高速で高機能な全文検索が可能なデータベースカーネル

Database Kernel Enabling High Speed and Advanced Fulltext Search

池田　哲也*　　竹川　弘志*　　平岡　卓也*
Tetsuya IKEDA　　Hiroshi TAKEGAWA　　Takuya HIRAOKA

要　　旨

WindowsとLinux上で動作する高速な全文検索機能を備えたデータベースカーネルを作成した．キーワードによるブーリアンおよびランキング検索に加えて，自然文による検索や関連語による検索語の拡張といった高度な全文検索機能も提供する．これらはソフ研本製の全文検索サーバーが提供している機能と同等であるが，登録操作の性能は全文検索サーバーを大きく上回り，検索操作の性能もほぼ同等である．

また，索引再構成やバックアップはシステムの利用を止めずに行うことができ，電源障害などの事態にも再起動時に自動的に整合性を回復する機能を備えているため，メンテナンスのために生じる運用停止時間を軽減することができる．

インターフェースはJDBC(Java)または独自のクライアントライブラリー(C++)で，国際標準のSQLの一部に全文検索と運用管理用の拡張を加えた構文で使用できる．

ABSTRACT

A database kernel enabling high speed fulltext search is proposed. The database kernel works on both Windows and Linux and provides boolean and ranking search from keywords. Advanced fulltext search such as free text search using natural language and expanded query terms using related words are supported as well.

The fulltext search features are almost equal to those of the Fulltext Server, which is also a product of the Software R&D Group. The database kernel achieves faster registering operations and achieves almost the same search performance. Additionally, index files can be created and backup data can be executed without suspending system operations. When the system terminates unsafely in such a case as a sudden shutdown of the power source, the database kernel automatically recovers data by restarting the system.

These features reduce unavailable time resulting from maintenance of the system using the database kernel. The database kernel can be accessed through JDBC API (Java) or the original client library (C++) using a part of SQL. SQL is a global standard query language, expanded for fulltext search and maintenance operations.

* ソフトウェア研究開発本部　ユビキタスソリューション研究所
Ubiquitous Solution Lab, Software R&D Group

1. 背景と目的

文書を電子文書として保管することが当たり前になるにしたがって，大量の電子文書から求める文書を素早く発見することが重要になっている．特に扱われる電子文書の拡大により，文書構造が決まっていて書誌事項による検索で十分な定型文書だけでなく，文書構造の定まらない不定形文書も格納されるようになり，全文検索の重要性は高くなっている．

全文検索とは文書検索のうち文書の本文自体を対象とするものであり，全文検索システムは性能（検索速度，検索精度），使いやすさ，運用管理のしやすさの観点で主な技術改良がなされてきた[1]．これらはそのまま文書管理システムの要件ともいえるものである．

われわれのこの分野の研究成果は速度と精度の両面で優れた性能を持つ全文検索サーバー（Full Text Search，以下FTS）として実装されている．FTSでは使いやすさの向上のために自然文による検索や検索結果文書のフィードバックによるランキングの調整の機能が追加されている[2]．さらにわれわれは運用管理のしやすさを向上させるためにFTSで実証された技術をもとにしたデータベースカーネル（Database Kernel，以下DBK）を開発している．

DBKは文書管理システムに用いられることを第一の目的としている．そのためには，検索のみならず登録も高速な全文検索機能，バックアップなどによる運用停止時間の低減，文書管理システム全体のツールでの一元管理といった特長が要求されると考えている．

DBKはFTSに実装されていた高性能，高機能の全文検索機能に加えて，属性検索と組み合わせるための基本的な問い合わせにも対応しているため，他の汎用データベースを用いなくても単独で文書管理システムが実装できる．さらに，データベースの取り外しなどの運用管理操作も通常のデータ操作と同じインターフェースで行うことができるため，文書管理システムのツールで一元管理することができる．

DBKはさらに運用停止時間の低減を図っている．運用停止時間とはここでは文書管理システムを計画的または非計画的に停止させる時間をいう．計画的な停止にはバックアップなどが挙げられ，非計画的な停止には停電などによるサーバーダウンと電源復帰後の復旧処理が挙げられる．ユビキタス環境の発展によりシステムへのアクセスが空間的のみならず，時間的にも多様化するにしたがい，これらのいわゆるメンテナンスによる停止時間は無視できない障害になりつつある．

DBKではトランザクション（データベースに対する不可分な一連の処理）ごとにデータベースの仮想的な複写を見せるマルチバージョン同時実行制御という方式により，運用を止めないまま構成ファイルのコピーが可能なオンラインバックアップや突然のサーバーダウン後の再起動で自動的にデータベースの整合性を回復する自動リカバリーにより，これら運用停止時間の低減を図っている．

本稿ではFTSにも用いられている性能向上技術に触れた後にDBKにおいて改良した点を述べる．さらにその後マルチバージョン同時実行制御による運用停止時間の低減について解説する．

2. 技術

2-1 全文索引

全文検索には大きく分けて2種類の方法がある．ひとつは全文検索エンジンと呼ばれる単機能のソフトウェアを用いる方法であり[3) 4) 5) 6) 7)]，もうひとつは全文検索機能のあるデータベースカーネルを用いる方法である[8) 9) 10)]．

全文検索エンジンは一般に文書の全文を対象とした検索に特化したソフトウェアで，全文に対する条件を与えるとそれに合致する文書の識別子（全文検索エンジンが付与するIDまたは与えられたURLなど）または文書データそのものを返すものである．文書管理システムで用いる場合は全文以外による検索も必要であるから汎用のデータベースカーネルと組み合わせることになる．FTSも全文検索エンジンである．

全文検索エンジンによる方法は機能が単純で簡単に導入できるため，WWWや特許のようにすでに存在する大量の文書に対して全文検索を行うような用途によく利用される．また，文書データとのインターフェースさえあれば実体が格納される場所は制限しないので，グループウェア，データベース，ローカルディスク上のファイルといった複数のデータソースを横断的に検索する用途にも使用される[11)]．一方，一般には登録に時間がかかることやトランザクションの概念がないことから，更新操作の結果は定期的にまとめて反映する

ものが多く，更新操作の結果を即座に検索に反映することができない．また，属性に対する条件を含んだ検索が必要な場合はデータベースカーネルと組み合わせる必要があり，オーバーヘッドになる．

データベースカーネル自体が全文検索機能を持つ場合，全文はデータの属性のひとつとして扱われ，その属性に対して全文検索ができる索引が定義されることになる．索引の特長はデータベースカーネルごとに異なる．たとえばMicrosoft SQL Server[8] では単語間の距離も含んだ検索（近傍検索），フレーズの単語への自動分解（フレーズ検索）といった機能がある．IBM DB2[9] やOracle[10] にはさらに英単語の語尾変化のような表記の揺れに合わせた検索（あいまい検索），自然言語の文や文章での検索（自然文検索），シソーラスによる検索語の展開といった機能がある．

文書管理システムの実装に全文検索機能を持つデータベースカーネルを利用すると，全文検索エンジンのように文書の識別子との整合性をアプリケーションが意識する必要がなく実装しやすくなる，オーバーヘッドの減少により高速になる，管理するサブシステムがひとつ減る分運用管理が楽になるといった利点がある．また，更新された全文データを即座に索引に反映することが可能になる*．一方，高度な全文検索機能を備えたデータベースカーネルは一般に高価であり，またそのデータベースカーネル以外のデータソースを検索対象にするためには索引のついた表に全文データをコピーしなければならない点が不利である．

DBKはリレーショナルデータベースであり，全文検索機能を持つデータベースカーネルのひとつといえる．DBKはFTSで実装されていた近傍検索，あいまい検索，自然文検索，検索結果のフィードバックによる検索語の拡張[2] といった機能を同じく備えている．検索結果のフィードバックによる検索語の拡張は，検索漏れを減少させ検索精度を向上させる技術である．利用者から与えられた条件による検索の結果から利用者の要求に合致する文書（適合文書）を選び，適合文書に含まれる語から関連語を取り出して検索条件に追加する．この方法の利点は関連語辞書を別途用意する必要がないことと，利用者が検索語の重み付けをコントロールすることにより適合度の高い文書が上位に来やすいことが挙げられる．

* 上で例示したデータベースカーネルはいずれも更新されたデータの索引への反映タイミングをトランザクションの確定時，定期的，更新後即時から選べるようになっている．

2-1-1 遅延更新

FTSやDBKの全文索引は単語または連続したn文字の組を索引単位とし，索引単位ごとの出現文書と文書中の出現位置を記録した構造（転置ファイル）により実現される．転置ファイルに対する更新操作（登録や削除を含む）は，全文データを分解して得られるすべての索引単位に対して位置情報の追記や削除を行うことになるため，通常は文書数が増えるにつれて処理時間が増大する（Fig.1）．この問題の解決のために遅延更新という技術が用いられている[12]．以下で概略を説明する．

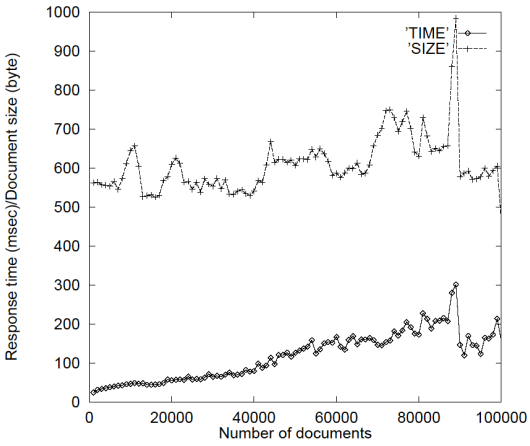

Fig.1 Relationship between the number of documents and insertion time.（参考文献12より引用）

転置ファイルにおいて文書数が増えるほど更新操作の処理時間が増大するという問題の解決には，登録が高速な索引を登録用として別に用意する方法が用いられる．登録用索引の内容は一時的なものなので何らかのタイミングで主要な転置ファイル（以下，主転置ファイル）に反映する必要がある．

FTSやDBKではこの登録用索引として主転置ファイルとまったく同じ構造の転置ファイルを，サイズが一定量を超えないようにすることで登録が高速な索引として用いることにした．これには登録直後から検索できるなどの利点がある．

登録用転置ファイルの内容を主転置ファイルに反映する操作（以下，マージ処理）は一定時間を経過するごとにあるいは登録用転置ファイルのサイズが一定の大きさになったときにバックグラウンドで実行される．マージ処理中にも登録処理を受け付けることができるように登録用転置ファイルを

複数個用意し，マージ処理の対象になっていない転置ファイルを登録用として使用するようになっている．Fig.2に登録用転置ファイルが2つのときの動作を示す．

削除操作についても同様に削除用転置ファイルを用意し，削除操作の代わりに削除用転置ファイルへの挿入が実施される．登録された後に同じ文書に対する削除が行われる場合，登録用転置ファイルにその文書のデータがあれば削除用転置ファイルへ挿入するのではなく登録用転置ファイルからの登録を行う．したがって，削除操作には登録用転置ファイルへの検索操作が必要であり，その分登録操作よりも削除操作のほうが時間がかかることになる．

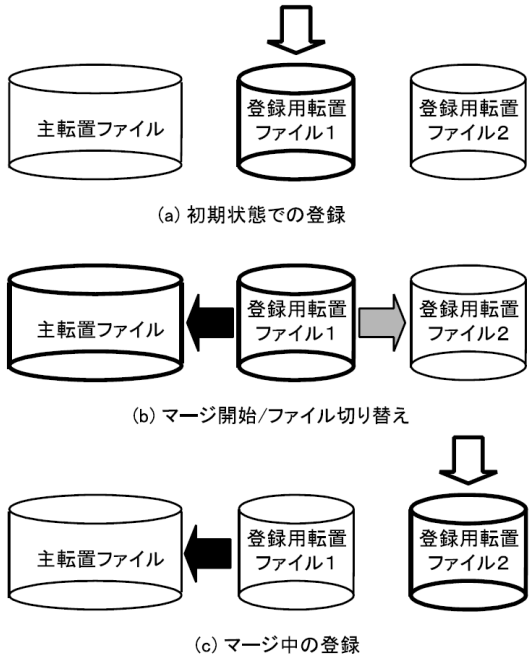

Fig.2　Switching mini-inverted files for insertion.
（参考文献12より引用）

遅延更新により全文索引への登録時間はFig.3に示すように大きく改善される．登録や削除が登録済み件数によらず高速になることで，他のデータベースカーネルのように全文索引へのデータの登録タイミングを選択する必要がなく常に即時登録ができ，更新操作の結果をすぐに検索対象にすることができる．

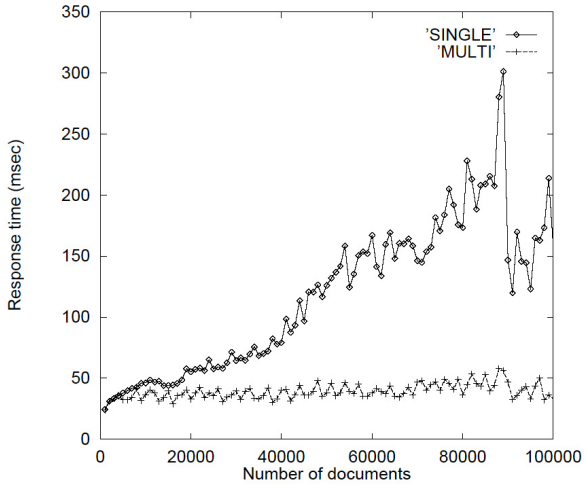

Fig.3　Insertion time by single inverted file and multiple inverted files.（参考文献12より引用）

2-1-2　バッチ登録

索引はすでにデータが登録されているリレーションに対して定義することができる．このような場合，連続して多数の全文データを登録することおよびリレーション中の全データが登録されるまで検索要求を処理しないことを用いて特殊な登録方法（バッチ登録）が可能である．

通常の登録操作は，文書がひとつ登録されるたびに検索可能な形になっている必要があるため，全文データから切り出される索引単位ごとに転置ファイルに対して位置情報リストの修正を行う．

これに対してバッチ登録では，すべての登録が完了するまで公開する必要がないため，メモリ上に索引単位と位置情報リストの対応関係を保持しておき，そのサイズが一定数を超えるごとに転置ファイルへの反映を行う．このようにすることでファイルへのアクセスを減少し，高速な索引へのデータ登録が実現できる．

バッチ登録は特にFTSを含む全文検索エンジンで使用される方式であるが，バッチ登録中は検索操作が行えないため，その間のサービス受付を停止するか，同じシステムをもう1セット用意して一方がバッチ登録中の場合に他方がサービスの受付を継続し，バッチ登録終了後に切り替えるといった手法が用いられる．後者の方法でサービスの受付を継続した場合でも，バッチ登録中に更新されたデータが検索に反映されるのは次のバッチ登録を待たなければならない．それに対し，DBKでは索引へのデータ登録中であっても更新操作を含む通

常のデータ操作が行え，データ登録終了時には登録中に行われた更新操作の結果も反映された索引が作成される．この機能は先に挙げた商用のデータベースカーネルでも実現されている機能で，本稿ではこれをオンライン索引作成と呼ぶ．オンライン索引作成により，たとえばリレーション本体にデータを登録しただけの状態で運用を開始し，バックグラウンドで全文索引を作成するようなことが可能になる．

データベースに対して行った操作内容は，論理ログと呼ばれるデータベースとは別のファイルへ記録される．オンライン索引作成は，論理ログに記録されている更新操作の内容を後から索引に反映することにより実現される．

2-1-3　その他の改良

DBKにおける性能改善のためのその他の改良点について述べる．

まず，物理的なファイルへの書き込み回数の減少が挙げられる．FTSではトランザクションが確定するごとに変更内容を物理的なファイルへ書き込んでいたのに対し，DBKでは可能な限りメモリ上のバッファに保持しておき，バックグラウンドで定期的に物理的なファイルへの書き込みを実施している．これにより複数の更新操作の結果をまとめて書き込むことができるため書き込む回数が減少し，さらにファイルに対するランダムアクセス（一般に比較的低速な操作である）よりもシーケンシャルアクセス（一般に比較的高速な操作である）が増えるので高速になる．

また，転置ファイル自体の登録，検索性能を上げることも性能改善には有効である．DBKではFTSにおける転置ファイルのフォーマットを分析し，位置情報を格納する方式を工夫することで1回の登録または検索操作で読み込みが必要になるデータ量をほぼ半減させることに成功した．

読み込むデータ量が半減することは，メモリ上に同じサイズのキャッシュを用意していても効率がほぼ倍になる計算になるのでファイルの読み込み回数が減り，結果として高速になる．

2-2　マルチバージョン同時実行制御

複数のトランザクションを同時に実行したときに，データベースの一貫性を保証するためには，一連のトランザクションの実行が直列可能であることを保証しなければならない．直列可能性（serializability）を保証するトランザクションの同時実行制御の方法はいろいろなものが提案されている[14]．それらの目標の一つとして，隔離性（isolation）と並行実行性能の両立がある．一般的にトランザクション間の隔離性を上げると，トランザクションの並行実行性能は下がる．

トランザクションは，データベースを更新しない読取用トランザクションと，データベースを更新する更新トランザクションに分類できる．Microsoft SQL Server[8] やIBM DB2[9] などが使用する一般的な同時実行制御の方法である2相ロックを用いる場合，データベース中の操作対象を，参照時には読取(S)モード，更新時には更新(X)モードで，それぞれロックしなければならない．分離性を保証するために，読取モードと更新モードは競合する．例えば，読取用トランザクションと更新トランザクションが同じ対象を，それぞれ参照，更新するとき，後からロックを試みたトランザクションは先にロックしたトランザクションがロックをはずすまで，待機（ロック待ち）することになる．

DBKはトランザクションの同時実行制御の方法として，マルチバージョン同時実行制御（Multi-Version Concurrency Control，以後，MVCCと呼ぶ）を採用している．Oracle[10] やPostgreSQL[15] なども同様である．MVCCでは，一貫性が保証されているある時点でデータベースを複写し，読取用トランザクションは複写元の，更新トランザクションは更新先のデータベースをそれぞれ操作することで，読取用トランザクションと更新トランザクションの間の競合をなくすことができる．このある時点で複写されたデータベースのことを，データベースのそれぞれの時点での版（versionやsnapshot）と呼ぶ．読取用トランザクションは，最新でない版のいずれかひとつを参照する．更新トランザクションは，最新版のみを参照，更新する．そのため，ある版はその版を元にして新しい版が生成された時点で不変になり，読取用トランザクションに提供される隔離性は，ISO SQL[16] における最高のSERIALIZABLEとなる．

MVCCは版を記憶するために，2相ロックと比較してより多くの記憶領域が必要となる．版の生成時にデータベースを丸ごと複写したのでは，時間およびその記憶のためのコストがあまりに大きすぎる．そこで，ある版を更新するとき，その版の更新される部分のみ複写してから更新することで，ある時点の版をそれと直前の版の共通部分と，複写先である更

新された部分の和によって表現できる（Fig.4を参照のこと）．文書管理用などの追記型のデータベースの場合，データベース全体から見れば更新される部分はほんの一部分なので，かなり小さなコストで，ある時点の版を記憶することができる．この更新対象となる複写される部分を記録する単位が，版を構成する最小単位になる．

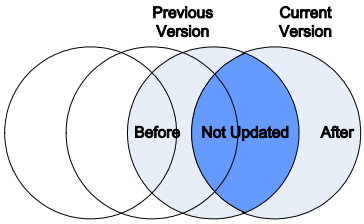

Fig.4 Relationship between previous version and current version.

DBKでは，データベースは複数の論理的なファイル（以後，論理ファイルと呼ぶ）から構成され，この論理ファイルは固定サイズのページに分割されており，版はこのページ単位に構成される．あるページを更新するとき，そのページの最新版をコピーし，新しい版とし，それを更新することになる．論理ファイルはオペレーティングシステムが提供するファイル（以後，OSファイルと呼ぶ）に格納される．論理ファイルのページと同様にOSファイルも固定サイズのブロックに分割され，それぞれのブロックはOSファイルの先頭からのオフセットで識別できる．あるページのひとつの版は同じサイズのひとつのブロックに格納される．それぞれの版には生成されたときの時刻と，その版の元となった版を格納するブロックのオフセットが記録される．つまり，論理ファイルの先頭からN番目のページの複数の版は，そのページの最新版を先頭とし，最古版を末尾とする線形リストで管理される．あるページの最新版のブロックを高速に探索するために，そのページが論理ファイルの先頭からN番目であることをキーにして，最新版を格納するオフセットを探す木（以後，探索木と呼ぶ）も論理ファイル中に版とは別に格納される．ある読取用トランザクションが参照すべき版は，この探索木と線形リストをたどりながら，そのトランザクションを開始した時刻より前に生成された版，かつ，開始した時点で実行中の更新トランザクションにより生成されていない版のうち，最新のものを探すことになる（Fig.5を参照のこと）．現在実行中および今後開始されるトランザクションから参照される可能性がなくなった版は削除される．

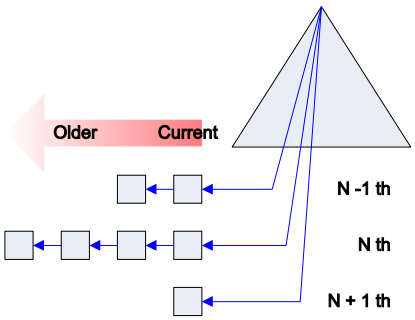

Fig.5 Search tree and singly linked list for seeking for versions.

版はトランザクションの同時実行制御だけでなく，電源断などのシステム障害からの回復に使用される．OSファイルの更新内容を表す新しい版は，処理の高速化のためにバッファリングされ，そのディスクへの書き込みは遅延される．そのため，システム障害が発生すると，バッファリング内容が失われ，データベースの一貫性も失われるかもしれない．DBKは不定期的にチェックポイント処理を行う．チェックポイント処理ではフラッシュされていないデータベースの更新内容をディスクへ書き込む．障害回復では，まず，探索木に記憶されている先頭の版のオフセットをシステム障害が発生する直前のチェックポイント時点の版のオフセットに更新する．次に，論理ログを使い，そのチェックポイント以降に確定（commit）されたトランザクションでの操作を再実行し，そのチェックポイント時に実行中のトランザクションのうち，確定されなかったものの操作を取りやめる．

また，DBKは版を使用することでOSファイルのコピーによるオンラインバックアップを実現する．つまりDBKのバックアップは，DBKを使用するサービスにまったく影響を与えずに，特別なツールでなく，ただOSファイルをコピーするツールにより，データベースを構成するOSファイルをコピーすることで行える．ある読取用トランザクションでバックアップの開始を指示すると，データベースを構成するOSファイルのフラッシュされていない版をフラッシュし，バックアップの終了が指示されるまで，前述の探索木が更新されても，OSファイルへのフラッシュは遅延される．そのため，バックアップされたデータベースはバックアップ開始時の探索木とバックアップ中に追加された版が入り混じったものとなる．リストアは，まず，コピーされたOSファイルを元に

戻してから，バックアップを行った読取用トランザクションが参照すべき版のオフセットで探索木に記憶されている先頭の版のオフセットを上書きすることにより可能である．

前述したように，OracleやPostgreSQLでもMVCCを提供する．OracleでMVCCをどのように実装しているかは不明である．PostgreSQLでは更新時に更新前のタプルをそのまま残しておき，更新後のタプルを新たに挿入し，参照時に適切なタプルを選択することにより，タプル単位の版を実装している．DBKのページ単位の版はタプル単位のものと比べてランダムアクセスになりやすいという欠点はあるが，DBKではPostgreSQLのようなオンラインバックアップのための特別なツールは必要とせず，OSが提供するコピーコマンドから市販されている高度なバックアップツールなどの様々なツールをそのまま使用してオンラインバックアップできる．

2-3　モジュール構成

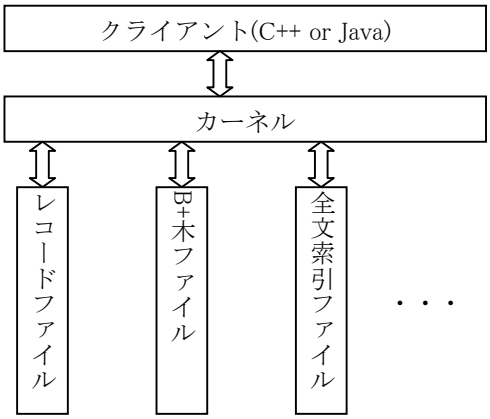

Fig.6　Module structure of DBK.

ここでDBKのモジュール構成に触れておく．DBKはカーネルモジュール，ファイルドライバー，クライアントモジュールにより構成される．ファイルドライバーは構造の異なる論理ファイルの種類ごとに用意されるもので，全文索引のためのファイルドライバー，リレーションの順次スキャンのためのレコードファイルドライバー，数値や短い文字列用索引のためのB+木ファイルドライバーなどがある．ファイルドライバーはオープン，読み書きといった共通インターフェースを備えており，必要に応じてカーネルモジュールに動的にロードされる．この仕組みにより新しい構造のファイルを組み込むことが比較的容易になっている．たとえば，全文索引のためのファイルドライバーは遅延更新のための複数の転置ファイルをひとまとめにして共通インターフェースが実装されており，カーネルモジュールからは遅延更新の仕組みは隠蔽されている．

クライアントモジュールはC++言語用とJava言語[17]用があり，Java言語用のクライアントモジュールはJDBCドライバーとなっている．

3. 成果

3-1　性能比較

全文索引の登録および検索性能をFTSとの比較で示す．測定にはDell Power Edge 750（CPU：Pentium4 3.40GHz，メモリ：1GB，OS：Windows Server 2003 Standard Edition）を用いた．

3-1-1　登録性能

Table 1は5万件のデータ（合計サイズ約900MB）が登録された全文索引が作成されている状態で，平均約20KBのデータを10件連続して挿入または削除したときの1件あたりの平均処理時間をFTSとDBKで比較したものである．

Table 1　Insertion and deletion time comparison between DBK and FTS.

	登録時間（秒）	削除時間（秒）
FTS	0.561	0.380
DBK	0.194	0.088
比	35%	23%

DBKの転置ファイルはFTSに比べてファイルサイズが約半分であるので，一度メモリ上に読み込んだページ中に必要な位置情報リストが含まれる確率が高く，その分ファイルからの読み込み回数が少ない．また，変更内容のファイルへの書き出しがチェックポイント処理まで遅延される．これらにより登録，削除ともに大幅な処理時間短縮が実現できた．

3-1-2　検索性能

Fig.7は登録性能と同じく5万件900MBの全文データが登録された全文索引に対して，ヒット件数の異なる104パターンの検索語でDBKと全文検索サーバーのそれぞれについて検索したときの応答時間の傾向を近似式で示したものである．

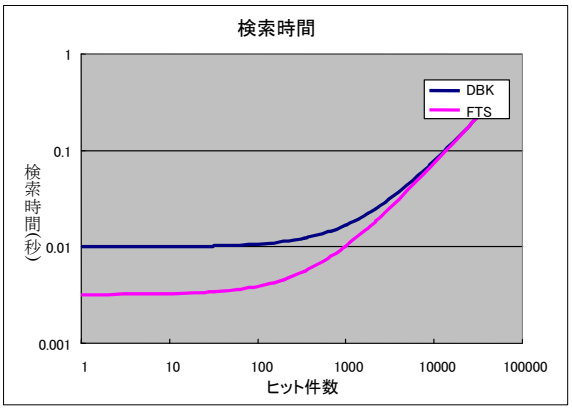

Fig.7　Searching time comparison between DBK and FTS.

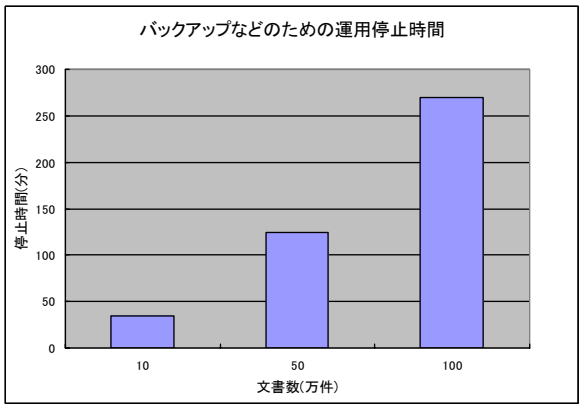

Fig.8　Service unavailable time caused by maintenance.

登録処理とは異なり，検索では必要な位置情報リストの数が少ないので，ファイルサイズの影響はほとんどない．一方，FTSに比べてDBKはデータベースカーネルとしてのインターフェースを通してやや汎用性の高いデータ型で結果を返すため，その分のオーバーヘッドがかかっている．これらの要因にもかかわらず，ほぼ同等の性能を実現できた．

3-2　運用停止時間の低減

運用停止時間の低減は，DBKの運用実績の期間が短いために直接の成果はまだない．しかし，FTSを用いたシステムでバックアップなどのために運用を停止した時間により間接的に効果を示すことはできる．

Fig.8はFTSを用いた情報検索システムにおいて規模の大小から典型的な例をとり，バックアップのために運用を停止した時間の実績を示したものである．この時間にはバックアップ前にデータベースとFTSの整合性を検査する時間も含まれている．DBKではどちらも運用を停止せずに実施できるので0にできる．

4. 今後の展開

DBKが提供する高速で高機能な全文検索は，文書を対象とするさまざまな製品に応用可能である．今後は性能面，機能面ともにさらなる向上を図りながら，応用先の探索を行う予定である．

参考文献

1) 全文検索システム協議会：「全文検索システムとは何か？」2003年版，(2003).
2) 真野博子，伊東秀夫，小川泰嗣：文書検索におけるランキング検索技術，リコーテクニカルレポート，No.29，(2003).
3) 全文検索システムNamazu，http://www.namazu.org/.
4) Verity Ultraseek，http://www.verity.com/products/ultraseek/.
5) 高速全文検索登録ソフトウェアPanaSearch，http://panasonic.biz/panasearch/.
6) MitakeSearch，http://www.hp.com/jp/mitake/.
7) 超高速日本語全文検索エンジンInfoBee3，http://bee.ntt-it.co.jp/.
8) Microsoft SQL Server Books Online，http://www.microsoft.com/japan/msdn/library/.
9) IBM DB2ユニバーサルデータベース8.1マニュアル，http://www.db2.jp/siryo/db2online/.
10) Oracle Database 10gマニュアル，http://otn.oracle.co.jp/.
11) ConceptBase Search 1000 Ver.4.0，http://www.justsystem.co.jp/km/product/cb101.html.
12) 小川泰嗣，山本研策，真野博子，伊東秀夫：全文検索システムのための複数転置ファイルを用いた登録高速化とランキング検

索，第13回データ工学ワークショップ(DEWS2002)，(2002).
13) フィリップ A バーンスタイン，エリック ニューカマー著，大磯 和広ほか訳：トランザクション処理システム入門，(1998).
14) ジムグレイ，アンドレアス ロイター著，喜連川優訳：トランザクション処理 概念と技法上，(2001).
15) PostgreSQL 7.4.3マニュアル，http://www.postgresql.jp/document/.
16) ISO/IEC 9075-2:2003 Information technology -- Database languages -- SQL -- Part 2：Foundation (SQL/Foundation)，(2003).
17) Java Technology，http://java.sun.com.

注1) Microsoft，SQL Server，WindowsはMicrosoft Corporationの商標です.
注2) DB2はIBM Corporationの商標です.
注3) OracleはORACLE Corporationの商標です.
注4) JavaおよびJDBCはSun Microsystems, Inc.の商標です.
注5) PowerEdgeはDELL Computer Corporationの商標です.
注6) PentiumはIntel Corporationの商標です.