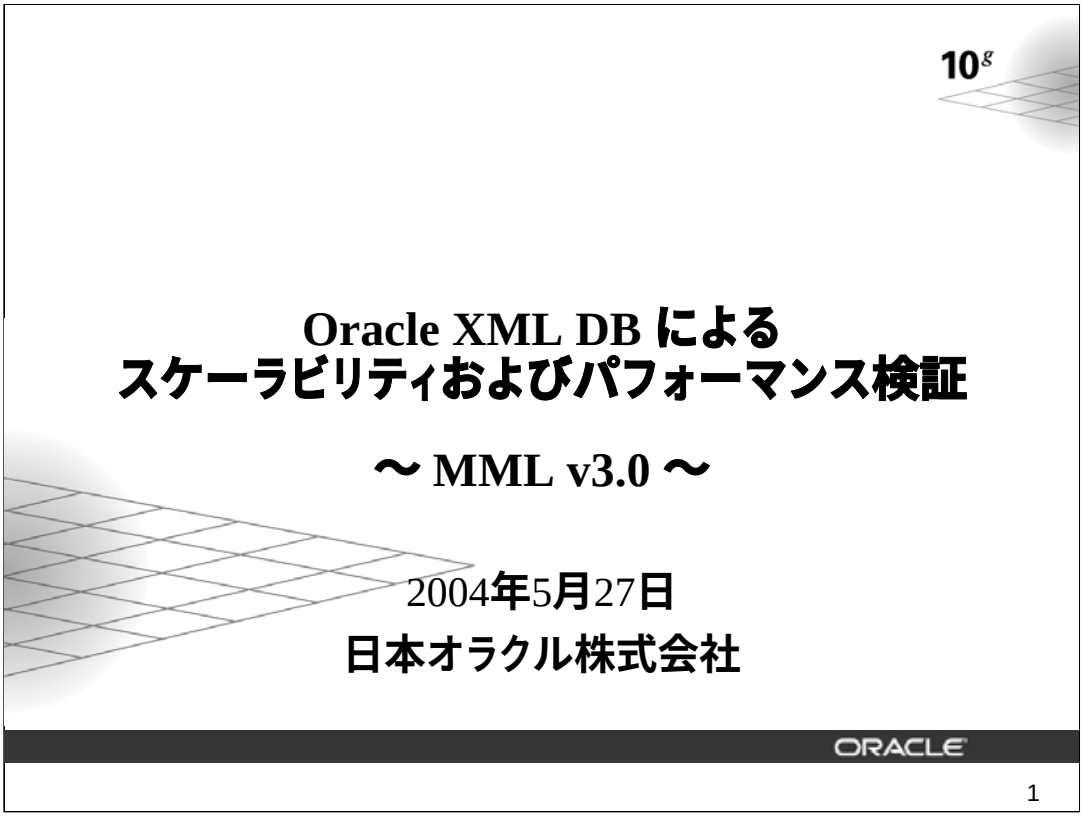
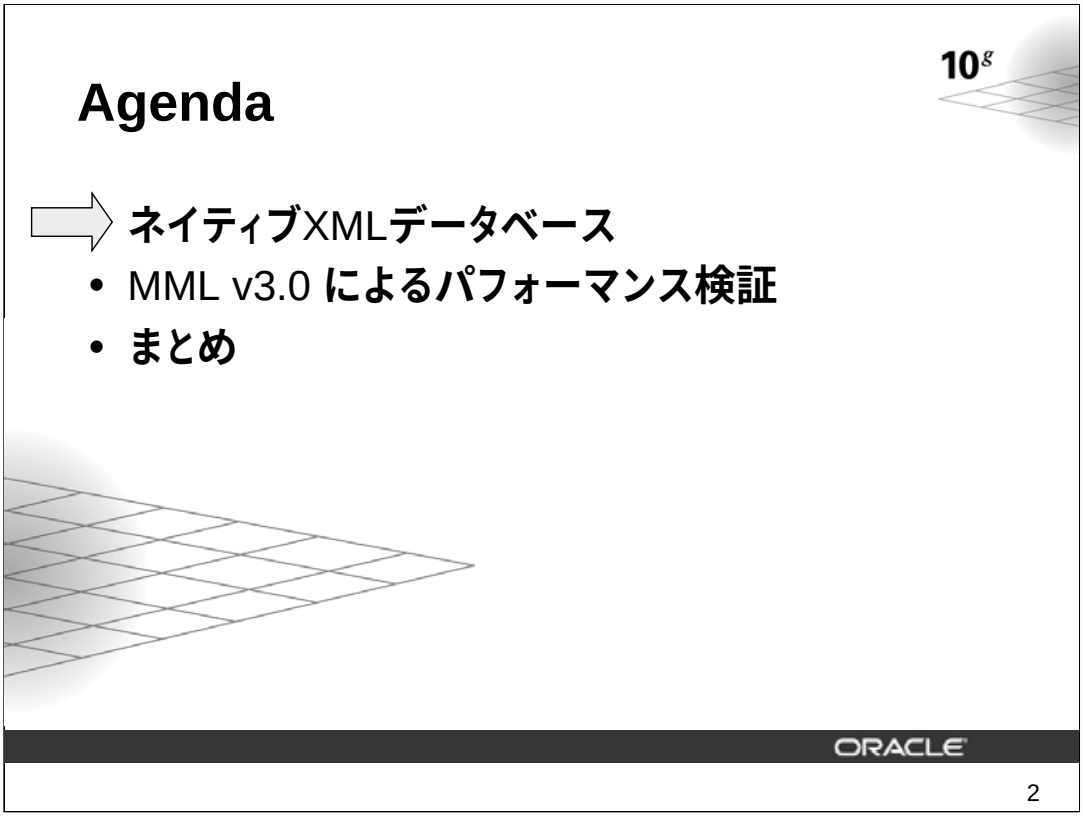


Memo

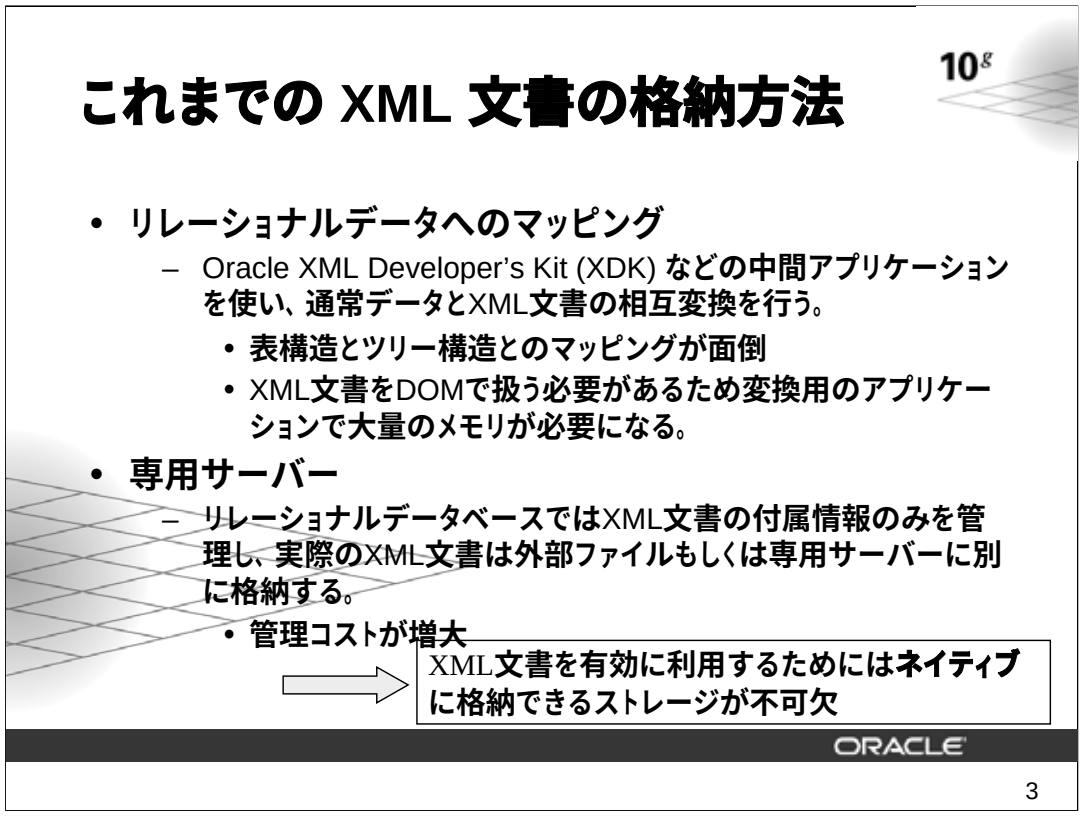


Oracle社では、現バージョンから将来のバージョンに渡って、XMLの機能を提供し続けることをコミットしています。

Oracle XML DBは、XML用のAPIを、ネイティブに実装しています。

Oracle XML DBは、W3Cで勧告されたグローバルな業界標準に準拠しています。主なものとして、XML Schema 1.0、XPath 1.0、XSLT 1.0 といったものが挙げられます。

また、Oracle社は、様々な標準化団体に所属しています。例えばW3C勧告のXML Schemaの策定には、Oracle社のメンバーも関わっています。

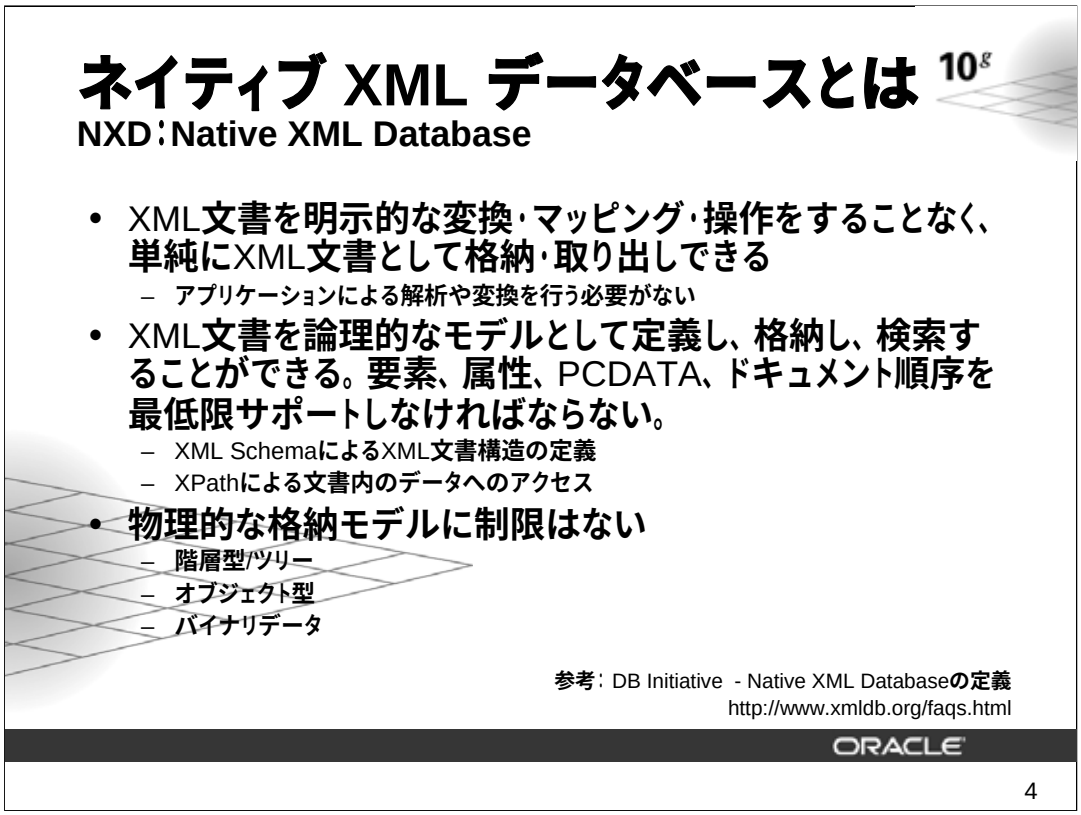


リレーショナルデータへのマッピングは、DB外部でOracle XDK(XML Developer's Kit)のコンポーネントの1つであるOracle XSU(XML SQL Utility)を利用したプログラムで構造化ストレージ(マッピング)を行っていました。

XML文書と表とのマッピングはあくまでもDBの外部で行うため、格納されたデータは、DBからみるとXML文書の1部としてではなく、単なる(オブジェクト)リレーショナル・データとみえます。

また、一度XML文書全体をメモリに読み込むことになるため、多くのメモリを消費していました。

他社のXML専用データベースを使用した場合、XML文書はリレーショナルデータベースと別のデータベースサーバーに格納されることになるため、必然的にコストが増大します。



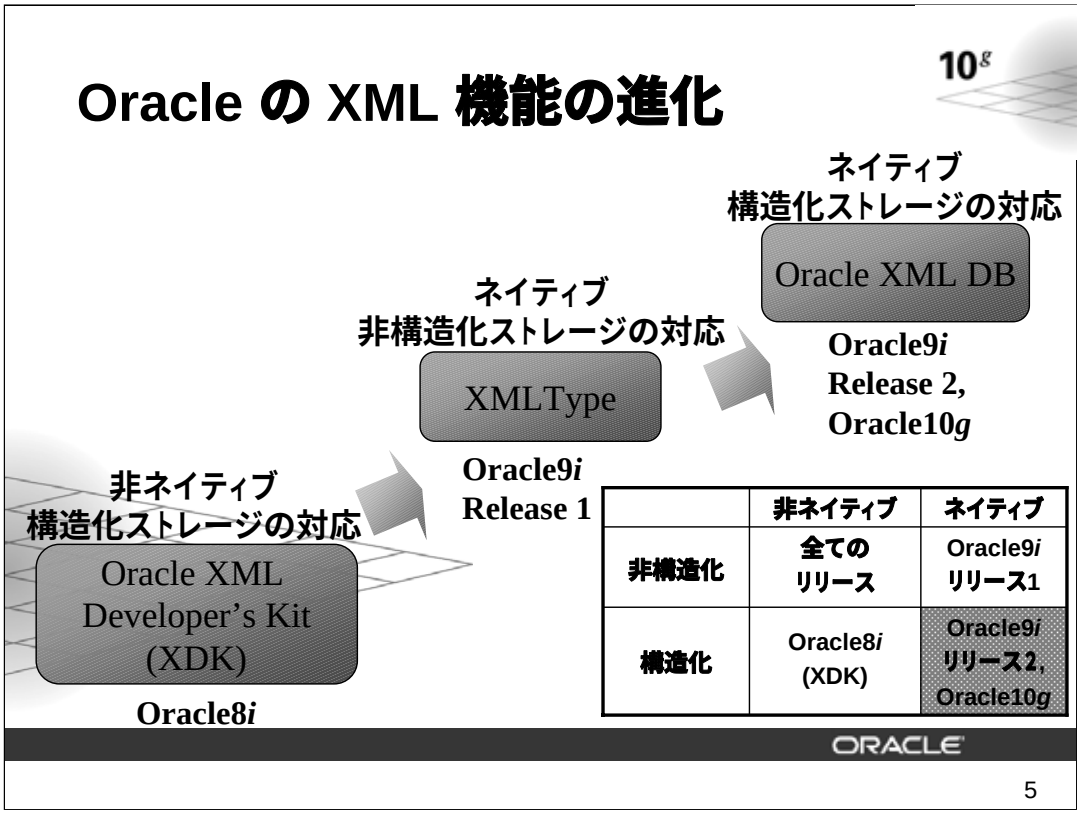
RDBにXML文書を格納するには限界があるため、XML文書を格納するためのデータベース（Native XML Database）が必要になります。NXDに必要な機能は次の通りです。

・ユーザーから見て、XML文書をそのまま出し入れ可能でなければなりません。ユーザーがXML文書を解析してデータベースの構造に合わせて格納するような方法は望ましくありません。

・XML文書に格納されたデータへのアクセスが、XPathなどの専用言語を使用してサーバー側で行えなければなりません。クライアント側でXML文書を読み込んでからデータにアクセスするのは効率が悪いためです。

・XML文書固有の検索方法のサポートも必要になります。例えば、XML形式で格納された住民票データから「秋田」というデータを検索する場合、「住所」タグの中の「秋田」を検索したいのか「名前」タグの中の「秋田」を検索したいのかなどの指定ができなければなりません

・XML文書の拡張に耐え得るものでなくてはなりません。XML文書の構造（スキーマ）が変更されても、データベース側での変更が一切必要ないものでなくてはなりません。また、さまざまな構造をもつXML文書を同じ表や列に格納できなくてはなりません。



Oracle Databaseでは、Oracle8*i*から、バージョンが上がるごとにXMLをサポートするための機能が追加されています。

非ネイティブ構造化ストレージ（Oracle 8*i*）

Oracle8*i*で追加されたOracle XDK(XML Developer's Kit)のコンポーネントの1つであるOracle XSU(XML SQL Utility)を利用する方法です。DB外部で、XSUを利用したプログラムで構造化ストレージ(マッピング)を行います。

XML文書と表とのマッピングはあくまでもDBの外部で行うため、格納されたデータは、DBからみるとXML文書の1部としてではなく、単なる(オブジェクト)リレーショナル・データとみえます。

ネイティブ非構造化ストレージ（Oracle 9*i* R1）

Oracle9*i* リリース1で追加されたXML専用のデータ型であるXMLTypeにXML文書を格納する方法です。

内部ではCLOB型データとして格納するため、 XML文書を非構造化データとして格納されます。

内部的な格納方法はCLOBですが、このデータ型にはXML文書を操作するためのメンバー・メソッドや、SQL関数が用意されおり、XML文書の解析やXPath検索が可能となります。

ネイティブ構造化ストレージ（Oracle 9*i* R2）

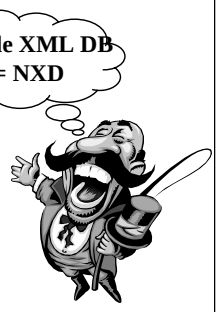
リリース2のXMLTypeを使用することで、ネイティブ構造化ストレージを実現することが可能です。

ユーザーからみるとリリース1と同様のXMLTypeの列にXML文書を格納しているように見えますが、自動的にXML文書の細分化が行われ、オブジェクト型の属性(列)にマッピングして格納されます。

Oracle XML DB と NXD の関係

10g

- XML文書を明示的な変換・マッピング・操作をすることなく、単純にXML文書として格納・取り出しできる
  - 従来の（Oracle9i Release 1で実装したXMLTypeに対する操作の）SQLによる操作に加えて、FTPやWebDAVによる文書の格納・取り出しが可能。
- XML文書を論理的なモデルとして定義し、格納し、検索することができる。要素、属性、PCDATA、ドキュメント順序を最低限サポートしなければならない。
  - XML Schemaに対応した構造化マッピングのXMLTypeデータ
  - 内蔵されたSchema ProcessorによるXML文書の自動検証
  - XPath式などを含む柔軟なSQL操作
- 物理的な格納モデルに制限はない
  - 以下の2つの格納モデルを実装
    - 構造化マッピング
    - テキスト形式（CLOB型）



ORACLE

6

Oracle XML DBでは、通常のinsert文やselect文、またはFTP、WebDAV経由での操作でXML文書の格納・取り出しが可能です。ユーザは変換・マッピングなどをする必要は一切ありません。

Oracle XML DBでは、XML Schemaをサポートしています。XML Schemaをデータベースに登録することで、XML DBに格納されたXML文書は自動的に分解され、データベース内部でオブジェクト型を使用した表に格納されます。XPathを使用した、XML文書の構造に沿った検索が可能になります。XML文書がXML Schemaに基づいた内容になっているかどうかの検証も、データベース内部で自動的に行われます。

XML文書は前述のようにオブジェクト型を使用した表に分解して格納することも可能ですが、単なる文字列として格納することも可能です。

このように、Oracle XML DBはネイティブXMLデータベースの要件を満たしたものであるといえます。

Oracle XML DB のアドバンテージ

10g

- リポジトリの実装
  - XML文書をファイルシステムとして格納・管理できる。
- 既存の表データとの結合
  - SQL文によるXML文書の操作を実装しているので、既存のOracleデータベースに格納された表データの結合検索を行うことができる。
- XML文書に対する整合性の実現
  - XML文書内の要素・属性に対して一意制約や外部参照制約（参照される表は従来のOracleデータベースの表をサポート）を付与することができる。
- 高速な全文検索
  - XML文書にテキスト索引を付与することにより、Oracle Textによる高速な全文検索を行うことができる。
- 容易なメンテナンス
  - 従来のOracleデータと同様のバックアップやリカバリーにも対応

ORACLE

7

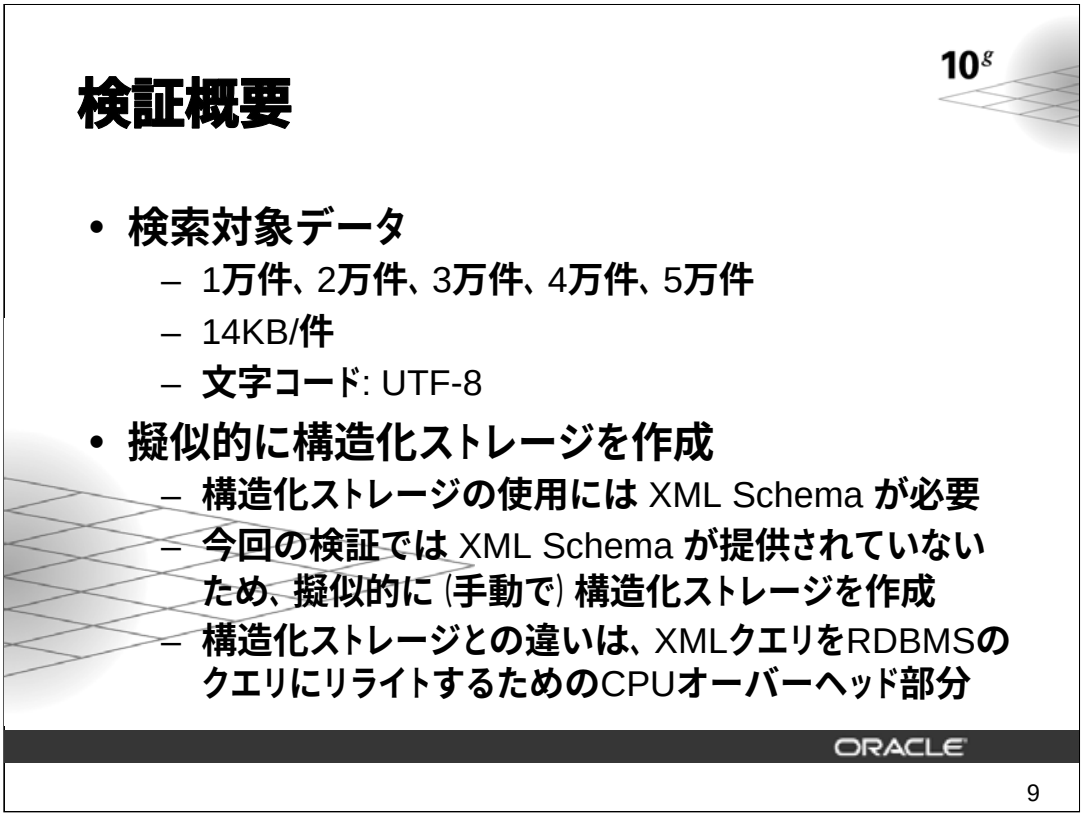
Oracle XML DBにはスライドに挙げたようなアドバンテージがあります。これらを実現したことで、XMLデータをデータベースで扱う上でのあらゆるニーズに対応しています。

IBMのDB2、MicrosoftのSQL Server 2000には、レポジトリはなく、XML Schemaへも対応していません。XML文書にテキスト索引を作成することもできません。

Agenda

- ネイティブXMLデータベース
- MML v3.0 によるパフォーマンス検証
- まとめ

Memo



検証用のサンプルデータは、MML v3.0 規格書 付録B サンプル1 を元に作成しました。作成方法の詳細は、11ページのスライドをご参照ください。

データ件数は、1万件、2万件、3万件、4万件、5万件の5通りです。

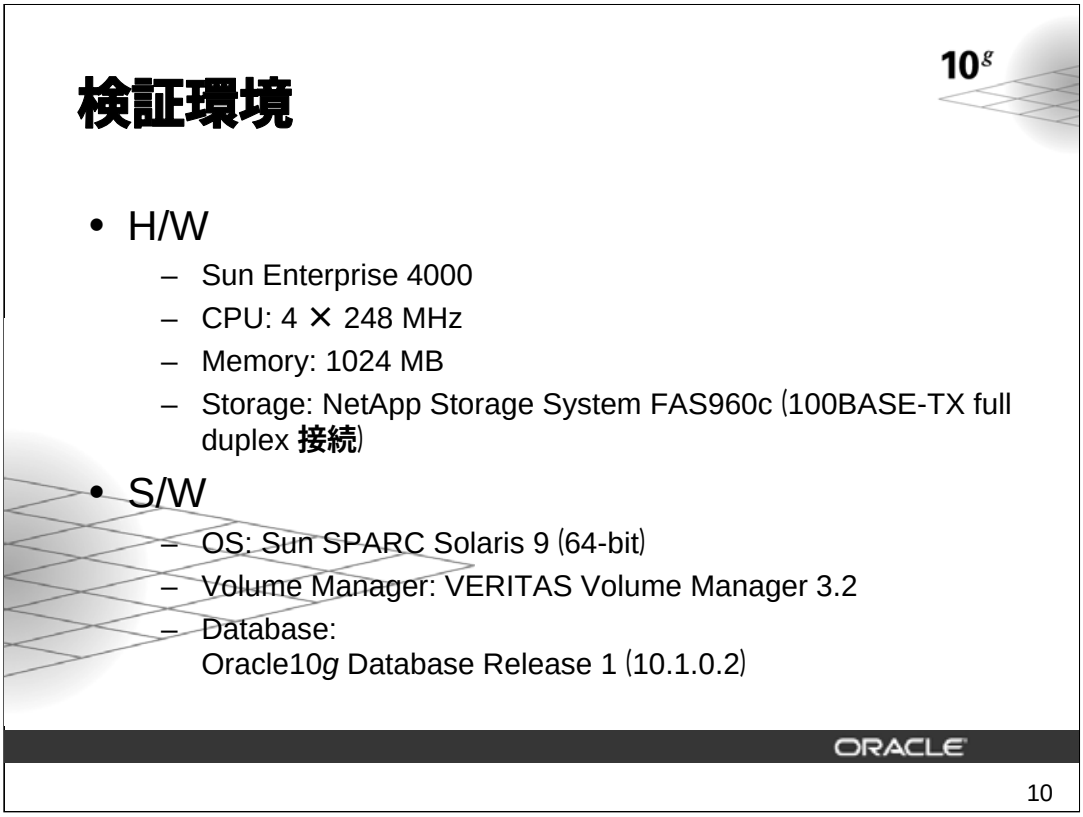
今回の検証で使用したデータベースのキャラクタ・セットがAL32UTF8であったため、事前に検索対象XMLの文字コードをUTF-8に変換してあります。

MML v3.0 では、DTDによって予め検索対象のXML構造が規定されているため、Oracle XML DBの構造化ストレージを使用することで、検索／更新処理を高速化させることが可能です。

しかし Oracle XML DB の構造化ストレージを使用するには、予め XML Schema の登録が必要です。今回の検証では、当該DTDに相当するXML Schemaが与えられなかったため、明示的にリレーショナル・データへのマッピングを行いました。

なお、このようなカスタム構造化手法は、これまでにも、XML検索の高速化を行うために一般的に用いられてきました。

Oracleが自動的にリレーショナル・データへのマッピングを行う方式と、このようにカスタムで構造化を行う方式の違いは、XMLクエリをRDBMSクエリにリライトするためのCPUオーバーヘッド部分です。



今回の検証では、ストレージとして、NAS（Network Attached Storage）デバイスであるNetApp Storage System FAS960cを使用しました。このストレージは、マシンからNFSマウントして使用します。今回の環境では、日本オラクル社内のLAN環境（100BASE-TX full duplex）を経由して検証を行いました。なお、検証マシンとストレージは、それぞれ別のネットワーク・セグメントに属します。また、検証用マシンは、占有環境ではありませんでした。

Full duplex（全二重）とは、双方向通信において、同時に双方からデータを送信したり、受信したりすることができる通信方式のことです。例えば、Ethernetの100BASE-TX仕様で全二重通信を行なった場合、上りと下りそれぞれ100Mbpsの速度となりますので、ケーブル上の転送速度は最大で実質2倍の情報量を転送しているといえます。

データ作成方法

- MML v3.0 規格書 付録B サンプル1 を元にテストデータを作成
  - /levelone/clinical_document_header/id/@EX
    - 1〜 [10000 | 20000 | 30000 | 40000 | 50000]
    - 一意なドキュメントID
  - /levelone/clinical_document_header/patient/person/id/@EX
    - 1〜10000
    - 患者ID
  - /levelone/clinical_document_header/id/@RT および /levelone/clinical_document_header/patient/person/id/@RT
    - 1.2.392.114319.1.5.1.1.1.1.1〜1.2.392.114319.1.5.1.1.1.1.200
    - 施設ID
  - /levelone/body/section/paragraph/content/local_markup/mml:docInfo/mml:title/@generationPurpose
    - MML0007: Generation Purpose に規定された24通り
    - record, recordAdmission, recordInpatient, recordConsult など
    - 文書詳細種別

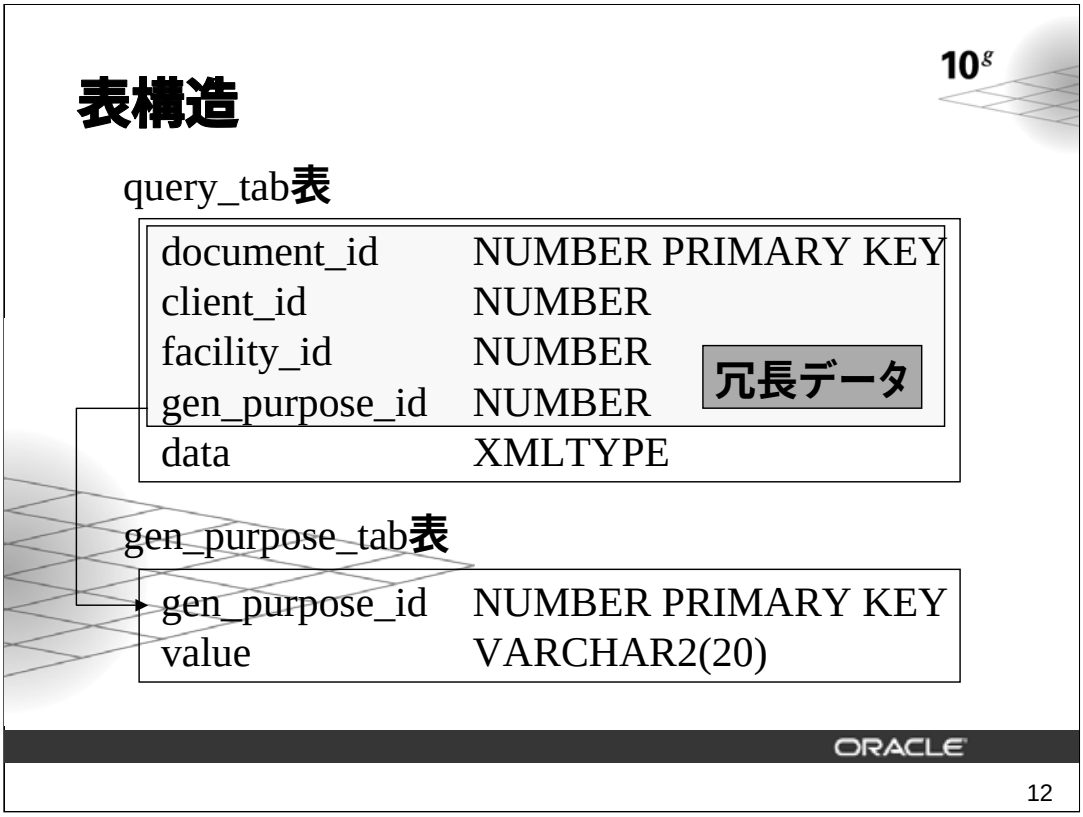
ORACLE

11

サンプルデータ作成のために値を変動させたのは、スライドに挙げられた5つのXPathに相当する部分です。それ以外の部分については、全データ同じ内容となっています。

今回の検証では、XMLTYPEデータ型の列に対して直接問合せを行うことはないため、XMLTYPE列に格納されたXML文書の内容は、検索パフォーマンスには影響を与えません。

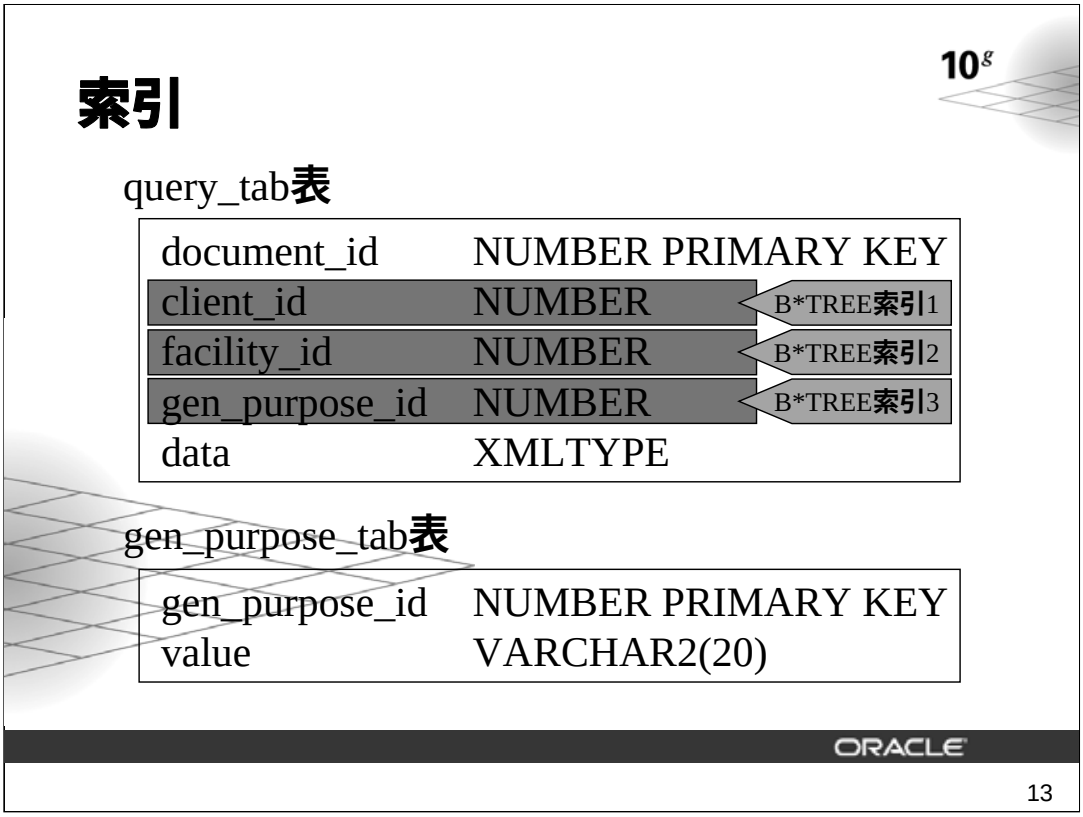
構造化ストレージを使用した場合、検索対象である特定タグの情報以外は、検索パフォーマンスに影響を与えません。したがって、実データを使用した環境においても、検索パフォーマンスは、今回の検証結果とほぼ同等のものが得られると言えます。



構造化ストレージをエミュレートするために、予め属性情報をRDBMSの定型列に抽出しておきます。

各XML文書のドキュメントIDをdocument_id列に、患者IDをclient_id列に、施設IDをfacility_id列に、文書詳細種別IDをgen_purpose_id列に、それぞれ格納します。

文書詳細種別IDとは、各文書詳細種別に割り振られた一意のIDです。文書詳細種別IDと文書詳細種別名の対応は、gen_purpose_tab表で行います。



検索を高速化するために、query_tab表のclient_id列、facility_id列、gen_purpose_id列のそれぞれに、B*TREE索引を作成します。

今回の検証では、B*TREE索引がある場合とない場合の双方について、検索パフォーマンスを測定します。

検索条件

10g

- **検索パターン**1
  - (**患者**ID,**施設**ID) = (2500,13)
    AND
    文書詳細種別='%Post%'
- **検索パターン**2
  - ((**患者**ID,**施設**ID) = (2500,13) OR (5001,26))
    AND
    (**文書詳細種別**='%Post%' OR '%Rad%')
- **検索パターン**3
  - ((**患者**ID,**施設**ID) = (2500,13) OR (5001,26) OR (9501,48))
    AND
    (**文書詳細種別**='%Post%' OR '%Rad%' OR '%Ad%')

ORACLE

14

検索パターンは、スライドに挙げられた3通りです。

検索条件の特徴として、複雑さは、

[検索パターン1] ＜[検索パターン2] ＜ [検索パターン3]

ヒット件数は、

[検索パターン1] ＜[検索パターン2] ＜ [検索パターン3]

となっています。

SQL問合せサンプル

10g

- **検索パターン3**

```
SELECT document_id FROM query_tab
  WHERE (client_id, facility_id) IN (
    (2500,13),(5001,26),(9501,48)
  )
  AND gen_purpose_id IN (
    SELECT gen_purpose_id FROM gen_purpose_tab
      WHERE value LIKE '%Post%'
         OR value LIKE '%Rad%'
         OR value LIKE '%Ad%'
  );
```

ORACLE

15

スライドは、検索パターン3に相当するSQL問合せサンプルです。
検索パターン1、2 に関しても同様です。

検索パターン1

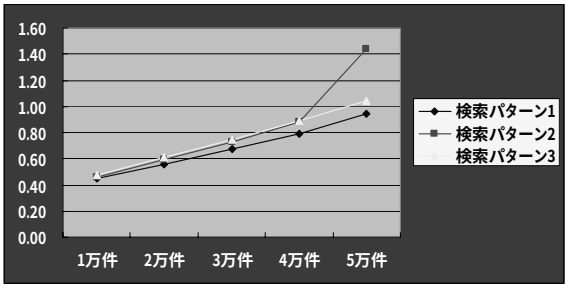
```
SELECT document_id FROM query_tab
  WHERE (client_id, facility_id) IN ((2500,13))
  AND gen_purpose_id IN (
    SELECT gen_purpose_id FROM gen_purpose_tab
      WHERE value LIKE '%Post%'
  );
```

検索パターン2

```
SELECT document_id FROM query_tab
  WHERE (client_id, facility_id) IN ((2500,13),(5001,26))
  AND gen_purpose_id IN (
    SELECT gen_purpose_id FROM gen_purpose_tab
      WHERE value LIKE '%Post%'
         OR value LIKE '%Rad%'
  );
```

検索パフォーマンス (1)

- DB起動直後
- B*TREE索引なし

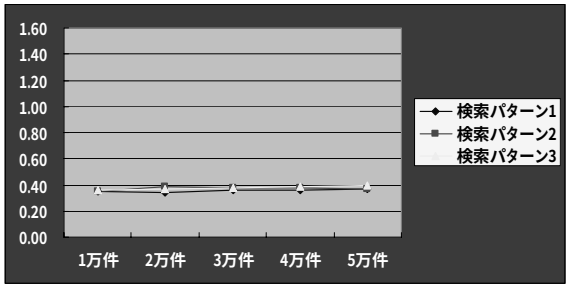


対象データ件数	1万件	2万件	3万件	4万件	5万件
検索パターン1	0.45	0.56	0.67	0.79	0.94
検索パターン2	0.46	0.59	0.73	0.88	1.44
検索パターン3	0.48	0.61	0.75	0.89	1.04

Memo

検索パフォーマンス (2)

- DB起動直後
- B*TREE索引あり



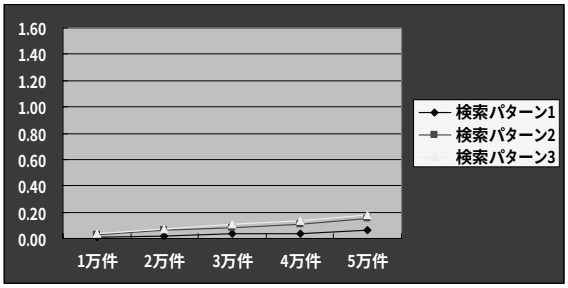
対象データ件数	1万件	2万件	3万件	4万件	5万件
検索パターン1	0.35	0.34	0.36	0.36	0.37
検索パターン2	0.35	0.39	0.38	0.38	0.37
検索パターン3	0.36	0.37	0.38	0.39	0.40

Memo

検索パフォーマンス (3)

10g

- キャッシュ有効
- B*TREE索引なし



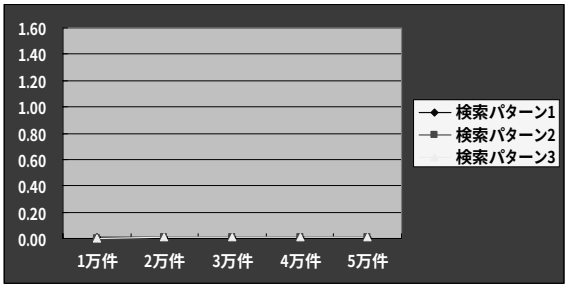
対象データ件数	1万件	2万件	3万件	4万件	5万件
検索パターン1	0.01	0.02	0.04	0.04	0.06
検索パターン2	0.03	0.06	0.08	0.11	0.15
検索パターン3	0.04	0.07	0.11	0.13	0.18

ORACLE

Memo

検索パフォーマンス (4)

- キャッシュ有効
- B*TREE索引あり



対象データ件数	1万件	2万件	3万件	4万件	5万件
検索パターン1	0.01	0.01	0.01	0.01	0.01
検索パターン2	0.00	0.01	0.01	0.01	0.01
検索パターン3	0.00	0.01	0.01	0.01	0.01

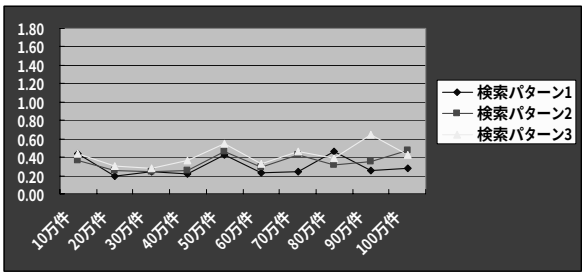
ORACLE

19

Memo

スケーラビリティ検証 (1)

- DB起動直後
- B*TREE索引あり



対象データ件数	10万件	20万件	30万件	40万件	50万件	60万件	70万件	80万件	90万件	100万件
検索パターン1	0.44	0.19	0.24	0.22	0.43	0.23	0.24	0.46	0.26	0.28
検索パターン2	0.36	0.25	0.24	0.26	0.46	0.29	0.42	0.32	0.35	0.47
検索パターン3	0.44	0.31	0.28	0.37	0.55	0.33	0.46	0.39	0.64	0.42

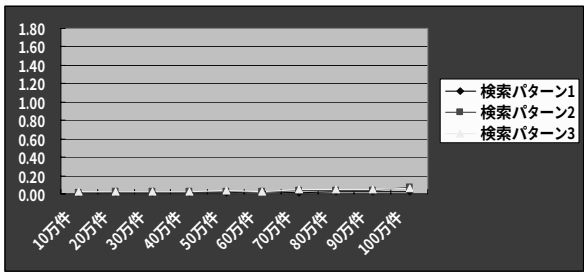
ORACLE

20

Memo

スケーラビリティ検証 (2)

- キャッシュ有効
- B*TREE索引あり



対象データ件数	10万件	20万件	30万件	40万件	50万件	60万件	70万件	80万件	90万件	100万件
検索パターン1	0.01	0.01	0.01	0.01	0.01	0.02	0.01	0.02	0.02	0.02
検索パターン2	0.01	0.02	0.02	0.02	0.03	0.03	0.04	0.04	0.04	0.07
検索パターン3	0.02	0.02	0.02	0.03	0.04	0.03	0.05	0.05	0.05	0.06

ORACLE

21

Memo

考察 (1)

10g

- データ件数とレスポンスタイムの相関
  - B*TREE索引を使用した検索では、データ件数の増加に伴うレスポンスタイムの増加がほとんど見られなかった
    - B*TREE索引の深さ (depth) は、データ件数 n に対して、$\log n$ のオーダーで増大
  - B*TREE索引を使用しない検索においては、一次関数的相関
    - 読込ブロック数はデータ件数にほぼ比例する
    - 縦軸切片が0より大きいのは、CPUオーバーヘッド等の要因による

ORACLE

22

Memo

考察 (2)

- 誤差の発生要因
  - 検証用マシンとストレージ (NetApp) のネットワークセグメントが異なっていた
  - 検証に使用したマシンが占有環境ではなかった

Memo

Agenda

- ネイティブXMLデータベース
- MML v3.0 によるパフォーマンス検証
- まとめ

Memo

まとめ

10g

- リレーショナル・マッピングの使用は、XMLデータ検索スケーラビリティを得るために非常に有効
- B*TREE索引の使用により、スケーラビリティは劇的に向上する
  - スケーラビリティは、データ件数 n に対し、log n のオーダー
- 通常運用時にはキャッシュが有効となっているため、非常に高速な検索を行うことが可能

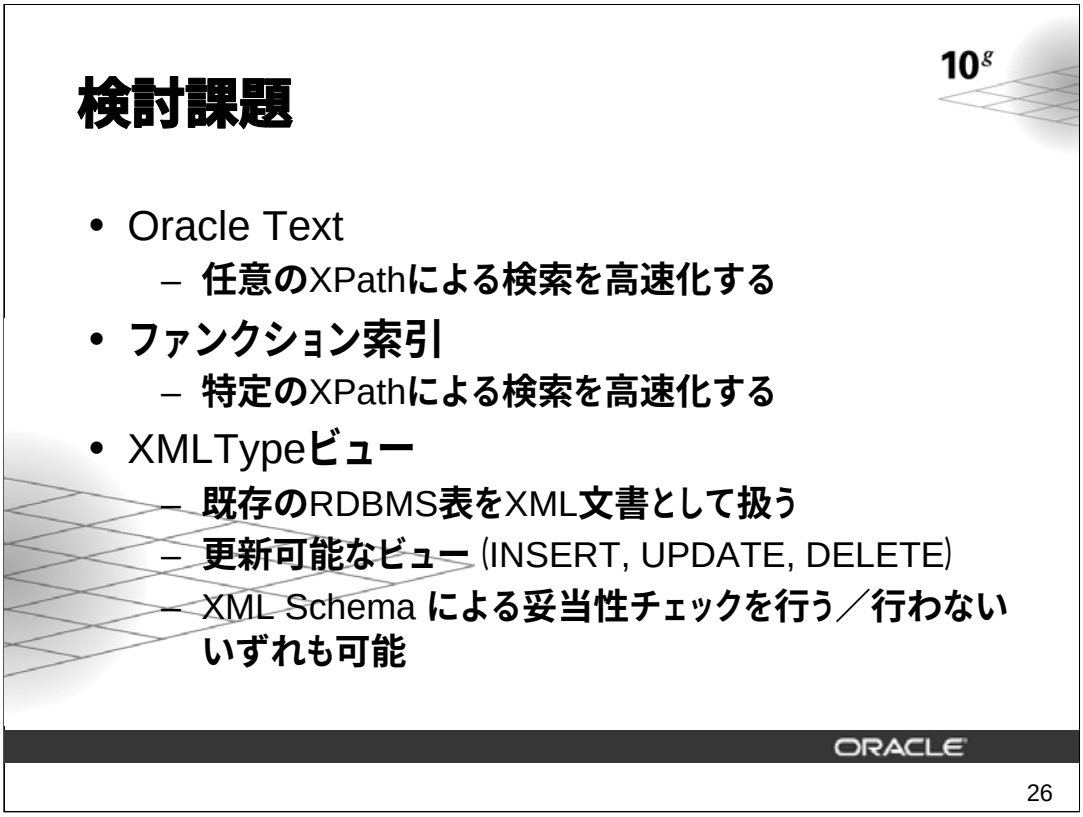
ORACLE

25

今回の検証結果から、リレーショナル・マッピングの使用は、XMLデータの検索スケーラビリティを得る上で非常に有効であることが分かりました。

また、B*TREE索引の使用は、検索パフォーマンスを飛躍的に向上させるのみならず、データ件数の増大に伴うレスポンスタイムの増大を、log *n* のオーダーで抑制します。

通常運用時には、キャッシュが有効となっているため、データベース起動直後に比べて検索パフォーマンスが飛躍的に向上します。



今回の検証では、対象XMLデータのDTDが規定されていることを考慮し、構造化ストレージを使用しました。この方式は、XMLデータの検索スケーラビリティを得るためには非常に有効です。

それ以外で、XMLデータの検索パフォーマンスを向上させるための考慮点としては、以下のものが挙げられます。

Oracle Text

Oracle Text は、Oracle データベース・オブジェクトに完全に統合された全文検索エンジンです。

XMLタグ内に文章が含まれており、なおかつその文章内の単語に基づく検索を実行する場合には、Oracle Textが有効です。

Oracle Text では、XMLのタグ構造に基づく全文検索を行うことも可能です。

ファンクション索引

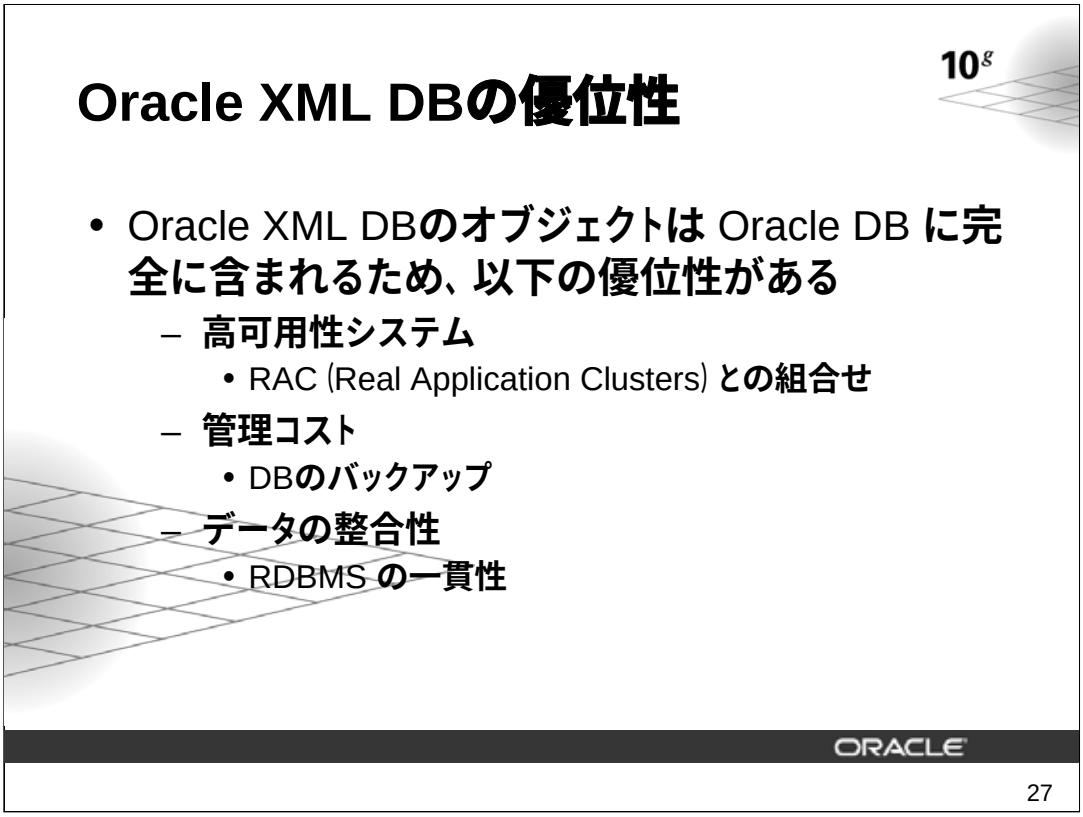
検索を実行するXMLノードが限定されている場合には、ファンクション索引の使用が効果的です。検索が頻発するタグに対して、予めそのタグ値を抽出し、B*TREE索引を作成する機能です。

XMLTypeビュー

既存のRDBMSの表構造を保持しつつ、そのデータをXML文書として参照・更新したい場合には、XMLTypeビューを使用します。

XMLTypeビューでは、XML Schemaによる妥当性チェックを行うことも可能です。

上記機能のうち、Oracle Text、ファンクション索引に関しては、構造化ストレージと組み合わせて使用することが可能です。



Oracle XML DBは、Oracleデータベース・オブジェクトに完全に統合されているため、以下の優位性があります。

高可用性システム

Oracleの高可用性ソリューションであるRAC（Real Application Clusters）と組み合わせた運用が可能です。RACを使用することで、ミッション・クリティカルな医療分野において、ダウンタイムのない運用を行うことが可能です。

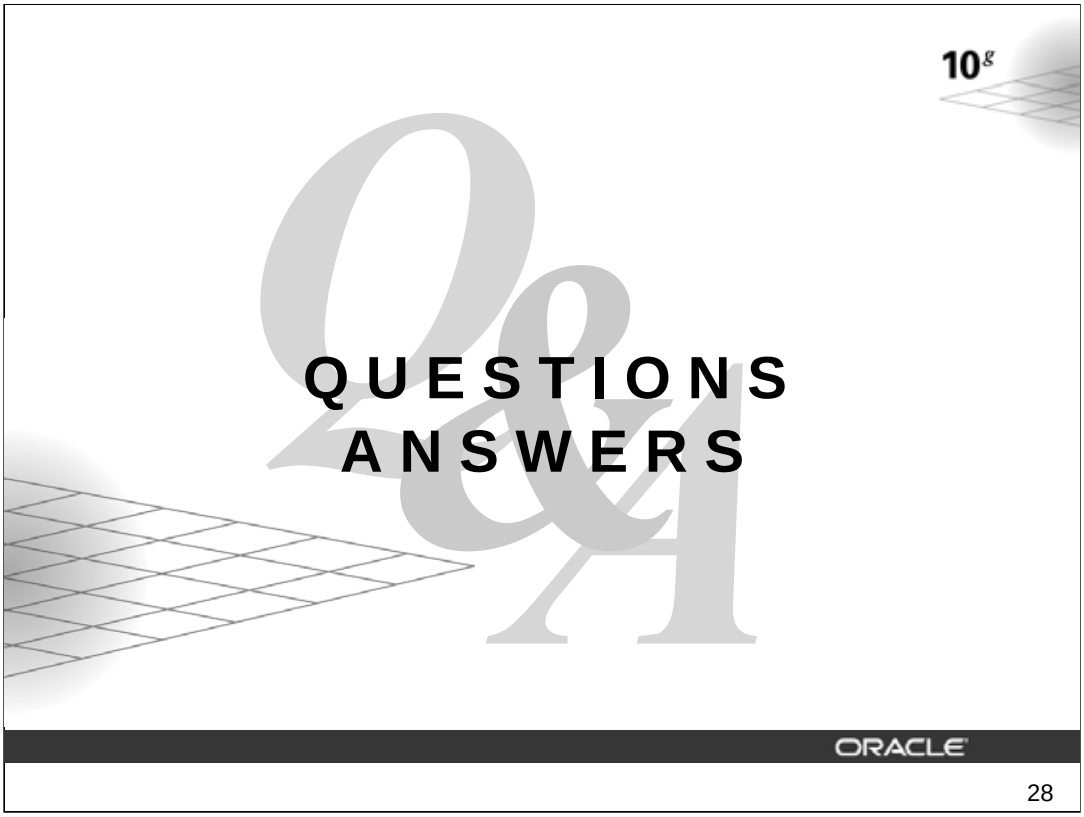
管理コスト

Oracle XML DBでは、XMLデータはすべてOracleデータベース内に保持されます。これにより、データの一元管理、およびバックアップ・リカバリ手順の簡素化を行うことが可能です。

バックアップおよびリカバリ等の管理タスクは、通常のOracleデータベースの運用と全く同様です。

データの整合性

リポジトリDBとしてOracleを使用することで、XMLデータの整合性を保持することができます。通常のファイルシステムだけでは、排他制御、トランザクション・レベルなどを保障することはできません。



Memo

Memo