解　説

データベース処理におけるベンチマーク†

喜連川　優††　伏見信也†††
田中　茂††††　丸山光行††††

1. はじめに

1970年代の後半より従来のファイルシステムからデータベース管理システムへと移行が進み，今日ではデータベース管理システムは計算機システムにおけるもっとも大きなソフトウェアシステムの一つといえる．一方で，現時点ではデータベース管理システムに対する標準のベンチマークはいまだ存在せず，その確立は急務といえる．本稿では，最近，標準データベースベンチマークの確立を目的として米国においてその活動が開始された TPC ベンチマーク，ならびに，データベース研究者に広く利用されつつあるウィスコンシンベンチマークについて解説する．

2. データベースベンチマーク

科学技術計算に対するベンチマークは数多く存在するのに対し，データベース処理に関するベンチマークはいまだ十分に確立されていない．ベンチマークの確立が遅れた原因は第一に本格的なデータベースの構築，利用が進んできたのは比較的最近であり，従来はそれほど性能に対する要求がなかったこと，第二に，科学技術計算用ベンチマークの場合は測定環境が比較的単純であるのに対し，データベースの場合はデータは主記憶ではなく通常ディスクに置かれることから，ディスク上のデータ編成法，チャネルやディスク装置などのシステム構成，さらにユーザの問い合わせ入力のための通信システムなど非常に多くの要因によってシステムの性能が決定され測定環境が複雑であることなどにあると考えられる．しかしながら，データベースの普及にともない，その処理性能に関する標準的測定手法の確立が望まれており，いくつかの試みが開始されている．

現行のデータベース処理は大きく二つに分類できる．一つは座席予約や銀行業務などにみられる定型的処理であり，これは比較的単純なレコードの変更・追加・削除などの処理を中心とするものであり，トランザクション処理と呼ばれる．他方は非定型的な処理で，たとえば，一月の売上データからの今後の人気商品の予想や，地域別の商品売上分析など，定型業務によって貯えられたデータに対していろいろな角度から非定型的な処理を施すものを指す．この非定型処理は従来の階層型やネットワーク型のデータベースでは困難であり，柔軟なアクセスパスを提供する関係データベースにおいて本格的に利用されるようになった．関係データベースに対するベンチマークとして，ウィスコンシンベンチマークについて **3.** で述べる．またトランザクション処理に対するベンチマークとして，TPC ベンチマークに関して **4.** でその動向を述べる．

3. 関係データベース処理に対するベンチマーク

3.1 歴史と背景

関係データベース処理に対するベンチマークの研究が活発化したのは比較的最近である．1980年代を振り返ると，P. Hawthorn, D. J. DeWitt によるデータベースマシンの解析的性能比較の研究[1),2)]を契機とし，続いてウィスコンシン大学のデータベースマシンの研究者たちがデータベースマシンの有効性を実証しようと市販のデータベースシステムとの比較評価のためのベンチマークを作成した[3)]．このベンチマークは，

- 使用されるデータベースが系統的かつ単純に設計されており，出力レコード数などを制御しやすい
- 定義されている問い合わせが単純で関係演算全体に対して網羅的である

† Benchmarks for Data Base Systems by Masaru KITSUREGAWA (Univ. of Tokyo, Institute of Industrial Science), Sinya FUSHIMI (Mitsubishi Electric. Co.), Shigeru TANAKA and Mitsuyuki MARUYAMA (Fujitsu Limited).

†† 東京大学生産技術研究所
††† 三菱電機(株)
†††† 富士通(株)

●可搬性（どのシステムでも比較的容易に実行可能である）がある

などの特長を有し，またそのほかに共通の性能評価の基盤たるベンチマークが存在しなかったことなどから，データベースマシンに限らず，一般の関係データベースシステムのベンチマークとしてデータベース研究者を中心に比較的普及が進んだ．このベンチマークはウィスコンシンベンチマークと呼ばれる．我が国でもデータベースシステムの性能評価に利用されている[15]．

3.2 ウィスコンシンベンチマーク[3),4)]

3.2.1 目　　的

ウィスコンシンベンチマーク[1)]は，1983年にウィスコンシン大学のデータベース研究グループによって発表された関係データベースシステムのベンチマークである．本ベンチマーク開発のもともとの動機は，1970年代に活発化したデータベース処理の高速化技術（特にデータベースマシン技術）に関する研究成果の実証であった．すなわち，

（1） 検索，並列処理などの特殊なハードウェア，及びアーキテクチャの有効性の検証

（2） データベース専用 OS の有効性の検証

（3） 問い合わせ処理の最適化アルゴリズムの評価

などが具体的目標であり，このために比較対象として一部商用システムが用いられた．

3.2.2 概　　要

本ベンチマークの内容及び特長は以下のようにまとめられる．

●〔基本前提〕本ベンチマークはシングルユーザベンチマークである．すなわち，複数ユーザにより同時に問い合わせが実行された場合の性能は対象としない．これは前記(1)，(2)などの目的のため，問題を複雑化する要因を排除した結果である．

●〔測定結果〕ベンチマーク結果としては問い合わせに対する処理の経過時間（応答時間）を報告する．処理に要した CPU 時間やディスク入出力回数についての測定を特に指定しない．

●〔問い合わせ〕少数の「典型的アプリケーション」をベンチマークとするのではなく，多数の基本演算を網羅的（21種，127項目の問い合わせ）に実行する．

●〔データベース〕データ長，データ型，データの順序などを細かく定義し，系統的に操作しやすい7つの関係をデータベースとして使用する．

●〔公表，監査など〕システムに対しての機能的要求事項（障害回復や同時実行制御の完成度など）や監査，報告事項など，公正なベンチマークとして備えるべき要件は定義されていない．すなわち，本ベンチマークは研究者や技術者によって良識に基づいて実施，解釈されることを前提としており，システムの単純な優劣比較を目的としていない．

ここで，ベンチマークとして実施する問い合わせの数が比較的多いことは注目に値する．トランザクション処理では，アプリケーションプログラムの構造，動作は比較的単純で，これらは大部分のアプリケーションプログラムに共通している．したがって，後述の TPC ベンチマークにみられるように，ある種の「典型的アプリケーション」をベンチマークとして定義し，これら少数のベンチマークにより比較的精度の高い予測データ，比較データを得ることができる．一方，関係データベース処理においては，データに対する非定型処理を目的とするがゆえに，「典型的問い合わせ」や「典型的負荷状態」を見出すことは困難である．このような背景からウィスコンシンベンチマークでは，複雑な関係データベースアプリケーション空間の基底ベクトルともいうべき基本演算を網羅的に実施することによってシステムの評価を試みているのである．

3.2.3 データベース

本ベンチマークでは複数の関係（テーブル，ファイル）が用いられるが，それらは以下のような共通のレコード形式をもつ．この形式により，レコード長は182バイトとなる．

unique1：0から始まり，当該関係レコードの数−1の範囲で乱数により生成された重複のない2バイトの整数値．

unique2：0から始まり，レコードごとに順に1増加された重複のない2バイトの整数値．

two, four, ten, twenty, hundred, thousand, twothous, fivthous, tenthous：

おのおの (0, 1), (0, 1, 2, 3), (0, 1, …, 9), (0, 1, …, 19) (0, 1, …, 99), …, (0, 1, …, 9999) の範囲の2バイトの乱数値．

even100, odd100：2バイト数であるが内容の定義は原論文に記述がない．

stringu1, stringu2, string4：

おのおの52バイトの文字列であり，先頭，26文字目，及び最後の文字がAからVの文字値を繰り返す．その他の文字はすべて値Xをもつ．stringu2 は文字列として昇順にソートされており，stringu1 はこれ

unique1	unique2	two	ten	hundred	thousand
378	0	1	3	13	615
816	1	0	4	4	695
673	2	0	6	26	962
910	3	0	2	52	313
180	4	0	0	20	74
879	5	1	9	29	447
557	6	1	7	47	847
916	7	0	4	54	249
73	8	0	6	26	455
101	9	0	2	26	657

図-1 データベースの例（一部属性省略）

がランダムに選ばれている．string4 は文字列値が下記の 4 種類に限られる．

"Axxx…xxxAxxx...xxxA"
"Hxxx...xxxHxxx...xxxH"
"Oxxx...xxxOxxx...xxxO"
"Vxxx...xxxVxxx...xxxV"

図-1 にこの形式により生成された関係の一部（属性も一部省略）を示す．この定義を用いて，レコード数 1000 個 (onektup)，2000 個 (twoktup)，5000 個 (fivektup)，10000 個 (tenktup1, tenktup2) などの関係が生成される．これら関係においては関係ごとの属性の識別のため属性の末尾に unique1A などのように関係を示すアルファベット記号が付加されている．さらに，場合によって，unique2 に対してクラスタ索引* またはハッシュによる直接アクセスパスが，また unique1 と hundred1 に対して非クラスタ索引が生成される．

属性によってソート，ランダムなどのデータの順序を定めているため，ディスクアクセスに関する種々のアクセスパターンを容易に実現することができる．たとえば属性 unique1 はランダムにレコードが入力されているためレコードがディスクの物理ページに分散して配置されることとなり，この属性を用いて索引により順アクセスした場合は入出力回数がほぼレコード数に比例したものとなる．逆に属性 unique2 はソート順にデータが入力，配置されるため，この属性を用いて同様な操作を行った場合は，入出力回数は大きく減少することになる．

```
range of t is tenKtup1
range of x is tenKtup2
    retrieve into skr101 (t. all) where t. unique2D<1000
    retrieve into skr102 (x. all) where x. unique2E>8999
    retrieve into skr103 (t. all) where (t. unique2D>791) and (t. unique2D<1792)
    retrieve into skr104 (x. all) where (x. unique2E>311) and (x. unique2E<1312)
    retrieve into skr105 (t. all) where (t. unique2D>902) and (t. unique2D<1903)
    retrieve into skr106 (x. all) where (x. unique2E>467) and (x. unique2E<1468)
    retrieve into skr107 (t. all) where (t. unique2D>887) and (t. unique2D<1888)
    retrieve into skr108 (x. all) where (x. unique2E<985) and (x. unique2E<1986)
    retrieve into skr109 (t. all) where (t. unique2D>534) and (t. unique2D<1535)
    retrieve into skr1010 (x. all) where (x. unique2E>647) and (x. unique2E<1648)
```

図-2 選択演算（10% 選択率）の問い合わせ

```
range of t is tenKtup1
range of w is tenKtup2
retrieve into tmpj1 (t. all, w. all)
where (t. unique2D=w. unique2E) and (w. unique2E<1000)
range of t is tenKtup2
range of w is tenKtup1
retrieve into tmpj2 (t. all, w. all)
where (t. unique2E=w. unique2D) and (w. unique2D<1000)
```

図-3 結合演算の問い合わせ

3.2.4 問い合わせ

ベンチマークとして定義されている問い合わせは大きく選択，結合，射影，集約演算，削除，更新，追加に分類される．このうち，選択及び結合については索引（クラスタ／非クラスタ）の有無や結果レコード数について細かく場合分けされた定義が行われている．

図-2 及び図-3 におのおの選択演算と結合演算に関する問い合わせの一部を示す．問い合わせは INGRES で用いられている問い合わせ言語 QUEL[11] で記述されている．

図-2 に示した選択演算の内容は，属性 unique2 に対して幅 1000 の選択範囲を指定することにより，10000 個のレコードから 1000 個（10%）のレコードを選択し，結果をファイル出力するものである．ここでは 10 通りのベンチマークが順に実行されるが，これらは二つの関係 tenktup1 と tenktup2 を交互に使用していることに注意が必要である．これにより各試行間での主記憶上のデータバッファ

* レコード群がある属性に関し，ソート順に 2 次記憶に格納されているとき，レコード群はこの属性に関してクラスタ化されている，と言う．これはレコードの論理順序と物理的な格納アドレスが一致していることに由来する．クラスタ化属性に対して与えられた索引をクラスタ索引と呼ぶ．逆に論理順序と格納順序が異なっている属性に対する索引は非クラスタ索引と呼ばれる．

のキャッシュ効果が減少し，すべての問い合わせの試行で必ずディスク入出力が実行されることが保証されて，アドホックな問い合わせの実行性能が測定されることになる．また，同様の演算をクラスタ索引／非クラスタ索引のある場合について実行することにより，システムのオプティマイザの実行方式（関係の全数探索と索引による選択の使い分けなど）の有効性を評価することができる．

図-3 に示した結合演算の内容は，一方の関係から属性 unique2 により 10% のレコードを選択し，その結果をもう一方の関係と属性 unique2 により結合して結果をファイルに出力するものである．選択演算の場合と同様に，索引を付加し，同様な問い合わせを実行することによってオプティマイザの評価を行うことができる．また，このベンチマークでは選択演算と結合演算に用いる属性が同一であるため，関係演算の分配則によって選択演算を両方の関係に対して実行することが可能である．これによりオプティマイザがこのような関係代数レベルの最適化を実施しているか否かが判定でき，またこの技法の有効性が確認される．結合演算に対するベンチマークとしては，このほかにあらかじめ選択を施した関係との単純結合や，三つの関係の結合などが定義されている．なお，文献3）は入手が容易でないので，付録に本ベンチマークの全定義を示す．

3.2.5 ウィスコンシンベンチマークの問題点

このようにウィスコンシンベンチマークでは OS のバッファ管理や索引機構，最適化アルゴリズムの動作条件に対して細かい配慮がなされ，高速データベースシステム／データベースマシン研究成果に対する実証評価基準となり得るものである．また，現実的観点からは，たとえば結合演算の処理コストが得られることによりデータベースの正規化による結合演算数増加の制限度合いや，対記憶コストにおいての索引作成の有効性に関する判断基準などが得られよう．しかし，一方で本ベンチマークは作成者自身が「データベースベンチマーク研究における第一ステップ」と言っている[3),4)]ように，ベンチマークとしては未完成であり，いくつかの重要な問題点も指摘されている．これらの代表的なものを以下にあげる．

（1） 各ベンチマークの意味や背景をよく理解した研究者，技術者によってベンチマーク結果を評価する場合にはベンチマーク結果から有益な結果が得られるが，エンドユーザに対してのベンチマークとしては完成度が低く，安易な数字の比較は危険である．特に第三者による監査など，結果の妥当性に対して配慮がないため，索引，データ編成，ディスクスペースの割り付け，メモリ容量，ディスク制御装置とディスクドライブの配置，OS の改版状態など，データベースシステムが仮定している環境の実現度合いが明確でないまま測定値が報告される可能性がある．たとえば参考文献 7) には IBM 4341/SQL/DS システムでの経験として，同じ問い合わせに対してデータの編成を変えると応答時間が 9 時間から数分に変化した例が報告されており，ベンダのシステムエンジニアの協力を得たシステムチューニングの有無が性能に大きく影響を与え得ることを警告している．

（2） 網羅的であるがゆえに問い合わせ数が多すぎてベンチマーク実施に多大の時間を要する．

（3） 逆にデータベースの実運用の観点からは，依然として網羅性が不足する．たとえば，データベースは定期的に再構成してアクセス性能を維持することが一般的であるが，これに要するデータベースのロード／アンロード，索引生成などの機能のベンチマーク項目がない．また結合，更新などに関する演算種類も不足している．

（4） シングルユーザベンチマークであり，マルチユーザ環境での性能値が得られない．特に，ディスクごとの入出力回数や処理に要した CPU 時間が報告されないため，多重処理時の性能低下に関しては本ベンチマーク結果からはなんら予測手段がない．

（5） 現時点ではデータベースが小さすぎ（最大の関係で 2 MB，データベースサイズは合計で 6 MB 程度），大型のマシンではすべての関係を主記憶上において処理することが可能である．「なるべく少ない資源で短時間に精度の高い評価データを得る」，というベンチマークの原則に従い，本ベンチマークではデータベースサイズを比較的小さいものとしていたのに対し，ハードウェア技術の進歩によりこの設定が現実性を失ってきている．

（6） ベンチマーク定義が不正確である．定義の不明確な属性（odd100，even100 など）があり，また測定対象となるシステムが備えるべき機能（障害回復，同時実行制御など）が明らかでない．これにより，特に更新演算などにおいて機能的に完成度の高いマシンほど低い性能が報告される可能性がある．

（7） データベース定義が極端に人工的である．たとえば本ベンチマークでは整数をキーとして多用す

るが，現実にはほとんどのキーは 10 進データか文字列であり，整数を用いることは少ない．システムによっては 10 進の算術命令やデータの圧縮により実際の使用環境では大きな性能向上を実現しているものもあり，これら実使用に応じた最適化技術の効果が反映されていない．また，データが一様分散しているのも現実のデータベースとは大きく状況が異なる．

（8） QUEL により問い合わせが記述されている．QUEL による記述は比較的容易に測定システムの問い合わせ言語に変換することができるが，たとえば検索結果を一括してファイル出力する機能は ANSI SQL にはなく，実際には測定システムが提供するなんらかの機能によりこの仕様をシミュレートすることになる．

これら問題点以外にも本ベンチマークの仕様については設計者自身による評価がある[14]．

4. トランザクション処理に対するベンチマーク

4.1 TPC ベンチマークの背景

トランザクション処理については，広く受け入れられる確立した性能評価方法がなかった．その理由として，バッチ処理からオンライン処理まで適用分野に応じて多様な利用形態が存在することやオンライントランザクション処理（以下では OLTP と略す）の性能はハードウェアとソフトウェアの総合されたシステムの性能であることなどが考えられる．このような環境の中で，ベンダは各社各様の性能評価を行ってきたというのが実情であった．

しかし，ここ数年，アメリカのベンダを中心として，類似したモデルに基づいたトランザクション処理の性能値が多く公表されるようになってきた．TP1や ET1 の名前で呼ばれるベンチマークの性能値である．これらのモデルが類似している理由は，いずれも，1985 年に Datamation 誌に掲載された論文[16]の Debit-Creditベンチマークに基づいているためであが，これらのモデルに明確な定義があるわけではない．

Debit-Credit は，アメリカの銀行の口座取引を単純化したモデルである．窓口の端末から投入されるメッセージに従ってデータベースをアクセスし，取引結果を端末に送り返す．データベースは，Account（口座），Teller（出納係），Branch（支店），History（履歴）の 4 種類であり，Account, Teller, Branch を更新し，History に取引履歴を格納する．また，Debit-Credit モデルには，データベースの大きさや端末数などの環境の大きさをトランザクション処理件数に比例させるという特徴がある．図-4 と図-5 に，Debit-Credit の環境とトランザクションプロファイルを示す．

類似したモデルによる性能値の公表は，性能評価のグラウンドが共通になる可能性を示して好ましい反

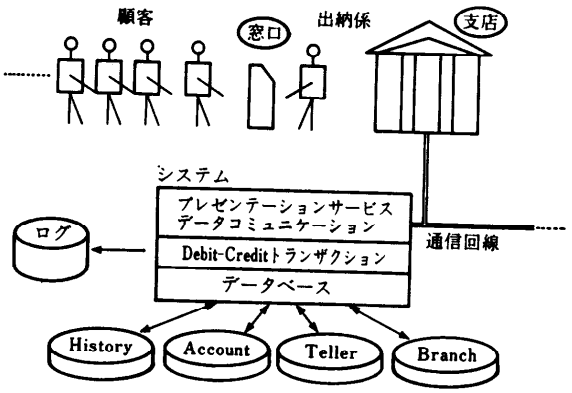
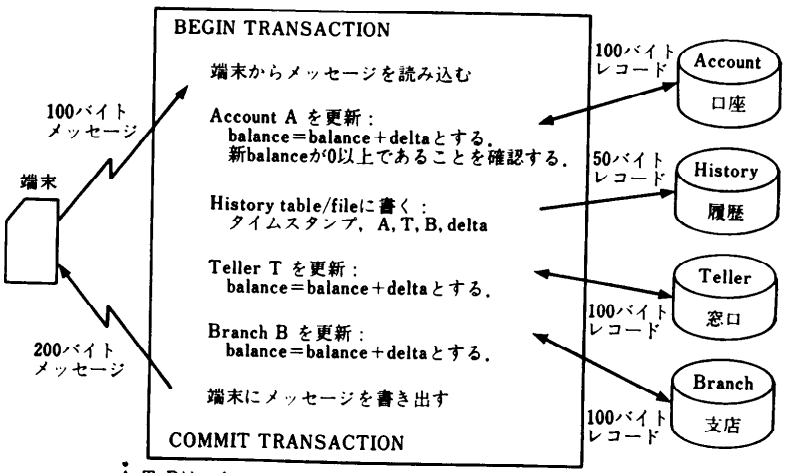


図-4 Debit-Credit の環境



*A, T, Bは，Account, Teller, Branchのレコード/行のキー

図-5 Debit-Credit モデルのトランザクションプロファイル

面，類似しているが同一でないことによる混乱も生じている．具体的には以下のような混乱である．

（1） 通信とデータベースの両方の処理を行うシステムとデータベース処理のみを行うシステムの性能評価が，同じ性能指標（tps と cost/tps）により行われているが，両者の性能値は比較不可能であり紛らわしい．

（2） Debit-Credit モデルには，規定の曖昧な点や標準として未完成な点がみられる．たとえば，トランザクションの性質や測定期間は規定がなく，cost/tps のコストの対象を計算機室としている．また，通信プロトコルも X.25 に特定している．これらの結果，ベンダ間に解釈の相違が生まれ，評価基準や評価方法が統一されていない．

このような混乱を解消し OLTP ベンチマークを標準化することを目的に設立されたのが TPC (Transaction processing Performance Council) である．

TPC は，1989 年 6 月に Omri Serlin (ITOM International Corporation) と Tom Sawyer (Codd & Date Consulting) により提案され，同年 8 月に 8 社をメンバとして発足したベンダ主導の OLTP ベンチマーク標準化機構である．現在は，アメリカ，ヨーロッパ，日本の 30 社以上の主要なシステムベンダ，データベースベンダがこれに加盟している．

1989 年 10 月末時点の加盟メンバを以下に示す．

Arix, AT&T, BiiN, Cognos, CAI, *CDC, Cullinet, DG, *DEC, Fujitsu America, HP, Hitachi, IBM, *ICL, Informix, NCR, NEC, Olivetti, Oracle, Prime, *Pyramid, RT, Sequent, Sequoia, Siemens, Software AG, *Stratus, Sun Microsystems, *Sybase, *Tandem, Teradata, Unify, Unisys, *Wang (* は最初からのメンバ)

TPC は，通信とデータベースの両方の処理を適度に含むモデルに TPC Benchmark ™A という名前を付け，最初にこのモデルの検討を行った．このベンチマークは，Debit-Credit モデルに対応するものであるが，オリジナルモデルに対して，規定を厳密化し標準としての完成度を高めた．TPC Benchmark ™A は，パブリックレビュー (TPC に加盟していない機関や企業，個人のコメントを求める公開レビュー) のフェーズを経て，1989 年 10 月の最終ミーティングにより確定された．TPC では現在，データベース処理の負荷を高めたベンチマーク (TPC Benchmark ™B) の検討を行っている．

図-6 に，現在までの TPC の歴史を簡単に紹介する．

4.2 TPC ベンチマーク[17]

ここでは，標準として確立されている TPC Benchmark ™A について，目的と概要を簡単に紹介し，モデルの特徴をなすトランザクションとデータベースの定義，評価と測定方法について説明する．以下，TPC Benchmark ™A のことを単に TPC ベンチマークとよぶ．

4.2.1 TPC ベンチマークの目的

TPC ベンチマークは，多端末セション，適度の I/O アクセス，トランザクションインテグリティにより特徴付けられる OLTP システムを性能と価格の両面から評価することを目的としたものである．性能

TPC発足までの主な出来事

'84. 夏	Jim Gray の論文	Debit-Credit モデルの原形
'84. 7-12,	Omri Serlin の WG-PMS	
'84. 2.	Tandem TR 85.2	
'85. 4.1	Datamation 誌へ寄稿 ("Anon et al.")	Debit-Credit モデルの公表
'85—'86	Debit-Credit, ET1, TP1 テスト実施される	
'87—'88	Debit-Credit, TP1 テストの公表 (Tandem Nonstop SQL など)	性能値に対する混乱発生

TPC発足後の今日までの活動

'88. 6.	Omri Serlin-Tom Sawyer の TP 設立提案	ベンチマークの原案
'88. 8.10	TPC が 8 メンバーで正式に発足	
	TPC Benchmark ™A の検討開始	通信とデータベースの両方の処理を含むベンチマーク (Debit-Credit に対応)
	〔TPC Benchmark ™A の Draft 案を検討〕	
'89. 5.	TPC Benchmark ™A の Draft 案をフィックス	
'89. 6.	TPC Benchmark ™A のメンバ企業によるレビュー（カンパニレビュー）	
'89. 8-9.	TPC Benchmark ™A の公開レビュー（パブリックレビュー）	
'89. 10.	TPC Benchmark ™A の最終ミーティング，決定	
	TPC Benchmark ™B の検討開始	データベース処理の負荷の高いベンチマーク

図-6 TPC の歴史

は1秒あたりのトランザクションスループット（tps: transaction per second）により，また価格は単位性能あたりの価格（cost/tps: cost per tps）により評価する．また，これらの評価は顧客に公開され，再現可能であることが前提とされている．

4.2.2 TPC ベンチマーク概要

TPC ベンチマークでは，ベンチマークモデルの定義以外に，tps の測定と評価の方法，cost/tps の算出に使用するコストの計上方法も定めている．また，tps と cost/tps に加えて顧客に公開すべき情報，ベンチマーク結果の監査項目なども定めている．

（a） TPC ベンチマークモデルの定義

TPC ベンチマークのトランザクションは，図-5 に対応するプログラムの論理をもっている．トランザクションについては，入出力メッセージや備えるべき ACID 性を細かく定義している．ACID 性とは，Atomicity, Consistency, Isolation, Durability のトランザクションの4つの性質のことである（**4.2.3** 参照）．

トランザクションプログラムがアクセスするデータベースは，Account, Branch, Teller, History の4種類であり，おのおの，1回ずつアクセスするだけであり，データベースの負荷は軽い部類に属する．データベースについては，レコードの内容や分割（partitioning）の方法，入力メッセージとアクセスするデータベースの関係などを定義している．

通信については，回線経由の WAN だけでなく LAN や端末のプロセッサ直結構成も認めている．クライアント・サーバモデルも取り込み，端末を含む通信の構成も定義している．

Debit-Credit の特徴である tps と規模（データベースのレコード数と端末数）の関係も定義されている．

（b） 測定と評価方法

tps は，90%のトランザクションが2秒未満の応答時間をもつという制約のもとで評価する．測定に関しては，測定システムの構成，測定期間，端末エミュレーションの方法，メッセージの発生方法などのテストの実施方法に関するものから，応答時間の測定方法まで，細かく定義している．

（c） コストの求め方

TPC ベンチマークの評価項目の価格には，ハードウェアとソフトウェアを含むシステムの購入価格と5年間のメンテナンスコストを含む．システム購入価格には，回線や LAN を除くすべてのシステム構成要素が含まれる．たとえば，端末や開発環境も含まれる．

（d） ベンチマーク結果の公開と監査

ベンチマーク結果としては，性能と価格だけでなく，システム構成やモデルへの適合性なども公開すべ

表-1 Debit-Credit モデルと TPC Benchmark ™A との主要な相違

項　　目	Debit-Credit	TPC Benchmark ™A
トランザクション プロファイル	トランザクションの中で端末 I/O, balance はチェックする.	トランザクションの外で端末 I/O, balance はチェックしない.
システムの性質	ACID 性について陽に既定されていない	強い ACID が要求される
レスポンスタイム	95% のトランザクションが1 sec. 以下で終了する.	90% のトランザクションが2 sec. 以下で終了する.
測定場所	テスト対象システム (SUT)	ドライバーシステム
テスト期間	言及していない	中断のない15分間，データベースの遅延処理も規定
端末数/tps	100 台	10 台
接続 (Interconnect)	WAN (X. 25)	WAN, LAN またはプロセッサ直結
tps	一種類	tpsA-Wide (WAN) と tpsA-Local (LAN)，両者は相互比較できない.
価格の求め方	計算機室内部にかかるコストのみ	通信回線や LAN を除くすべてのコスト
完全な公開	言及していない	要求される：監査が推奨される
History データベース	一つのものに格納	分割格納可能
Account, Branch, Teller データベース	Account 100,000 レコード/tps Branch 10 レコード/tps Teller 100 レコード/tps	Account 100,000 レコード/tps Branch 1 レコード/tps Teller 10 レコード/tps 相対レコードアクセスは禁止

データベース	レコード長	レコードに含むフィールド	アクセス方法	レコード数/tps
Account	100バイト	Account-ID, Branch-ID Account balance	識別子によるランダム更新 (データをプログラムに返す)	100,000 レコード
Branch	100バイト	Branch-ID, Branch balance	識別子によるランダム更新	1 レコード
Teller	100バイト	Teller-ID, Branch-ID Teller balance	識別子によるランダム更新	10 レコード
History	50バイト	Account-ID, Teller-ID Branch-ID delta, タイムスタンプ	シーケンシャル格納 (追加)	〔測定〕 8時間分のレコード 〔価格〕 90日分のレコード

図-7 データベースの定義

きことを規定している．また，公正を期すために第三者による監査が推奨されている．これらの公開項目や監査項目も細かく規定されている．

表-1 に，Debit-Credit モデルと PTC Benchmark™A の主要な相違を示す．

4.2.3 トランザクションの定義

トランザクションプロファイルは，メッセージの読み込みと書き出しをトランザクションの外で行うことを除き，図-5 と変わらない．また，トランザクション内の操作順番は特に規定されていない．

ACID 性については，Atomicity は，トランザクションが all or nothing であることを，Consistency は，4つのデータベース Account, Branch, Teller, History の内容に矛盾がないことを要求している．Isolation は，TPC ベンチマーク以外のトランザクションとの排他が可能で，かつ明確なグラニュールをもつことを要求している．ただし，システムまたはアプリケーションのどちらが排他要求を発行してもかまわない．

Durability は，TPC ベンチマークで特に厳密に定義している．すべての故障に耐えられるシステムは存在しないことを前提にした上で，以下の単一故障に対して，コミットが完了したトランザクションの効果とデータベースの無矛盾性をリカバリプロセスなどにより保証すべきことを要求している．

— データベースやログデータを含む耐久媒体の回復不可能な故障．耐久媒体には，磁気ディスクなどの本質的に不揮発な媒体と電源内蔵の揮発媒体がある．

— システムクラッシュやシステムハングなどの一時的な中断

— メモリの全体または部分の故障

4.2.4 データベースの定義

TPC ベンチマークでは，ロジカルなデータベースを定義しており実現方法には触れていない．しかし，データベース，特にリレーショナルデータベースを強く意識している．TPC ベンチマークが扱うデータベースは，Account, Branch Teller, History の4つであり，おのおの，口座データ，支店データ，出納係データ，取引履歴データを表している．トランザクションプログラムは，Account, Branch, Teller のデータを更新し，History にデータ格納するだけで，データベースの操作は比較的軽いものとなっている．また，History はすべてのトランザクションのデータ格納要求が逐次化される場所であり，ボトルネックポイントとして設定されている．図-7 に，データベースの定義を示す．データベースに関する TPC ベンチマークの他の特徴を以下に示す．

— Account の更新結果はプログラムに返す必要がある．Branch と Teller は返さなくてもよい．

— プログラムがレコードの物理的なアドレスを意識してはいけない．たとえば，相対レコード番号によるアクセスは認められない．

— データベースの分割 (partitioning) は，レコード群単位の水平分割は認められる．しかし，フィールド群単位の垂直分割は許されない．

— データベースの書き出しをトランザクションのコミットに同期させる必要はない．ただし，この書き出しの遅延により，システムクラッシュ時のリカバリ時間が長くなってはならない．

— 入力メッセージの 85% は，投入元の端末が属する Branch とその配下の Account を更新し，15% は投入元の端末が属さない Branch とその配下の Account を更新する (85-15 ルール)

4.2.5 測定と評価方法

90%が2秒未満の応答時間をもつトランザクション

の1秒あたりのスループットが tps であり，応答時間と tps は厳密を期すために測定対象システム(SUT: Sytem Under Test) の外側にドライバシステムを接続して測定する．ドライバシステムは，また，メッセージの発生や受取を行う端末エミュレータ (RTE: Remote Terminal Emulator) でもある（もちろん，エミュレータではなく実端末を使ってもよい）．**図-8** に，測定システムの構成を示す．

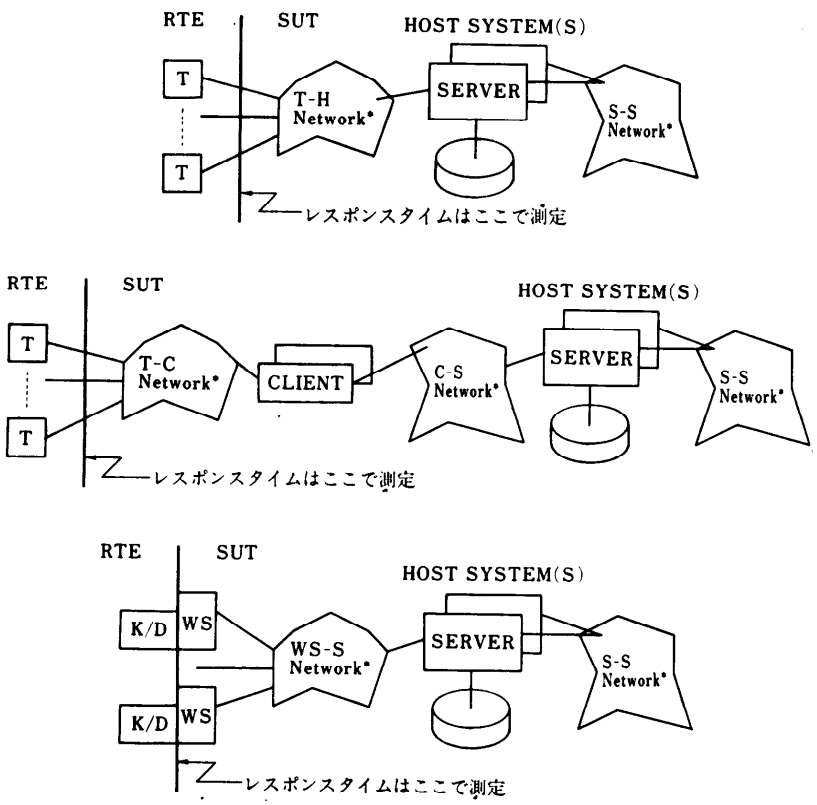
測定と評価に関する TPC ベンチマークの他の特徴を以下に示す．

— 測定期間は定常状態に達してから 15 分間以上とする．しかし，SUT は8時間以上連続走行できる環境（たとえば，ログデータ格納媒体）をもつ必要がある．

— メッセージは，思考時間と応答時間を合わせたメッセージ発生間隔（サイクル時間という）の平均が 10 秒以上となるように発生させる．

また，アクセス対象データベースは，85-15 ルールに従って均一に選択する必要がある．

— 測定結果については，応答時間の頻度を示すグラフ（**図-9**），tps と応答時間の関係を示すグラフ（**図-10**），応答時間の最大値と平均値を報告しなければならない．



記号: C—Client, H—Host, K/D—Keybord/Display, S—Server, T—Terminal, RTE—Remote Terminal Emulator, SUT—System Under Test, ＊—オプション

図-8 測定システムの構成

5. データベースベンチマークの将来

5.1 関係データベースに対するベンチマークの将来

3. で述べたように，ウィスコンシンベンチマーク自身は技術的また現実的側面双方において幾つかの問題があり，この反省からウィスコンシンベンチマーク以降の研究は上記問題点を解決する方向へと進んでいる．これら研究動向は，ベンチマークそのものの洗練化以外に，単純ベンチマーク群の実行結果から特定アプリケーションの性能予測を行う技法の確立[7]，多重処理性能評価ベンチマークの開発[5),6)]，応答時間など以外の新しい性能評価指標の確立[12),13)]，標準化 (SQL による定義)[12)-14)]などがある．またウィスコンシンベンチマーク自身もデータベースサイズを拡張するために改良が加えられている（この新しいベンチマークでは 100 万レコードまでが処理対象とされる）[10]．

一方，これら研究的・技術的立場とは別に，データベースシステムの性能比較ツールという現実的立場から標準的ベンチマーク確立に対する要求は強い．前述のように，元来ウィスコンシンベンチマークはデータベース管理システム（データベースマシン）の最適化機能や基本性能を把握する（特殊なアルゴリズムやハードウェアが実際に有効であるかなどの検証）ためのものであり，システム全体の性能の優

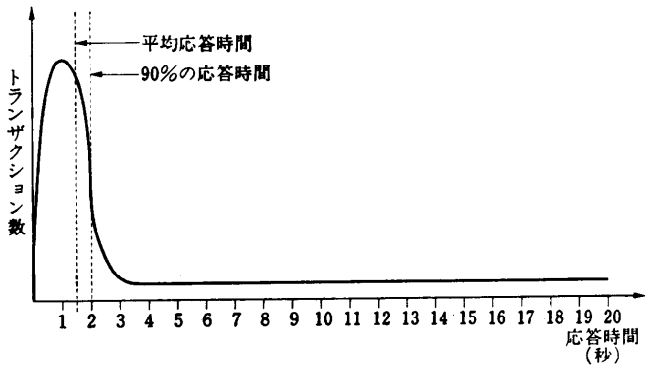


図-9 応答時間の頻度グラフ

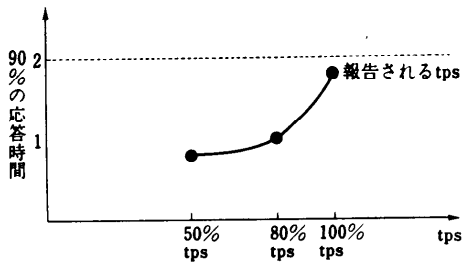


図-10 tps と応答時間の関係を示すグラフ

劣を決定するようには設計されていない．しかし実際にはこの点が理解されず，また商用システムの性能値が論文発表により一方的に公表されたため，当然のことながら一部該当ベンダから本ベンチマークに対して強い批判が行われた．関係データベースが本格的に普及し始めた現在，ユーザ・開発者・研究者が簡易にシステム性能を把握するためのツールとして標準ベンチマークの開発は急務である．このためには TPC ベンチマークのように広く学界，業界の意見を求めつつ，関係データベース処理に対する標準的ベンチマークを確立していく姿勢が求められよう．

5.2 TPC の役割と将来

TPC は，OLTP の観点からトランザクション処理の標準化に取り組んでいる．ある程度のシステム構成を取り入れてはいるが，モデル自身は，OLTP の多様な利用形態のうちの単純な形態しか表していない．この意味では，実際の顧客システムの性能を示すベンチマークの役割を担うことはできない．しかし，OLTP システムの性能比較の一つの基準として，現在の Debit-Credit に基づくベンチマークの混乱の解消は期待できる．

OLTP の性能評価を一つのベンチマークでカバーすることは難しい．利用形態の多様さに対応して，幾つかのベンチマークを用意する必要がある．TPC では，データベース処理の負荷を高めたベンチマークを TPC Benchmark™ B として検討中であるが，利用形態の多様性を取り入れながらも体系的にベンチマークを提供していく必要があると思われる．

6. おわりに

データベース処理における2種類のベンチマークについて述べた．ベンチマークによる性能評価技法では，その評価項目がシステム全体の性能を適切に反映するように設計する必要があるが，これは一般に容易ではない．データベースシステムに対するベンチマークに関してはいまだ研究が不十分であり，今後より詳細な検討が望まれる．TPC の活動は高く評価できるが，ベンチマークセットをより豊富にすることが不可欠である．米国での標準化を待つことなく我が国においても独自のベンチマーク作成活動を押し進めることが必要である．

参考文献

1) Hawthorn, P. and DeWitt, D. J.: Performance Evaluation of Database Machines, IEEE Trans. on Software Engineering (Mar. 1982).
2) DeWitt, D. J. and Hawthorn, P.: Performance Evaluation of Database Machine Architectures 1981 Proc. of Very Large Database Conference (Sep. 1981).
3) Bitton, D., DeWitt, D. J. and Turbyfill, C.: Benchmarking DatabaseSystems: A Systematic Approach, Technical Report # 526, Computer Science Dep., Univ. of Wisconsin (Dec. 1983).
4) Bitton, D., DeWitt, D. J. and Turbyfill, C.: Benchmarking Database Systems: A Systematic Approach, 1983 Proc. of VLDB Conference (Oct. 1983).
5) Boral, H. and DeWitt, D. J.: "A Methodology for Database System Performance Evaluation", Proc. of 1984 SIGMOD Conference, Boston (June 1984).
6) Bitton, D. and Turbyfill, C.: "Design and Analysis of Multi-User Benchmarks for Database Systems", TR 84-589, Dep. of Computer Science, Cornell Univ. (Jan. 1984).
7) DeWitt, D. J.: A Single User Evaluation of Gamma Database Machine, Database Machines

and Knowledge Base Machines, Kitsuregawa ed. Kluwer Pub. (1987).
8) Stonebraker, M.: Tips on Benchmarking Data Base Systems, ibid.
9) Hawthorn, P.: "Variations on a Benchmark, ibid.
10) DeWitt, D. J. et al.: A Performance Analysis of the Gamma Database Machine, Proc. of 1988 SIGMOD Conference (1988).
11) Stonebraker, M.: The Design and Implementation of INGRES, ACM TODS 1, 3 (Sep. 1976).
12) Turbyfill, C., Orji, C. and Bitton, D.: AS3 AP-A Comparative Relational Database Benchmark, Proc. of COMPCON '89 (1989).
13) Turbyfill, C.: Comparative Benchmarking of Relational Database Systems, Ph. D Dissertation, Cornell Univ. (1987).
14) Bitton, D. and Turbyfill, C.: "A Retrospective on the Wisconsin Benchmark" in 「Readings in Database Systems」, Morgan Kaufmann Publishers (1989).
15) Kitsuregawa, M. et al.: Query Execution for Large Relation of Functional Disk System, Proc. of Int. Conf. on Data Engineering (Feb. 1989).
16) Anon. et al.: "A Measure of Transaction Processing Power", Datamation, pp. 112-118, April 1, 1985.
17) TPC, "TPC BENCHMARK™ A Draft 6-PR Proposed Standard (PUBLIC REVIEW VERSION)", August 21, 1989

(平成元年11月21日受付)

付録 1 Wisconsin ベンチマークの定義

Schema Specification for INGRES Benchmark

```
create onektup(unique1A=i2, unique2A=i2, twoA=i2, fourA=i2, tenA=i2,
twentyA=i2, hundredA=i2, thousandA=i2, twothousA=i2, fivethousA=i2,
tenthousA=i2, odd100A=i2, even100A=i2, stringu1A=c52, stringu2A=c52, string4A=c52)

create twoktup(unique1B=i2, unique2B=i2, twoB=i2, fourB=i2, tenB=i2,
twentyB=i2, hundredB=i2, thousandB=i2, twothousB=i2, fivethousB=i2,
tenthousB=i2, odd100B=i2, even100B=i2, stringu1B=c52, stringu2B=c52, string4B=c52)

create fivektup(unique1C=i2, unique2C=i2, twoC=i2, fourC=i2, tenC=i2,
twentyC=i2, hndredC=i2, thousandC=i2, twothousC=i2, fivethousC=i2,
tenthousC=i2, odd100C=i2, even100C=i2, stringu1C=c52, stringu2C=c52, string4C=c52)

create tenktup1(unique1D=i2, unique2D=i2, twoD=i2, fourD=i2, tenD=i2,
twentyD=i2, hundredD=i2, thousandD=i2, twothousD=i2, fivethousD=i2,
tenthousD=i2, odd100D=i2, even100D=i2, stringu1D=c52, stringu2D=c52, string4D=c52)

create tenktup2(unique1E=i2, unique2E=i2, twoE=i2, fourE=i2, tenE=i2,
twentyE=i2, hundredE=i2, thousandE=i2, twothousE=i2, fivethousE=i2,
tenthousE=i2, odd100E=i2, even100E=i2, stringu1E=c52, stringu2E=c52, string4E=c52)
```

Queries Used to Construct Bprime1 and Bprime2

```
range of t is tenKtup2
retrieve into Bprime1(t.all) where (t.unique2E<1000)

range of w is tenKtup1
retrieve into Bprime2(w.all) where (w.unique2D<1000)
```

Specification of Storage Structures for INGRES Benchmark

Used for Query in Table 3, Column 1

```
modify tenKtup1 to hash on unique2D

modify tenKtup2 to hash on unique2E
```

Used for All Queries with Indices

```
modify tenKtup1 to isam on unique2D
index on tenKtup1 is wz(unique1D)
index on tenKtup1 is a1(hundredD)
modify wz to isam on unique1D
modify a1 to isam on hundredD

modify tenKtup2 to isam on unique2E
index on tenKtup2 is wq(unique1E)
index on tenKtup2 is a2(hundredE)
modify wq to isam on unique1E
modify a2 to isam on hundredE
```

Benchmark Queries in INGRES Format

```
range of t is tenKtup1
range of x is tenKtup2
```

Selection with 1% selectivity factor
Table 1, Column 1 and Table 2, Column 1

```
retrieve into skr101(t.all) where t.unique2D<100
retrieve into skr102(x.all) where x.unique2E>9899
```

retrieve into skr103 (t. all) where (t. unique2D>301) and (t. unique2D<402)
retrieve into skr104 (x. all) where (x. unique2E>675) and (x. unique2E<776)
retrieve into skr105 (t. all) where (t. unique2D>964) and (t. unique2D<1065)
retrieve into skr106 (x. all) where (x. unique2E>451) and (x. unique2E<552)
retrieve into skr107 (t. all) where (t. unique2D>171) and (t. unique2D<272)
retrieve into skr108 (x. all) where (x. unique2E>458) and (x. unique2E<559)
retrieve into skr109 (t. all) where (t. unique2D>617) and (t. unique2D<718)
retrieve into skr1010 (x. all) where (x. unique2E>838) and (x. unique2E<939)

Selection with 10% selectivity factor
Table 1, Column 2 and Table 2, Column 2

retrieve into skr101 (t. all) where t. unique2D<1000
retrieve into skr102 (x. all) where x. unique2E>8999
retrieve into skr103 (t. all) where (t. unique2D>791) and (t. unique2D<1792)
retrieve into skr104 (x. all) where (x. unique2E>311) and (x. unique2E<1312)
retrieve into skr105 (t. all) where (t. unique2D>902) and (t. unique2D<1903)
retrieve into skr106 (x. all) where (x. unique2E>467) and (x. unique2E<1468)
retrieve into skr107 (t. all) where (t. unique2D>887) and (t. unique2D<1888)
retrieve into skr108 (x. all) where (x. unique2E>985) and (x. unique2E<1986)
retrieve into skr109 (t. all) where (t. unique2D>534) and (t. unique2D<1535)
retrieve into skr1010 (x. all) where (x. unique2E>647) and (x. unique2E<1648)

Selection with 1% selectiviy factor using non-clustered index

retrieve into skr101 (t. all) where t. unique1D<100
retrieve into skr102 (x. all) where x. unique1E>9899
retrieve into skr103 (t. all) where (t. unique1D>301) and (t. unique1D<402)
retrieve into skr104 (x. all) where (x. unique1E>675) and (x. unique1E<776)
retrieve into skr105 (t. all) where (t. unique1D>964) and (t. unique1D<1065)
retrieve into skr106 (x. all) where (x. unique1E>451) and (x. unique1E<552)
retrieve into skr107 (t. all) where (t. unique1D>171) and (t. unique1D<272)
retrieve into skr108 (x. all) where (x. unique1E>458) and (x. unique1E<559)
retrieve into skr109 (t. all) where (t. unique1D>617) and (t. unique1D<718)
retrieve into skr1010 (x. all) where (x. unique1E>838) and (x. unique1E<939)

Selection with 10% selectivity factor using non-clustered index

retrieve into skr101 (t. all) where t. unique1D<1000
retrieve into skr102 (x. all) where x. unique1E>8999
retrieve into skr103 (t. all) where (t. unique1D>791) and (t. unique1D<1792)
retrieve into skr104 (x. all) where (x. unique1E>311) and (x. unique1E<1312)
retrieve into skr105 (t. all) where (t. unique1D>902) and (t. unique1D<1903)
retrieve into skr106 (x. all) where (x. unique1E>467) and (x. unique1E<1468)
retrieve into skr107 (t. all) where (t. unique1D>887) and (t. unique1D<1888)
retrieve into skr108 (x. all) where (x. unique1E>985) and (x. unique1E<1986)
retrieve into skr109 (t. all) where (t. unique1D>534) and (t. unique1D<1535)
retrieve into skr1010 (x. all) where (x. unique1E>647) and (x. unique1E<1648)

Select 1 Tuple to Screen

retrieve (t. all) where t. unique2D=2001
retrieve (x. all) where x. unique2E=2452
retrieve (t. all) where t. unique2D=3014
retrieve (x. all) where x. unique2E=3613
retrieve (t. all) where t. unique2D=3275
retrieve (x. all) where x. unique2E=4799
retrieve (t. all) where t. unique2D=3745
retrieve (x. all) where x. unique2E=3950
retrieve (t. all) where t. unique2D=2217
retrieve (x. all) where x. unique2E=2418

Selection with 1% selectivity factor. Retrieve to Screen

retrieve (t. all) where t. unique2D<100
retrieve (x. all) where x. unique2E>9899
retrieve (t. all) where (t. unique2D>301) and (t. unique2D<402)
retrieve (x. all) where (x. unique2E>675) and (x. unique2E<776)
retrieve (t. all) where (t. unique2D>964) and (t. unique2D<1065)
retrieve (x. all) where (x. unique2E>451) and (x. unique2E<552)
retrieve (t. all) where (t. unique2D>171) and (t. unique2D<272)
retrieve (x. all) where (x. unique2E<458) and (x. unique2E<559)
retrieve (t. all) where (t. unique2D>617) and (t. unique2D<718)
retrieve (x. all) where (x. unique2E>838) and (x. unique2E<939)

JoinAselB

```
range of t is tenKtup1
range of w is tenKtup2
retrieve into tmpj1 (t. all, w. all)
where (t. unique2D=w. unique2E) and (w. unique2E<1000)
```

```
range of t is tenKtup2
range of w is tenKtup1
retrieve into tmpj2 (t. all, w. all)
where (t. unique2E=w. unique2D) and (w. unique2D<1000)
```

JoinABprime

```
range of t is tenKtup1
range of w is Bprime1
retrieve into tmpj3 (t. all, w. all) where (t. unique2D=w. unique2E)
range of t is tenKtup2
range of w is Bprime2
retrieve into tmpj4 (t. all, w. all) where (t. unique2E=w. unique2D)
```

JoinCselAselB

```
range of o is oneKtup
range of t is tenKtup1
range of w is tenKtup2
retrieve into tmpj5 (o. all, t. all) where (o. unique2A=t. unique2D)
and (t. unique2D=w. unique2E) and (w. unique2E<1000) and (t. unique2D<1000)
range of o is oneKtup
range of t is tenKtup2
range of w is tenKtup1
retrieve into tmpj6 (o. all, t. all) where (o. unique2A=t. unique2E)
and (t. unique2E=w. unique2D) and (w. unique2D<1000) and (t. unique2E<1000)
```

sJoinAselB

```
range of t is tenKtup1
range of w is tenKtup2
retrieve into tmpj1 (t. all, w. all)
where (t. unique1D=w. unique1E) and (w. unique1E<1000)
range of t is tenKtup2
range of w is tenKtup1
retrive innto tmpj2 (t. all, w. all) where (t. uniqe1E=w. uuique1D) and (w. unique1D<1000)
```

sJoinABprime

```
range of t is tenKtup1
range of w is Bprime1
retrieve into tmpj3 (t. all, w. all) where (t. unique1D=w. unique1E)
range of t is tenKtup2
range of w is Bprime2
retrieve innto tmpj4 (t. all, w. all) where (t. unique1E=w. unique1D)
```

JoinCselAselB

```
range of o is oneKtup
range of t is tenKtup1
range of w is tenKtup2
retrieve into tmpj5 (o. all, t. all) where (o. unique1A=t. unique1D)
and (t. unique1D=w. unique1E) and (w. unique1E<1000) and (t. unique1D<1000)
range of o is oneKtup
range of t is tenKtup2
range of w is tenKtup1
retrieve into tmpj6 (o. all, t. all) where (o. unique1A=t. unique1E)
and (t. unique1E=w. unique1D) and (w. unique1D<1000) and (t. unique1E<1000)
```

Projection Query with 1% Selectivity Factor

```
range of x is tenKtup1
range of y is tenKtup2
retriev into pro1 (x. twoD, x. fourD, x. tenD, x. twentyD, x. hundredD, x. string4D)
retrieve into pro2 (y. twoE, y. fourE, y. tenE, y. twentyE, y. hundredE, y. string4E)
```

Projection Query 1000/1000

```
range of x is oneKtup
retrieve unique into prol (x.all)
```

Scalar Aggregate

```
range of t is tenKtup1
retrieve into min1 (x=min (t. unique2D))
retrieve into min3 (x=min (t. unique2D))
```

```
range of x is tenKtup2
retrieve into min (x=min (x. unique2E))
retrieve into min4 (x=min (x. unique2E))
```

Aggregate Function Min

```
range of t is tenKtup1
range of x is tenKtup2
retrieve into min1 (x=min (t. twothousD by t. hundredD))
retrieve into min2 (x=min (x. twothousE by x. hundredE))
retrieve into min3 (x=min (t. twothousD by t. hundredD))
retrieve into min4 (x=min (x. twothousE by x. hundredE))
```

Aggregate Function Sum

```
retrieve into sum1 (x=sum (t. twothousD by t. hundredD))
retrieve into sum2 (x=sum (x. twothousE by x. hundredE))
retrieve into sum3 (x=sum (t. twothousD by t. hundredD))
retrieve into sum4 (x=sum (x. twothousE by x. hundredE))
```

Deletion Queries

```
range of x is tenKtup1
delete x where x. unique2D=10001
delete x where x. unique2D=10002
delete x where x. unique2D=10003
delete x where x. unique2D=10004
delete x where x. unique2D=10005
```

```
range of y is tenKtup2
delete y where y. unique2E=10001
delete y where y. unique2E=10002
delete y where y. unique2E=10003
delete y where y. unique2E=10004
delete y where y. unique2E=10005
```

Modify Key Attribute

```
range of t is tenKtup1
range of x is tenKtup2
replace t (unique2D=10001) where t. unique2D=1491
replace x (unique2E=10001) where x. unique2E=1491
replace t (unique2D=1491) where t. unique2D=8075
replace x (unique2E=1491) where x. unique2E=8075
replace t (unique2D=8075) where t. unique2D=74
replace x (unique2E=8075) where x. unique2E=74
replace t (unique2D=74) where t. unique2D=7023
replace x (unique2E=74) where x. unique2E=7023
replace t (unique2D=7023) where t. unique2D=10001
replace x (unique2E=7023) where x. unique2E=10001
```

Modify Non-Key Attribute

```
replace t (unique1D=10001) where t. unique1D=1491
replace x (unique1E=10001) where x. unique1E=1491
replace t (unique1D=1491) where t. unique1D=8075
replace x (unique1E=1491) where x. unique1E=8075
replace t (unique1D=8075) where t. unique1D=74
replace x (unique1E=8075) where x. unique1E=74
replace t (unique1D=74) where t. unique1D=7023
replace x (unique1E=74) where x. unique1E=7023
replace t (unique1D=7023) where t. unique1D=10001
replace x (unique1E=7023) where x. unique1E=10001
```

Append Queries

append to tenKtup1 (unique2D=10001, unique1D=74, twoD=0, fourD=2, tenD=0, twentyD=10, hundredD=50,
thousandD=688, twothousD=1950, fivethousD=4950, tenthousD=9950, odd100D=1, even100D=100
stringu1D="MxxxxxxxxxxxxxxxxxxxxxxxxxxxGxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2D="GxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
string4D="OxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxxO")
append to tenKtup2 (unique2E=10001, unique1E=74, twoE=0, fourE=2, tenE=0, twentyE=10, hundredE=50,
thousandE=688, twothousE=1950, fivethousE=4950, tenthousE=9950, odd100E=1, even100E=100,
stringu1E="MxxxxxxxxxxxxxxxxxxxxxxxxxxxGxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2E="GxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
string1E="OxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxxO")
append to tenKtup1 (unique2D=10002, unique1D=74, twoD=0, fourD=2, tenD=0, twentyD=10, hundredD=50,
thousandD=688, twothousD=1950, fivethousD=4950, tenthousD=9950, odd100D=1, even100D=100,
stringu1D="MxxxxxxxxxxxxxxxxxxxxxxxxxxxGxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2D="GxxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
string4D="OxxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxxO")
append to tenKtup2 (unique2E=10002, unique1E=74, twoE=0, fourE=2, tenE=0, twentyE=10, hundredE=50,
thousandE=688, twothousE=1950, fivethousE=4950, tenthousE=9950, odd100E=1, even100E=100,
stringu1E="MxxxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2E="GxxxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
string4E="OxxxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxxO")
append to tenKtup1 (unique2D=10003, unique1D=74, twoD=0, fourD=2, tenD=0, twentyD=10, hundredD=50,
thousandD=688, twothousD=1950, fivethousD=4950, tenthousD=9950, odd100D=1, even100D=100,
stringu1D="MxxxxxxxxxxxxxxxxxxxxxxxxxxxGxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2D="GxxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
string4D="OxxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxxO")
append to tenKtup2 (unique2E=10003, unique1E=74, twoE=0, fourE=2, tenE=0, twentyE=10, hundredE=50,
thousandE=688, twothousE=1950, fivethousE=4950, tenthousE=9950, odd100E=1, even100E=100,
stringu1E="MxxxxxxxxxxxxxxxxxxxxxxxxxxxGxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2E="GxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
string4E="OxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxxO")
append to tenKtup1 (unique2D=10004, unique1D=74, twoD=0, fourD=2, tenD=0, twentyD=10, hundredD=50,
thousandD=688, twothousD=1950, fivethousD=4950, tenthousD=9950, odd100D=1, even100D=100,
stringu1D="MxxxxxxxxxxxxxxxxxxxxxxxxxxxGxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2D="GxxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
string4D="OxxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxxO")
append to tenKtup2 (unique2E=10004, unique1E=74, twoE=0, fourE=2, tenE=0, twentyE=10, hundredE=50,
thousandE=688, twothousE=1950, fivethousE=4950, tenthousE=9950, odd100E=1, even100E=100,
stringu1E="MxxxxxxxxxxxxxxxxxxxxxxxxxxxGxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2E="GxxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
stringu4E="OxxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxO")
append to tenKtup1 (unique2D=10005, unique1D=74, twoD=0, fourD=2, tenD=0, twentyD=10, hundredD=50,
thousandD=688, twothousD=1950, fivethousD=4950, tenthousD=9950, odd100D=1, even100D=100,
stringu1D="MxxxxxxxxxxxxxxxxxxxxxxxxxxxGxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2D="GxxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
string4D="OxxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxxO")
append to tenKtup2 (unique2E=10005, unique1E=74, twoE=0, fourE=2, tenE=0, twentyE=10, hundredE=50,
thousandE=688, twothousE=1950, fivethousE=4950, tenthousE=9950, odd100E=1, even100E=100,
stringu1E="MxxxxxxxxxxxxxxxxxxxxxxxxxxxGxxxxxxxxxxxxxxxxxxxxxxxxxC",
stringu2E="GxxxxxxxxxxxxxxxxxxxxxxxxxxxCxxxxxxxxxxxxxxxxxxxxxxxxxA",
string4E="OxxxxxxxxxxxxxxxxxxxxxxxxxxxOxxxxxxxxxxxxxxxxxxxxxxxxxO")