースを移動させている間の操作要求をどう処理するかという問題がある．この問題を解決する並行処理制御手法は，文献 [HH+96c] で提案している．また，別の問題として，移動するデータベースの位置管理をいかに効率的に行なうかという問題があるが，これについては文献 [HH+96b, HH+97b, HH+97c] において ATM ネットワークの特性をいかした位置管理手法をいくつか提案し，その比較評価を行なっている．さらに文献 [HH+97f] では，データベースを移動するだけではなく，動的に複製を作成する手法も提案し，その評価を行なっている．また，ディスクアクセスの速度がデータベース移動のボトルネックとなると考えられるため，文献 [HH+97g] では高速なデータベース移動を実現するために必要と考えられる主記憶データベースの構成手法を提案している．現在，これらの研究成果を踏まえたシステムの実装を開始している [AH+97]．

計算機ネットワークの広帯域化は，光ネットワーク技術が今後進展していくことによってさらに進むと考えられ，本論文で提案したデータベース移動を用いたトランザクション処理手法はさらに有効性を増していくものと考える．

## 5.2 今後の課題

計算機ネットワークの高度化および多様化は，さらにデータベースシステムの構築手法に大きな影響を与えていくものと考えられる．なかでも大きな変化として，PDA (Personal Digital Assistant) やノート型パーソナルコンピュータのような可搬型計算機（移動体），および，携帯電話やPHS (Personal Handy-phone System)，無線LANなどの無線通信技術を利用した移動体計算環境の出現がある．移動体計算環境は，いつでもどこでも計算機を利用して世界中のどの計算機にもアクセスできるという非常に魅力的な計算機環境であるため，今後広く発展すると考えられる．そのため，現在，移動体計算環境におけるさまざまなアプリケーションに関する研究が盛んに行なわれている．データベースシステムの構築手法に関しても，この移動体計算環境に適した手法を確立していくことが急務となっている．

移動体計算環境には，大きく分けて以下の三つの特性がある．

**無線通信** 有線通信よりも一般に使用できる帯域幅が狭い．さらに，雑音 (noise) や反射 (echo) に弱いため，伝送エラー率が高くなり，再送処理などのオーバヘッドのために結果的に遅延が大きくなる．また，基本的にブロードキャスト通信に適した通信方式であり，その点も考慮する必要がある．

**移動性** 移動体の移動によって，コネクションの切断と接続，ネットワークアドレスの変化などが起こる．また，移動体が移動するということは，計算機ネットワークの構成が動的かつ頻繁に変化することを意味する．

**可搬性** 移動体は通常バッテリー電源で動作するため，使用可能な時間が短い．また，消費電力を抑制するために，計画的に無線通信を切断することが起こる．さらに，処理能力や主記憶容量を大きくすると消費電力が大きくなるため，一般に移動体は処理能力があまり高くなく，主記憶容量も小さい．

以上のように，移動体計算環境には従来の有線による計算機ネットワークとは大きく異なる特性をもっていることから，これらの特性を反映したデータベースシステムを構築していくことが重要であると考える．

筆者は，移動体計算環境におけるデータベースシステムの重要性を認識し，すでに移動体計算環境におけるデータベース処理方式の検討を開始している [BH+95, BH+96b]．特に，移動体計算環境においては無線通信コストを削減することが重要であることから，文献 [BH+96, BH+97b, BH+98] において移動体がもつデータベースの複製を固定ネットワーク内に動的に配置することによってデータベースアクセスにかかる無線通信コストを削減する手法をいくつか提案し，その評価を行なっている．また，移動体計算環境では移動体の位置情報が貴重な情報源となる．しかし，位置情報は時々刻々と変化するため，位置情報問合せに対して一貫性をもって答えることは難しい．文献 [BH+97a] では，位置施錠という新たな施錠方式を提案し，これを用いた位置情報問合せの並行処理制御手法について論じている．

移動体計算環境におけるデータベースシステムにおいては，さらに多くの課題が残されている．例えば，無線通信がブロードキャスト型通信であることを有効利用し，データベースを定期的にブロードキャストすることによってデータベースアクセスにかかる無線通信コストを削減する試みがなされている．このような環境においては，移動体がデータを読み取るだけの場合は問題が起こらないが，データの更新が発生した場合にはデータベースの一貫性が犯される可能性がある．このような問題点を考慮したトランザクション処理手法の確立などが今後の課題として挙げられる．

# 謝辞

本研究を行なうにあたり，終始懇切なる御指導，御助言と格別なる御配慮を賜りました大阪大学大学院基礎工学研究科情報数理系専攻 宮原秀夫教授に深く感謝致します．

また，本論文をまとめるにあたり，貴重な時間を割いて頂き，懇切なる御指導と有益な御助言を賜りました大阪大学大学院基礎工学研究科情報数理系専攻 橋本昭洋教授，菊野亨教授，ならびに，大阪大学大学院工学研究科情報システム工学専攻 西尾章治郎教授に心より感謝致します．

また，本研究を推進するにあたり，直接の御指導，御助言，御討論を頂き，また様々な面で御世話になりました大阪大学大学院工学研究科情報システム工学専攻 塚本昌彦助教授に衷心より感謝申し上げます．

本論文は，筆者が平成4年4月から平成6年3月まで大阪大学大学院基礎工学研究科情報ネットワーク学講座において，引き続いて平成6年4月以降大阪大学大学院工学研究科情報システム工学専攻社会情報システム工学講座情報ベース工学領域において行なった研究成果をまとめたものである．その間，本研究に関する有益な御討論，御助言を頂きました大阪大学大型計算機センター 下條真司助教授，大阪大学大学院基礎工学研究科情報数理系専攻 村田正幸助教授，奈良先端科学技術大学院大学 山口英助教授，藤川和利助手，岡山聖彦助手，岡山理科大学理学部応用数学科劉渤江助教授に心より感謝致します．

第2章における研究を行なうにあたり，ともに研究を進め御討論，御協力を頂いた松下電器産業株式会社 田川健二氏，株式会社オージス総研 八幡孝氏に感謝の意を表します．

第3章における研究を行なうにあたり，ともに研究を進め御協力頂いた日本電信電話株式会社 齋藤淳氏，YM ダンス教室 仲秋朗氏，新日鉄情報通信システム株式会社 澤村卓哉氏，中部電力株式会社 日比野哲也氏に感謝の意を表します．

第4章における研究を行なうにあたり，ともに研究を進め御協力頂いた西尾研究室大学院生 庄司加奈子氏，秋山豊和氏に感謝の意を表します．

また，第3章と第4章における研究を通じ，ともに研究を進め有益な御討論，御協力を頂いた大阪大学大学院工学研究科情報システム工学専攻 原隆浩助手に心より感謝の意を表します．

最後に，本研究に関して有益な御討論を頂いた西尾研究室の諸氏に厚く御礼申し上げます．

# 参考文献


[AB87] Atkinson, M.P. and Buneman, P.: "Types and Persistence in Database Programming Languages," *ACM Computing Surveys*, vol.19, no.2, pp.105–190, 1987.

[AH+97] 秋山豊和, 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "データベース移動を用いた分散データベースシステムの設計と実装," 情報処理学会第 55 回全国大会講演論文集 (3), 4AC-11, pp.3-469–3-470, 1997.

[BH+95] Budiarto, 春本 要, 塚本昌彦, 西尾章治郎: "移動体計算環境におけるトランザクション制御方式," 電子情報通信学会データ工学研究会報告, DE95-51, pp.9–16, 1995.

[BH+96] Budiarto, Harumoto, K., Tsukamoto, M., Nishio, S., and Takine, T.: "Replica Allocation Strategies in Mobile Computing Environment," *Proc. International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC'96)*, Volume I, pp.99–102, 1996.

[BH+96b] Budiarto, Harumoto, K., Tsukamoto, M., and Nishio, S.: "On Transaction Management for Mobile Clients and Servers," *Proc. International Workshop on Multi-Dimensional Mobile Communications (MDMC'96)*, pp.402–406, 1996.

[BH+97a] Budiarto, Harumoto, K., Tsukamoto, M., and Nishio, S.: "Position

Locking: Handling Location Dependent Queries in Mobile Computing Environment," *Proc. International Conference on Worldwide Computing and Its Applications (WWCA'97)*, pp.C-3-2-1–C-3-2-15, 1997.

[BH+97b] Budiarto, Harumoto, K., Tsukamoto, M., and Nishio, S.: "Relocation Decision Policies for Mobile Replicas," *Proc. International Symposium on Digital Media Information Base (DMIB'97)*, 1997 (to appear).

[BH+98] Budiarto, Harumoto, K., Tsukamoto, M., Nishio, S., and Takine, T.: "On Strategies for Allocating Replicas of Mobile Databases," IEICE Transactions on Information and Systems, 1998 (to appear).

[BHG87] Bernstein, P.A., Hadzilacos, V., and Goodman, N.: Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.

[BK91] Barghouti, N.S. and Kaiser, G.E.: "Concurrency Control in Advanced Database Applications," *ACM Computing Survey*, vol.23, no.3, pp.269–317, 1991.

[BKK85] Bancilhon, F., Kim, W., and Korth, H.F.: "A Model of CAD Transactions," *Proc. 11th International Conference on Very Large Data Bases*, pp.25–33, 1985.

[BLW93] Banerjee, S., Li, V.O.K., and Wang, C.: "Distributed Database Systems in High-speed Wide-area Networks," *IEEE Journal on Selected Areas in Communications*, vol.11, no.4, pp.617–630, 1993.

[BO96] Buneman, P. and Ohori, A.: "Polymorphism and Type Inference in Database Programming," *ACM Transactions on Database Systems*, vol.21, no.1, pp.30–76, 1996.

[Date97] Date, C.J. 著, 藤原 譲 監訳: データベースシステム概論原書 6 版, 丸善, 1997.

[Devi93] Devine, R.: "Design and Implementation of DDH: A Distributed Dynamic Hashing Algorithm," *Proc. 4th International Conference on Foundations of Data Organization and Algorithms*, pp.101–114, 1993.

[EG+76] Eswaran, K.P., Gray, J.N., Lorie, R.A., and Traiger, I.L.: "The Notions of Consistency and Predicate Locks in a Database System," *Communications of the ACM*, vol.19, no.11, pp.624–633, 1976.

[Gray78] Gray, J.N.: "Notes on Data Base Operating Systems," Lecture Notes in Computer Science, no.60, pp.394–481, Springer Verlag, 1978.

[Gray81] Gray, J.N.: "The Transaction Concept: Virtues and Limitations," *Proc. 7th International Conference on Very Large Data Bases*, pp.144–154, 1981.

[HG+87] Herman, G., Gopal, G., Lee, K., and Weinrib, A.: "The Datacycle Architecture for Very High Throughput Database Systems," *Proc. 1987 ACM SIGMOD International Conference on Management of Data*, pp.97–103, 1987.

[HH+95] 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "ASN.1 データベースシステムにおけるインデックス機構の比較," 情報処理学会データベースシステム研究会報告, 95-DBS-104, pp.17–24, 1995.

[HH+96a] 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "ATM 環境におけるデータベース移動に基づくトランザクション処理手法," 情報処理学会データベースシステム研究会報告, 96-DBS-106, pp.49–56, 1996.

[HH+96b] 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "ATM 環境内で移動するデータベースの位置管理について," 情報処理学会データベースシステム

研究会報告, 96-DBS-109, pp.111–116, 1996.

[HH+96c] 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “データベース移動を用いた分散データベースシステムの並行処理制御について,” 情報処理学会マルチメディア通信と分散処理ワークショップ論文集, pp.179–186, 1996.

[HH+97a] Hara, T., Harumoto, K., Tsukamoto, M., and Nishio, S.: “Database Migration for Transaction Processing in ATM Networks,” *Proc. 11th International Conference on Information Networking (ICOIN-11)*, vol.1, pp.1B-4.1–1B-4.10, 1997.

[HH+97b] Hara, T., Harumoto, K., Tsukamoto, M., and Nishio, S.: “Location Management Methods of Migratory Data Resources in ATM Networks,” *Proc. ACM Symposium on Applied Computing (ACM SAC'97)*, pp.123–130, 1997.

[HH+97c] 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “ATM ネットワークにおけるデータベース移動のためのデータベース位置管理手法,” 電子情報通信学会論文誌 D-I, vol.J80-D-I, no.2, pp.137–145, 1997.

[HH+97d] 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “データベース移動に基づく分散データベースシステム DB-MAN の性能評価,” 情報処理学会第 54 回全国大会講演論文集 (3), 1R-8, pp.3-243–3-244, 1997.

[HH+97e] 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “データベース移動を用いた ATM ネットワークにおけるトランザクション処理,” 電子情報通信学会論文誌 D-I, vol.J80-D-I, no.6, pp.505–513, 1997.

[HH+97f] 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: “データベース移動に基づく動的複製配置法,” 電子情報通信学会データ工学研究会報告, DE97-23, pp.107–112, 1997.

[HH+97g] Hara, T., Harumoto, K., Tsukamoto, M., Nishio, S., and Okui, J.: "Main Memory Database for Supporting Database Migration," *Proc. 1997 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'97)*, Volume 1, pp.231–234, 1997.

[HH+98] Hara, T., Harumoto, K., Tsukamoto, M., and Nishio, S.: "DB-MAN: A Distributed Database System based on Database Migration in ATM Networks," *Proc. 14th International Conference on Data Engineering (ICDE'98)*, 1998 (to appear).

[HN+94] 春本 要, 仲秋 朗, 塚本昌彦, 西尾章治郎, 宮原秀夫: "抽象構文記法 1 (ASN.1) に基づくデータベースシステム," 情報処理学会第 97 回データベースシステム研究会報告 (94-DBS-97), pp.41–50, 1994.

[HSO94] Hart, K.W., Searls, D.B., and Overton, G.C.: "SORTEZ: a Relational Translator for NCBI's ASN.1 Database," Computer applications in the biosciences (CABIOS), vol.10, no.4, pp.369–378, 1994.

[HTN94a] 春本 要, 塚本昌彦, 西尾章治郎: "ASN.1 データベースプログラミング言語," 情報処理学会データベースシステム研究会報告, 94-DBS-99, pp.249–256, 1994.

[HTN94b] Harumoto, K., Tsukamoto, M., and Nishio, S.: "A Database System Based on ASN.1: Its System Architecture and Database Programming Language," *Proc. International Symposium on Advanced Database Technologies and Their Integration (ADTI'94)*, pp.160–167, 1994.

[HTN96a] 春本 要, 塚本昌彦, 西尾章治郎: "OODBMS を用いた ASN.1 データベースの実現とその評価," 電子情報通信学会論文誌 D-I, vol.J79-D-I, no.11, pp.975–983, 1996.

[HTN96b] Harumoto, K., Tsukamoto, M., and Nishio, S.: "Design and Implementation of an Information Repository for ASN.1-based Distributed Applications," *Proc. International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC'96)*, Volume II, pp.681–684, 1996.

[HYN96] 春本 要, 八幡 孝, 西尾章治郎: "協調作業支援のためのデータ管理モデル," 電子情報通信学会論文誌 D-I, vol.J79-D-I, no.5, pp.271–279, 1996.

[IM94] 市村 哲, 松下 温: "データベースとハイパメディア," bit 別冊「知的触発に向かう情報社会–グループウェア維新」, pp.159–179, 1994.

[Ishi91] 石井 裕: "コンピュータを用いたグループワーク支援の研究動向," コンピュータソフトウェア, ソフトウェア科学会, vol.8, no.3, pp.14–26, 1991.

[ISO87a] ISO 8824: Information Processing Systems — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1), 1987.

[ISO87b] ISO 8825: Information Processing Systems — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), 1987.

[ISO88] ISO/IEC 9594 1-8: Information Processing Systems — Open Systems Interconnection — The Directory, 1988.

[ISO91] ISO/IEC 9834: Information Technology — Open Systems Interconnection — Procedure, for the Operation of OSI Registration Authorities, 1991.

[JK93] Johnson, T. and Krishna, P.: "Lazy Updates for Distributed Search Structure," *Proc. 1993 ACM SIGMOD International Conference on Management of Data*, pp.337–346, 1993.

[Kato91] 加藤和彦: "オブジェクト指向データベースシステムの記憶構造," 情報処理, vol.32, no.5, pp.532–539, 1991.

[KKB88] Korth, H., Kim, W., and Bancilhon, F.: "On Long-Duration CAD Transactions," Information Sciences, no.46, pp.73–107, 1988.

[KL+84] Kim, W., Lorie, R., McNabb, D., and Plouffe, W.: "A Transaction Mechanism for Engineering Design Databases," *Proc. 10th International Conference on Very Large Data Bases*, pp.355–362, 1984.

[KS86] Korth, H.F. and Silberschatz, A.: Database System Concepts, McGraw-Hill, 1986.

[KS88] Korth, H.F. and Speegle, G.D.: "Formal Model of Correctness without Serializability," *Proc. 1988 ACM International Conference on Management of Data*, pp.379–386, 1988.

[LNS93] Litwin, W., Neimat, M.A., and Schneider, D.A.: "LH* – Linear Hashing for Distributed Files," *Proc. 1993 ACM SIGMOD International Conference on Management of Data*, pp.327–336, 1993.

[Lync83] Lynch, N.A.: "Multilevel Atomicity — A New Correctness Criterion for Database Concurrency Control," *ACM Transactions on Database Systems*, vol.8, no.4, pp.484–502, 1983.

[Masu93a] 増永良文: "マルチメディア文書の標準データベース格納構造の考察," 情報処理学会第 93 回データベースシステム研究会報告, pp.83–92, 1993.

[Masu93b] 増永良文: "ODA に基づくマルチメディア文書データベースの格納方式," 情報処理学会第 47 回全国大会講演論文集 (4), pp.4-97–4-98, 1993.

[Moss85] Moss, J.E.: Nested Transactions: an Approach to Reliable Distributed Computing, MIT Press, 1985.

[NH+93] 西山 智, 堀内浩規, 横田英俊, 小花貞夫, 鈴木健二: "OSI 管理情報ベース (MIB) 用データベースの設計と実装," 情報処理学会データベースシステム研究会報告, 93-DBS-95, pp.59–66, 1993.

[NH+94] 仲秋 朗, 春本 要, 齋藤 淳, 塚本昌彦, 西尾章治郎: "OODBMS を用いた ASN.1 データベースの実現," 情報処理学会マルチメディア通信と分散処理研究会報告, 94-DPS-65, pp.151–156, 1994.

[Nish91] 西尾章治郎: "オブジェクト指向データベースシステムにおける並行処理制御," 情報処理, vol.32, no.6, pp.540–549, 1991.

[NK+91] 中川路哲男, 勝山光太郎, 宮内直人, 玉田純, 水野忠則: "OSI ディレクトリシステムにおける DIB (ディレクトリ情報ベース) のオブジェクト指向アプローチによる実現," 情報処理学会論文誌, vol.32, no.3, pp.304–313, 1991.

[NM+90] 中川路哲男, 宮内直人, 勝山光太郎, 水野忠則: "OSI 抽象構文記法支援ソフトウェア APRICOT の開発と評価," 電子情報通信学会論文誌 D-I, vol.J73-D-I, no.2, pp.225–234, 1990.

[NO+93] 西山 智, 小花貞夫, 堀内浩規, 鈴木健二: "拡張可能 DBMS 構築技法に基づく高速 OSI ディレクトリ用 DBMS の設計と評価," 情報処理学会論文誌, vol.34, no.6, pp.1486–1496, 1993.

[NT+95] 西尾章治郎, 塚本昌彦, 春本 要, 下條真司: "広帯域ネットワークにおけるデータベースシステムの可能性," 電子情報通信学会データ工学

研究会報告, DE95-56, pp.49–56, 1995.

[NY+94] 西山 智, 横田英俊, 小花貞夫, 鈴木健二: "OSI ディレクトリ情報ベース (DIB) のためのハッシュを用いた高速名前解析処理方式," 情報処理学会論文誌, vol.35, no.12, pp.2762–2773, 1994.

[NZ90] Nodine, M.H. and Zdonik, S.B.: "Cooperative Transaction Hierarchies: A Transaction Model to Support Design Applications," *Proc. 16th International Conference on Very Large Data Bases*, pp.83–94, 1990.

[OBB89] Ohori, A., Buneman, P., and Breazu-Tannen, V.: "Database Programming in Machiavelli: a Polymorphic Language with Static Type Inference," *Proc. 1989 ACM SIGMOD International Conference on Management of Data*, pp.46–57, 1989.

[OY92] 大谷哲夫, 山本喜一, "協同文書における柔軟なアクセス制御機構," *Proc. Software Symposium'92*, pp.C-21–C-27, 1992.

[Papa86] Papadimitriou, C.: The Theory of Database Concurrency Control, Computer Science Press, 1986.

[Paul91] Paulson, L.C.: ML for the Working Programmer, Cambridge University Press, 1991.

[PK92] Prakash, A. and Knister, M.J.: "Undoing Actions in Collaborative Work," *Proc. 1992 ACM Conference on Computer-Supported Cooperative Work*, pp.273–280, 1992.

[PKH88] Pu, C., Kaiser, G.E., and Hutchinson, N.: "Split-transactions for Open-ended Activities," *Proc. 14th International Conference on Very Large Data Bases*, pp.26–37, 1988.

[Samp93] Sample, M.: Snacc 1.1: A High Performance ASN.1 to C/C++ Compiler, *snacc reference manual*, 1993.

[SA+94] Stonebraker, M., Aoki, P.M., Devine, R., Litwin, W., and Olson, M.: "Mariposa: A New Architecture for distributed data," *Proc. 10th International Conference on Data Engineering*, pp.54–65, 1994.

[Stal95] Stallings, W.: ISDN and Broadband ISDN with Frame Relay and ATM, Third Edition, Prentice-Hall International, 1985.

[SH+95a] 齋藤 淳, 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "ASN.1 データベースシステムにおけるデータ格納方式," 情報処理学会データベースシステム研究会報告, 95-DBS-101, pp.17–24, 1995.

[SH+95b] 澤村卓哉, 日比野哲也, 春本 要, 塚本昌彦, 西尾章治郎: "通信アプリケーション記述のための ASN.1 データベースプログラミング言語の拡張," 電子情報通信学会第 6 回データ工学ワークショップ (DEWS'95) 論文集, pp.175–182, 1995.

[SH+96] 庄司加奈子, 原 隆浩, 春本 要, 塚本昌彦, 西尾章治郎: "広帯域ネットワークにおけるデータベース移動の実現について," 電子情報通信学会データ工学研究会報告, DE96-10, pp.55–60, 1996.

[SPW90] Severance, C., Pramanik, S., and Wolberg, P.: "Distributed Linear Hashing and Parallel Projection in Main Memory Databases," *Proc. 16th International Conference on Very Large Data Bases*, pp.674–682, 1990.

[Sun87] Sun Microsystems, Inc.: XDR: External Data Representation Standard, RFC 1014, 1987.

[Sun88] Sun Microsystems, Inc.: RPC: Remote Procedure Call Protocol Specification Version 2, RFC 1057, 1988.

[Ullm88] Ullman, J.D.: Principles of Database and Knowledge-Base Systems, Volume I, Computer Science Press, 1988.

[Ullm89] Ullman, J.D.: Principles of Database and Knowledge-Base Systems, Volume II, Computer Science Press, 1989.

[VBW94] Vingralek, R., Breitbart, Y., and Weikum, G.: "Distributed File Organization with Scalable Cost/Performance," *Proc. 1994 ACM SIGMOD International Conference on Management of Data*, pp.253–264, 1994.

[Weik91] Weikum, G.: "Principles and Realization Strategies of Multilevel Transaction Management," *ACM Transactions on Database Systems*, vol.16, no.1, pp.132–180, 1991.

[XFS94] 徐 海燕, 古川哲也, 史 一華: "一貫性情報を用いたデータベースの並行処理制御," 情報処理学会論文誌, vol.35, no.12, pp.2752–2761, 1994.

[YF92] 依田育生, 藤井伸朗: "伝送網オペレーションにおける管理情報ベース (MIB) の構成法," 電子情報通信学会論文誌 B-I, vol.J75-B-I, no.8, pp.517–527, 1992.

[YHN94] 八幡 孝, 春本 要, 西尾章治郎: "汎用協調作業支援システム COCOA の設計およびその応用," 日本ソフトウェア科学会第 11 回大会論文集, C5-2, pp.209–212, 1994.