Seagaia 2004 in 京都
プログラマズキャンプ

XML DB：
パフォーマンス／スケーラビリティの点からディスカッション

Caché

日本ダイナシステム株式会社
鈴木　利明

(1)XMLデータ保存管理向けデータベース製品紹介 Caché

- Cachéは米国InterSystems社が開発、販売
  http://www.intersystems.co.jp
- 米国では病院情報システムの標準エンジン
  - 国立大学共通ソフト　経営分析　(FAIR)

Partners HealthCare System CTO
Steve Flammini**氏**

当社はCaché**を使用して、リレーショナル･モデルではなく、実世界を映し出すデータ･モデルを開発します。**

Intersystems**社の**HomePage**より**

書籍情報

- **オブジェクトデータベースCaché入門**
  W. **キルステン**／M. **イリンガー**／M. **キューン**／B. **レーリッヒ　著**
  大櫛　陽一　監修
  小田嶋　由美子／**インターシステムズジャパン株式会社　訳**
  B5　**並製**　429**頁**　**本体** 6,800**円＋税**
- **5月下旬出版予定**
  ISBN 4-431-71062-0
  CコードC3055

Caché**システムの特徴**

- **データは消さない**
- **少ない台数のサーバーで運用**
- **少人数の運用スタッフ**
- **多いクライアント、端末の同時接続**
- **高速レスポンス**
- VB，Web、Telnet、RS232C..**が同時に同じデータベースをアクセスする**

Caché プログラミングの特徴

- プログラマのスキルに合わせた開発
  - Object(Class)　(VB,JAVA,C++プログラマ)
  - SQL　(RDBからのプログラマ)
  - Object Scripts (Old M言語プログラマ)
- 複合型(混在型) がおすすめ

→ クライアントソフトのシームレスな移行

- クラス利用が便利

Caché と XML

- **階層型データ表現→**XML
  階層型データベース→M言語 (現 Caché)
- XML　　→ Class**格納**
  - XML**構造はクラス構造に対応する**
  - **タグは取り去る**
  - **取り込んでしまえば**Caché**の世界**
- Class**格納→** XML
  - **タグをデータにつけて出力する**

XML文書をインポートする

- %XML.Readerクラス
  - 入力XML文書の解析と検証
  - XML要素に対応する一時オブジェクトを作成する
- %XML.TextReaderクラス
  - XML 文書を巡回するのを容易にする機能
  - DTDまたはXMLスキーマに基づく完全文書検証を可能にする

XML文書をインポートする

例: %XML.Readerクラスを使用してXMLファイルをインポートする

```
ClassMethod ImportXML()
  {
   Set reader = ##class(%XML.Reader).%New()
   Do reader.OpenFile("C:¥XML¥NoticeBoard.xml")
   Do reader.Correlate("note","XMLImport.TextNote")
   While (reader.Next(.object,.sc)) {
          //Save object in database
          Set Status=object.%Save()
   }
  }
```

MML V3

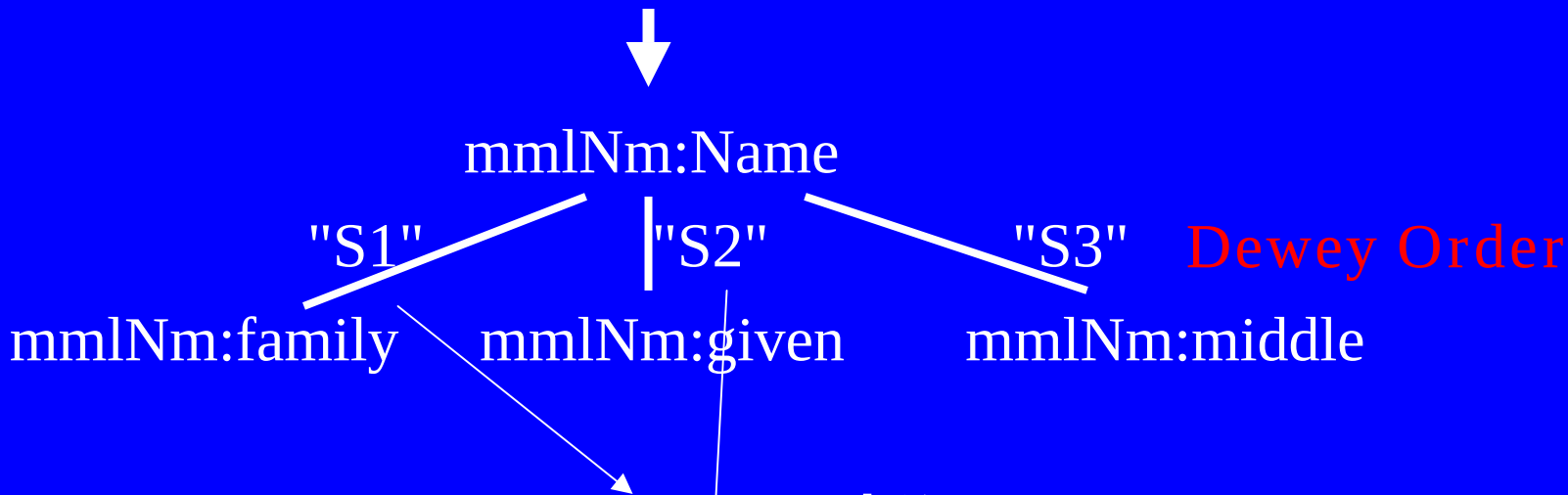
- MML V2.3 を CDA でラッピング
- DTD定義 3065行 (ENTITY展開後)
  - MML V2.3は858行
  - CDA部分に再帰用法がある (NOTE,..)
  - 8階層以上
- かなり複雑な定義である
- クラス化しても取り扱いが難しい
  - 楽にならない

現在の手法

- XML階層データ → 変数添字の階層構造
  - タグは削除、構造と値だけ保存
    - 省DBサイズ
    - XMLファイル12．5G → 6.6G DBサイズ
  - 値が存在しない項目は、変数としない
- ＤＴＤからパーサー、レンダラプログラムを生成するツールを利用
  - 22935行の６９ルーチン

構造射影

<!ELEMENT mmlNm:Name (mmlNm:family ,
mmlNm:given ,
mmlNm:middle?)



^MML(n,0,"S1",0,"S20",3,…,"S1",0)=**松沼**

^MML(n,0,"S1",0,"S20",3,…,"S2",0)=**裕彦**

添え字構造へ1:1 射影

^MML(3,0,"S1",0,"S23",1,-3)=MML

^MML(3,0,"S1",0,"S23",1,-2)=mmlheader

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S1",0)=12345678

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S1",0,-4)=JMARI7

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S1",0,-1)=facility

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S2",0,"S1",1,-2)=MML0025

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S2",0,"S1",1,-1)=I

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S2",0,"S1",1,"S4",0)=医師太郎

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S3",0,"S1",1)=総合病院7

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S3",0,"S1",1,-2)=MML0025

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S3",0,"S1",1,-1)=I

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S3",0,"S2",0)=JMARI7

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S3",0,"S2",0,-4)=MML0027

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S3",0,"S2",0,-1)=JMARI

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S4",0,"S1",1)=泌尿器科

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S4",0,"S1",1,-2)=MML0025

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S4",0,"S1",1,-1)=I

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S4",0,"S2",0)=21

^MML(3,0,"S1",0,"S23",1,"S3",1,"S1",0,"S1",0,"S4",0,"S2",0,-4)=MML0029

試験環境

- Caché v5.0.4
- Athlon 3200+ 1CPU, 1.5GRAM,
- Windows 2000 Professional
- Caché 750Mをバッファに設定
- シングルユーザーと複数ユーザーの試験
- 一部にノートPCでの検査値もあり
  - PIII 800Mhz, 512MRAM, 50Mバッファ

テストデータの件数

- 種　MML文書 25Kバイト(タグ付き)
  - 定義書から取り出す
  - 5個のモジュールが入っている
  - 573フィールド

5万件　　700M

50万件　　6.6Gバイト

500万件　　72.2Gバイト

テスト文書構成

- １患者に５０文書を与えた
  5万文書→1000人(患者)
- 病院は１０種類、乱数で与える
- <mml:title generationPurpose="文書種別">部分一致文字列</mml:title>
  文書種別は10種類乱数で与える
  部分一致文字列は、000～999 を順番に与えた

インデックス

<clinical_document_header><id AAN="病院名" EX="文書番号" RT="病院OID"/>

<mml:masterId><mmlCm:Id mmlCm:type="local">患者番号</mmlCm:Id>

漢字氏名

2種類のインデックス (通常のインデックスとbitmapインデックス) を調べた。

A. set ^MMLindex("HEX",患者番号病院毎,%ObjectId)=""

B. set $bit(^MMLbitIndex("HEX",患者番号病院毎,%i),%o)=1

文書種別, タイトルはインデックスを作成しなかった

通常のインデックス

文書番号

^MMLindex(“**漢字氏名**”,“**三井 健生**”,1101)=“”
^MMLindex(“**漢字氏名**”,“**三井 健生**”,1102)=“”
^MMLindex(“**漢字氏名**”,“**三井 健生**”,1103)=“”
^MMLindex(“**漢字氏名**”,“**三井 健生**”,1104)=“”
^MMLindex(“**漢字氏名**”,“**三井 健生**”,1105)=“”
^MMLindex(“**漢字氏名**”,“**三井 健生**”,1106)=“”
^MMLindex(“**漢字氏名**”,“**三井 健生**”,1107)=“”
:

この部分は自動的圧縮されて管理される

Bitmapインデックス

^MMLbitIndex(“漢字氏名”,“三井 健生”,1)=

<<$C(129,0,144,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)_"ayyyyy"_$C(127)>>

↑

n bit目が文書番号nを表す

bitデータは圧縮されている

６４０００bitを１変数で管理する(１ﾁｬﾝｸ)

試験問題

XML Queryは装備していないため
試験毎にプログラムを作成した
- 通常のインデックスとbitmapインデックス
- 文書種別、タイトルにはインデックスを作らず
  - これにインデックスをつけることもできる

- シングル、マルチユーザー
- 実検索速度のみ計測
  - ５０MML文書ファイル出力(1.1Mバイト)で2.5秒

試験

- 患者IDによる完全一致検索
- 病院IDによる完全一致検索
- 文書番号 AND タイトルの部分一致検索
- 患者ID AND 病院IDによる完全一致検索
- 患者ID AND 病院ID AND文書番号 AND タイトルの部分一致検索

試験結果　5万MML 1 user

	通常index	Bit index	Hit件数
患者検索	4.1ms	1.9ms	500
病院検索	11ms	0.06ms	15000
患者&病院	9.6ms	2.7ms	110
中間一致	60秒	←べた検索	31
患者&病院 中間一致	145ms	75ms	150/0

試験結果　500万MML 1 user

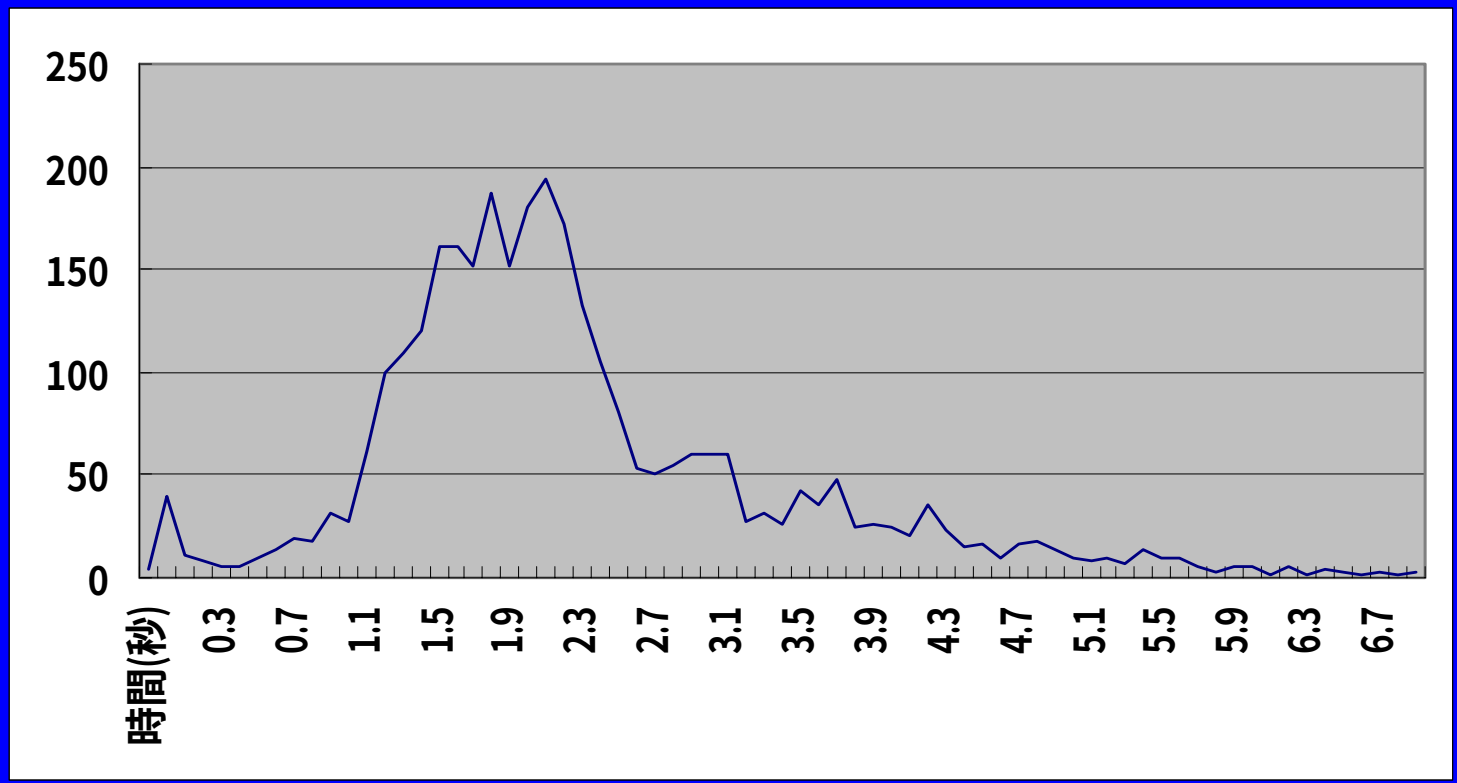
	通常index	Bit index	Hit件数
患者検索	18ms	15ms	5000
病院検索	4606ms	366ms	1500000
患者&病院	89ms (ノート:252ms)	33ms (ノート: 90ms)	150
中間一致	5800秒	←べた検索	2450
患者&病院 中間一致	145ms (ノート:252ms)	75ms (ノート: 90ms)	150/0

500万MML 1 user　Indexがメモリ上の時

	通常index(比率)	Bit index (比率)	
患者検索	0.3ms	0.037ms	
病院検索	983ms	11.4ms	
患者&病院	1.1ms	0.5ms	
中間一致	--	--	
患者&病院 中間一致	1.1ms	0.7ms	

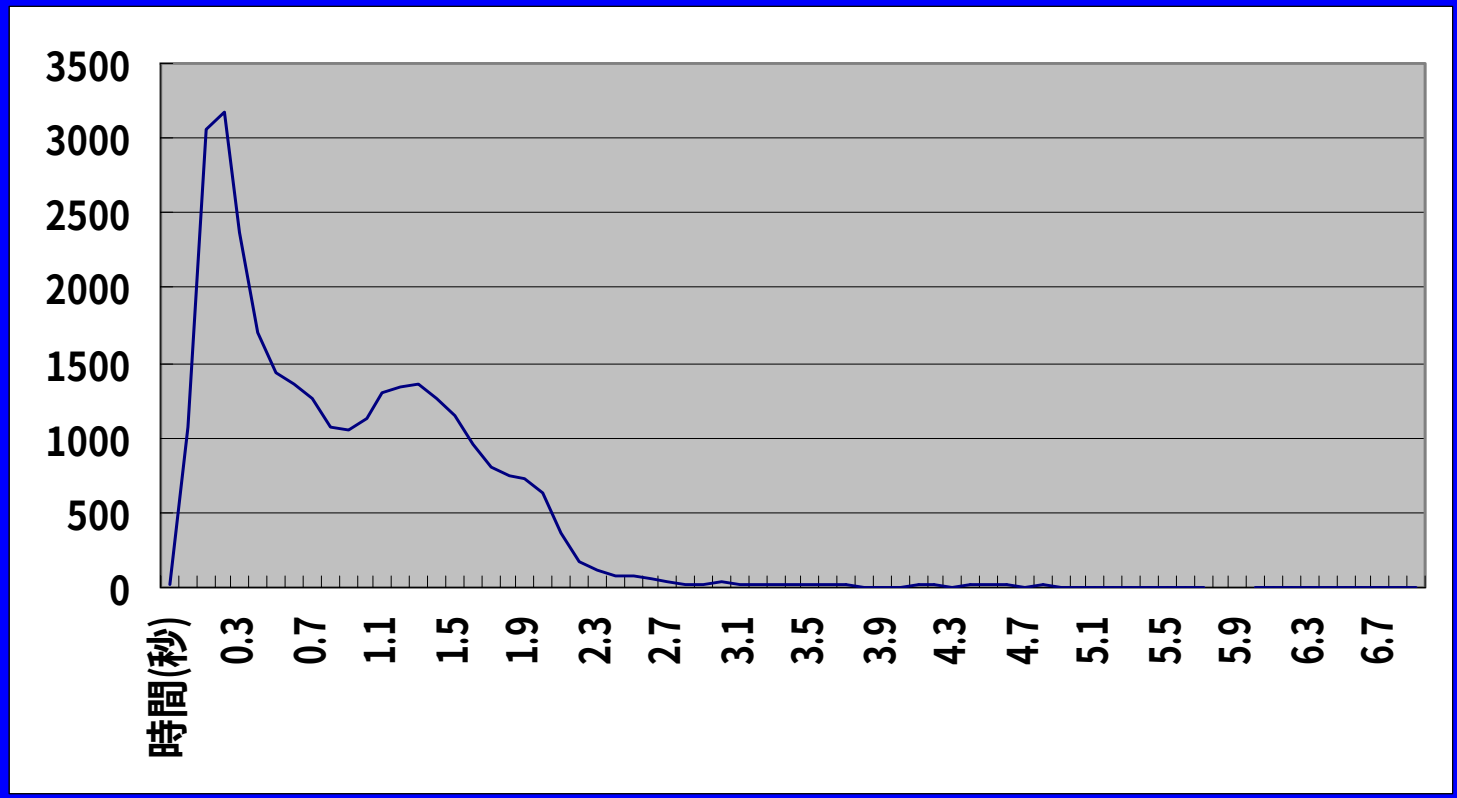
50user/10分 患者&病院&文書番号検索

総計 3401**回** /500**万**MML　　**1回目**



50user/10分 患者&病院&文書番号検索

総計 30250回 /500万MML　　続けて2回目



いろいろなパターンがある

MML**サーバーの容量計算**

- **宮崎大学付属病院**
  - **現在 2,158,541文書 (2004/05/17現在)**
  - 2600 MML文書／日 (28ヶ月間の平均)
  - 5万件は20日分に相当する
- 大規模病院ならば
  - 5000～1万MML文書／日
  - 150万～300万／年
  - 10年保存すると 1500万～3000万文書

今後: Caché MMLサーバー

- 現方法の限界は1億MML文書くらいか
  - 検証が必要
  - ハードディスク　2Tバイトが最低必要
  - 500万→79ﾁｬﾝｸ、1億→1580ﾁｬﾝｸ
- それ以上ならば、何かの手立てが必要
- XMLのインポート、エクスポート速度のUp
- XML Queryの装備が必要