具体的な講義の項目

- イントロダクション
- データベースのモデル
- 関係データベース（関係代数、関係論理）
- 正規化
- 問い合わせ言語
- トランザクション処理
- データベースの内部構造
- データベースの問い合わせ処理
- その他（オブジェクト指向DB、アクティブDB等々）

他のデータベースのモデル

- オブジェクト指向データベース（OODB）
- オブジェクトリレーショナルデータベース（ORDB）
- 演繹データベース （DDB）
- 演繹オブジェクト指向データベース （DOOD）
- アクティブデータベース（ADB）
- XMLデータベース（XMLDB）
- RDFデータベース（RDFDB）
- データストリーム処理
- …

オブジェクト指向データベース

- データベース＋オブジェクト指向
  - オブジェクト指向言語の研究が背景
    - オブジェクト指向言語の永続化
- オブジェクト指向データベースシステム宣言
  - by M. Atkinson, F. Banchilhon, D. DeWitt, K. Dittrich, F. Maier, S. Zdonik の 6 人により提唱 (1989)

OODB が持つべき要件（1）

- 複合オブジェクト（vs. 第一正規形）
- オブジェクト識別子（vs. タプル識別子）
- カプセル化（抽象データ型）
- 型/クラス（インスタンスとクラス）
- クラス階層（継承）
- 再定義、多重定義、遅延束縛（クラス階層下でのメソッド定義）

OODB が持つべき要件（2）

- 計算完備（計算可能なものは記述可能）
- 拡張可能性（基本の型から新しい型の生成）
- 永続性
- 二次記憶管理
- 同時実行制御
- 障害回復
- アドホックな問い合わせ機能

OODBの標準化

- 色々な記述形式が存在
  - OPAL (GemStone)
  - O-SQL (ONTOS)
  - CO_2 (O_2)
  - CODL (Bancilhon)
  - ...
- SQL-3 の中でも OODB の標準化を目指している

ODMG

- the Object Database Management Group
- Sun Mirco が 1991年に OODB ベンダに声をかけて設立したコンソーシアム
- ODMG 仕様
  - ODMG-93 1.0 (1993), 1.1 (1994), 1.2 (1995)
  - ODMG 2 (1997)
  - ODMG 3 (2000)
- 2001年解散

ODMG

- 仕様策定
  - オブジェクトモデル
  - オブジェクト定義言語 ODL (Object Definition Language)
  - オブジェクト問い合わせ言語 OQL (Object Query Language)
  - C++ バインディング
  - Smalltalk バインディング
  - Java バインディング

ODL によるオブジェクト定義の例 (メソッド、非正規形)

```
interface Movie
    (extent Movies
     key (title, year))
  { attribute string title;
    attribute integer year;
    attribute integer length;
    relationship Set<Star> stars
                    inverse Star::starredIn;
    float lengthInHours() raises(noLengthFound);
    starName(out Set<String>); }
```


ODL によるオブジェクト定義の例 (メソッド、非正規形)

```
interface Star
    (extent Stars
     key name)
  {
    attribute string name;
    attribute Struct Addr
            {string street, string city} address;
    relationship Set<Movie> starredIn
              inverse Movie::stars;
  }
```


OQL による問い合わせの例

```
myMovie = SELECT  m
          FROM    Movie m
          WHERE   m.title = "Gone With the Wind";
```

myMovie.stars: {Clark Gable, Vivien Leigh}
myMovie.length: 222 (min)
myMovie.lengthInHours(): 3.7 (hours)
myMovie.starName(myStars): no value
    myStars: {Clark Gable, Vivien Leigh}

OQL による問い合わせの例

```
SELECT  DISTINCT m
FROM    Movies m, m.stars s
WHERE   s.name = "Clark Gable";

SELECT  m
FROM    Movies m
WHERE   m.lengthInHours() > 2.5;

SELECT  DISTINCT s.address.city
FROM    Movie m, m.stars s
WHERE   m.title = "Gone With the Wind";
```

ODL によるオブジェクト定義の例(継承)

- 製品(Product) → PC → ラップトップ

```
interface Product
      (extent Products
       key model)
     {
       attribute integer model;
       attribute string manufacturer;
       attribute real price;
     }
```


ODL によるオブジェクト定義の例(継承)

```
interface PC : Product
      (extent PCs)
    { attribute integer speed;
      attribute integer ram;
      attribute integer hdd;}
interface Laptop : PC
    {
      attribute string screen;
    }
```

* Unit:
  - speed - GHz, ram – GB, hdd – GB, screen inch

ちょっと考えてみよう（11－1）

- 前出の Laptop の interface 定義に基づいて、'VEIO-10' という model で ram 64GB の Laptop のスクリーンサイズと価格を出力する問い合わせを OQL で記述せよ。

オブジェクトリレーショナルデータベース

- ORDB: Object-Relational Databases
- M. Stonebraker (UCB→MIT) が提唱
  - （狭義の）OODB のようにオブジェクト指向言語に強く密着しない
    - 永続プログラミング言語だけではない
- RDB をベースに、基本型の拡張、関数、継承の提供に限定
  - RDBの上位互換
    - OID の Join が可能
- 実例: PostgreSQL (cf. Ingres)

ORDB

- 大きな市場規模が期待できる
  - WWW のようなものを格納するには
    - RDB では柔軟性に欠ける
    - （狭義の）OODB の検索機能では不十分

	単純なデータ	複雑なデータ
問い合わせ機能なし	ファイルシステム	OODB 市場規模：1
問い合わせ機能あり	RDB 市場規模：100	ORDB 市場規模：150

ちょっと考えてみよう（11－2）

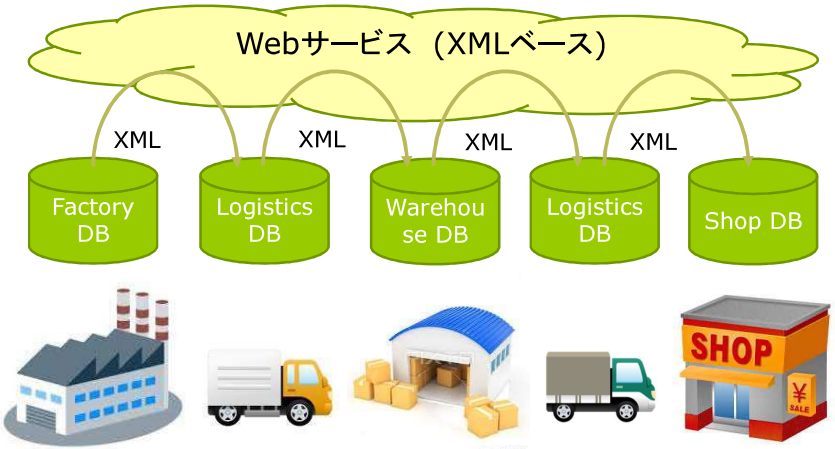
- ORDBはRDBの拡張としてオブジェクトを扱えるようにしたことで自由度を増しましたが、従来の正規化の理論が適用できません。なぜでしょうか。

XML

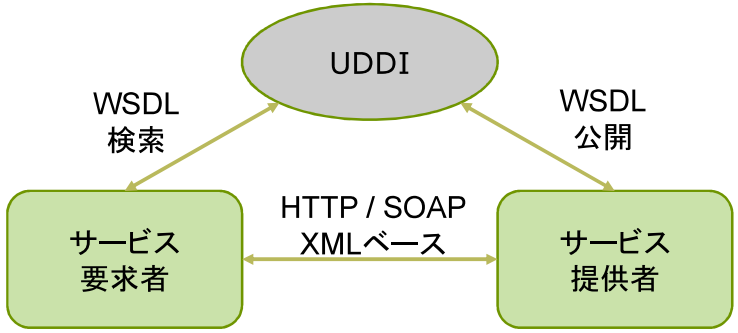
- Extensible Markup Language
  - 起源: SGML (Standard Generalized Markup Language) 1986
  - Web のHTML (Hyper Text Markup Language) と同様
  - W3C (World Wide Web Consortium) で規定（1998）
- 傾向
  - 近年、インターネットを経由して多くのデータがやり取り
    - 企業間（B2B）データのフォーマットとして多用
  - XML の形でデータを蓄積：XMLデータベース
    - 例
      - 論文データ: DBLP, ACM SIGMOD Record,
      - バイオ情報データ: TrEMBL, Swiss-Prot
      - …

B2B のイメージ

サプライチェーン・マネージメント



Web サービス

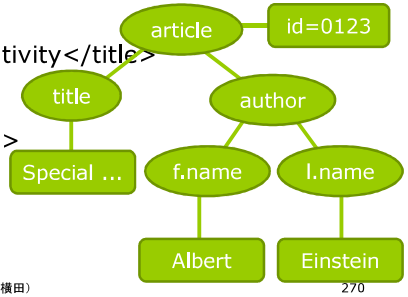


UDDI: Universal Description, Discovery and Integration
WSDL: Web Services Description Language
SOAP: Simple Object Access Protocol

XML基本シンタックス

- <start tag> と <end tag> で囲む
  - <title>Special Theory of Relativity</title>
- 階層構成 (属性も持つ)
  - NG:
    <author>Einstein<title></author>Relativity</title>
  - Well formed :

```
<article id="0123">
  <title>Special Theory of Relativity</title>
  <author>
    <f.name>Albert</f.name>
    <l.name>Einstein</l.name>
  </author>
</article>
```



- 木構造になる

DTD

- Document Type Definition
  - XML ドキュメントの要素作成ルール
- 例:

```
<!DOCTYPE journal-db [
  <!ELEMENT journal (article)+>
  <!ELEMENT article (title, authors)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT authors (author)+>
  <!ELEMENT author (f.name, l.name)>
  <!ELEMENT f.name (#PCDATA)>
  <!ELEMENT l.name (#PCDATA)>
  <!ATTLIST article id IDREFS #REQUIRED>
]>
```


RDB と XML-DB

- RDB:
  - テーブル:スキーマが固定
    - 全てのタプルが全属性を持つことを前提(空は null 値)
  - 各要素はアトミック
    - 第一正規形を前提
  - よく利用される問い合わせ言語: SQL
- XML-DB
  - 半構造:スキーマに自由度がある
  - 階層構造:要素としてさらに要素の階層を含める
  - 問い合わせ言語: Xpath, XQuery

XPath

- XML ドキュメント中から階層をたどりデータを探し出すための言語
  - XPath 1.0 1999 に W3C より開発
  - XPath 2.0 現在の版
- 1つ以上のノードとそれに含まれるデータを選択
  - 木構造の場所情報で表現
    - 軸(axis) :: node-test [predicate]
  - axis: child, descendant, parent, following-sibling, …
  - /child::articles/child::article/child::authors/child::author/child::l.name
- 簡略化
  - /articles/article/authors/author/l.name
  - /articles//l.name[Einstein]
  - //article[@id="0123"]

XQuery

- XML データのための問い合わせ言語
  - W3C により開発
  - XPath 2.0 の拡張
    - Xpath の Path 表現を利用
  - XML データの照会に合わせた出力整形の機能を持つ
    - 新たな XML ドキュメントを作成
    - SQL が新たなテーブルを作成するのに対応
- シンタックス
  - FLWOR
    - FOR, LET, WHERE, ORDER BY, RETURN

XQuery の FLWOR の例

```
for $x in doc("journal-db.xml")//article
let $a := $x/authors/author
where $x/year < 1930
order by $x//l.name descending
return
<articles-before-1930>
<title>{$x/title}</title>
<author>{$a/l.name, $a/f.name}</author>
</articles-before-1930>
```


出力イメージ

```
<articles-before-1930>
  <title>Special Theory of Relativity</title>
  <author>Einstein, Albert </author>
</articles-before-1930>
<articles-before-1930>
  <title>Uncertainty Principle</title>
  <author>Heizenberg, Werner</author>
</articles-before-1930>
```

ちょっと考えてみよう(11－3)

- XML形式で情報を蓄積することでRDBと比較してスキーマ設計の柔軟性が増しますが、集約演算等の対象には向きません。XMLDBはどんな用途に向いているか、向いていそうな具体的な用途を考えてみましょう。

XML データを格納する方法

- Native XML DB
  - XML の構造をそのまま蓄積
    - 例: Tamino
  - RDBMS で開発された多数の機能が利用できない
    - 含むトランザクション管理
- RDBMS を利用して格納
  - XML の一つのかたまりを文字列として RDB の要素として格納
    - DB2 等はこのアプローチ
    - XML の構造情報（親子関係等）は文字列を変換しないとわからない
      - RDB の検索機能だけでは構造情報は扱えない
  - XML の DTD の定義を RDB のスキーマに対応させて格納
    - B2B で転送される多くの情報は形式が一定
    - 半構造としての自由度は扱えない
      - 構造を自由に変更して中に階層構造を埋め込めない
  - XML の木構造の要素にラベルとつけて、要素をタプルとして格納
    - ラベルを工夫することで木構造の情報を使った検索が可能（変換は必要）

XML 格納の例

	Journal	
	<article id="0123"> <title>Special Theory of Relativity</title> <author> <f.name>Albert</f.name> <l.name>Einstein</l.name> </author> </article>	
	<article id="56789"> <title>Uncertainty Principle</tit <author> <f.name>Werner</f.name> <l.name>Heizenberg</l.name> </author> </article>	

<m.name>Karl</m.name>

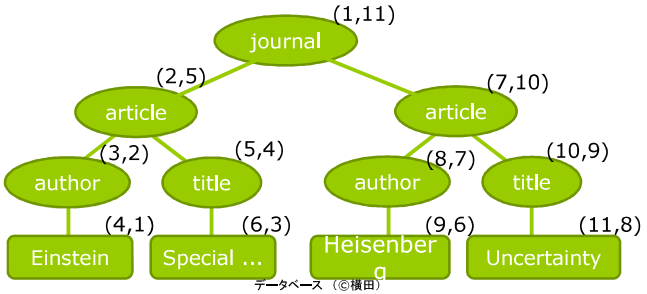
id	title	f.name	l.name	m.name?
01234	Special Theory of Relativity	Albert	Einstein	
56789	Uncertainty Principle	Werner	Heizenberg	

XML ラベリング手法

- 分割して格納した要素から、XML の構造情報を抽出して、再構築するためのラベル
  - XPath、XQuery、キーワード検索等、多くのアプリケーションで利用するための情報
- 導かれるべき情報
  - 包含関係（親子）、兄弟間の順番、ノードの深さ等
- アプローチ
  - 前順・後順（ Preorder-Postorder: PP）法
  - Dewey 順（Dewey Order: DO）法

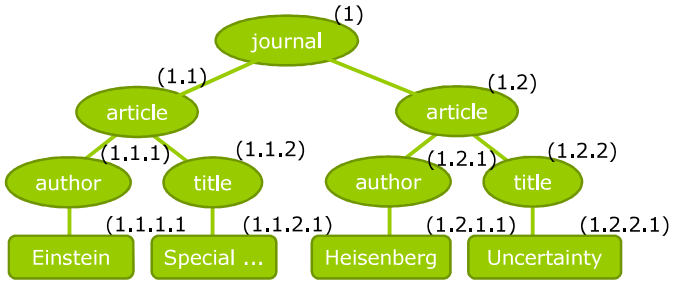
前順・後順（PP）法

- 全ノードに前順（根から始め、左の部分木を先に走査）の番号と、後順（一番左の部分木の左端から走査）の番号を付与
  - 子孫関係は、以下の式で判別
    祖先の前順番号＜子孫の前順番号　かつ
    祖先の後順番号＞子孫の後順番号



Dewey 順（DO）法

- ノードを（親のノード番号）.（子のノード番号）で表現
  - もともとは、図書館の書誌分類用のコード
    - 各分野の中でさらに小分類するために '.' を利用
    - 各分野、分類に10進数を割り当て



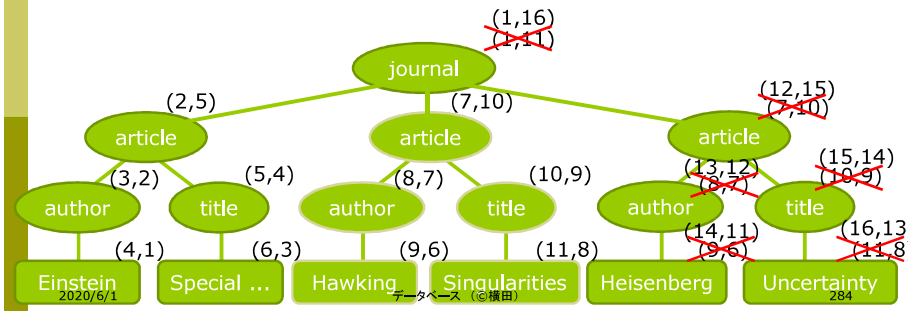
RDB へのラベル付ノード格納

Node	Pre	Post
journal	1	11
article	2	5
author	3	2
Einstein	4	1
title	5	4
Special …	6	3
article	7	10
author	8	7
Hesenberg	9	6
title	10	9
Uncertainty	11	8

Node	Dewey
journal	1
article	1.1
author	1.1.1
Einstein	1.1.1.1
title	1.1.2
Special …	1.1.2.1
article	1.2
author	1.2.1
Hesenberg	1.2.1.1
title	1.2.2
Uncertainty	1.2.2.1

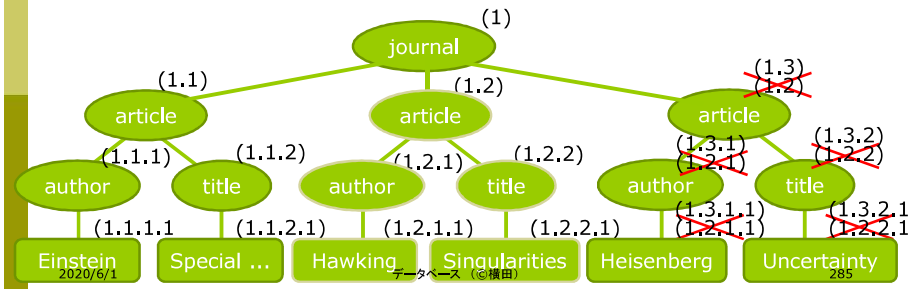
ノード（部分木）挿入時の問題 (PP)

- ノードあるいは部分木を途中に挿入しようとすると、その祖先と右側の部分木の番号を全て修正する必要がある
  - 連続した番号の利用を前提としているため



ノード(部分木)挿入時の問題(DO)

- DO の場合も同様の問題が発生
  - 挿入された部分木の祖先は変更の必要がないが、右側は変更する必要がある



問題解決方

- いくつかの方法が提案されている
- 数値に予めインターバルを入れる方法
  - SPARSE [Eda et al. 2002]
  - QRS (浮動小数点を利用) [Amagasa et al. 2002]
  - インターバルを使い切る可能性
- 制限の無い方法
  - DO-VLEI Code [ICDE 2005]
  - ORDPATH [O'Neil et al. 2004]

VLEI Code [ICDE2005]

- Variable Length Endless Insertable コード
- 定義
  - 1 で始まるビットシーケンス
  - 次の条件を満足: $1\cdot\{0|1\}^*$ $\quad v\cdot 0\cdot\{0|1\}^* < v < v\cdot 1\cdot\{0|1\}^*$
  - 例
    - 10<1<11
    - 100<10<101<1<110<11<111
- いかなる連続する VLEI コード v_l と v_r の間に新しいコード v を以下の式により作成することができる
  - IF $l(v_l) \le l(v_r)$ then $v = v_r \cdot 0$ else $v = v_l \cdot 1$
    ($l(v)$ is v's length)
  - 例: $1<11$ -> $1<110<11$

DO-VLEI [ICDE2005, IEICE2009]

- VLEI コードを DO 法に適用
  - 挿入に際し、他のラベルを変更する必要がない
- ドットも含めてバイナリーシーケンスに変更可能
  - ".1" → (11), "1" → (10), "0" → (0)
  - 例: "1.10.11" → (10 11 0 11 10)