

サービス基盤における 異種・分散データベース統合とその応用

産総研情報技術研究部門
サービスウェア研究グループ
小島 功

kojima@ni.aist.go.jp
<http://www.dbgrid.org>



概要

- **分散環境におけるデータベース処理とは？（基礎的内容）**
 - いわゆる、データグリッドにおけるデータベースアクセス
 - 応用の紹介・機能上の特徴など
- **ミドルウェアと標準の現状**
 - Middleware: OGSA-DAI/DQP AMGA, GReIC,,,,
 - Standard: WS-DAI 実装が本格化。
 - 異種データ統合の現状と課題
- **WS-DAIの概要**
 - OGF (Global Grid Forum)における標準化活動
- **OGSA-DAIの詳細**
 - 機能の説明
- **応用など。**

背景

- **情報の爆発的増加**
 - － 応用での情報量が増えている
 - 秒あたりGBの情報量を出すセンサー、
 - 100TBのCG映画(Pixar)
 - 膨大なネット上の情報など
 - － ディスクが安い
- **格納形式はさまざま**
 - － DB,ファイルなど
 - － XML,HTML,エクセルにいたるまで
- **分散環境でよい統合・連携の枠組みが必要となる**
 - － 分散環境⇒グリッドをはじめとするサービス基盤上
 - － データのよい統合・連携の枠組み？

データ・グリッドに代表される次世代のデータ基盤

グリッドなどの分散環境における、データ指向計算とは

– 大量のデータを扱う分散計算

- CERN のセンサデータの分散配布・共有・レプリカ管理
- 衛星データの配信など

– 地理的・組織的に分散した情報の統合計算

- IVO(仮想天文台)に基づくデータベース天文学
- 統合遺伝子データベース上のデータ解析・マイニングなど

これらの応用やインフラを広くデータ・グリッドと呼ぶ。

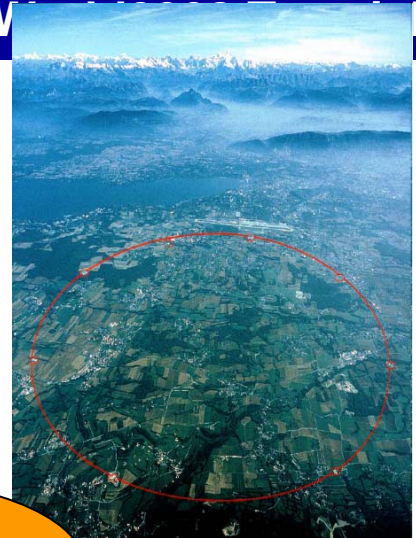
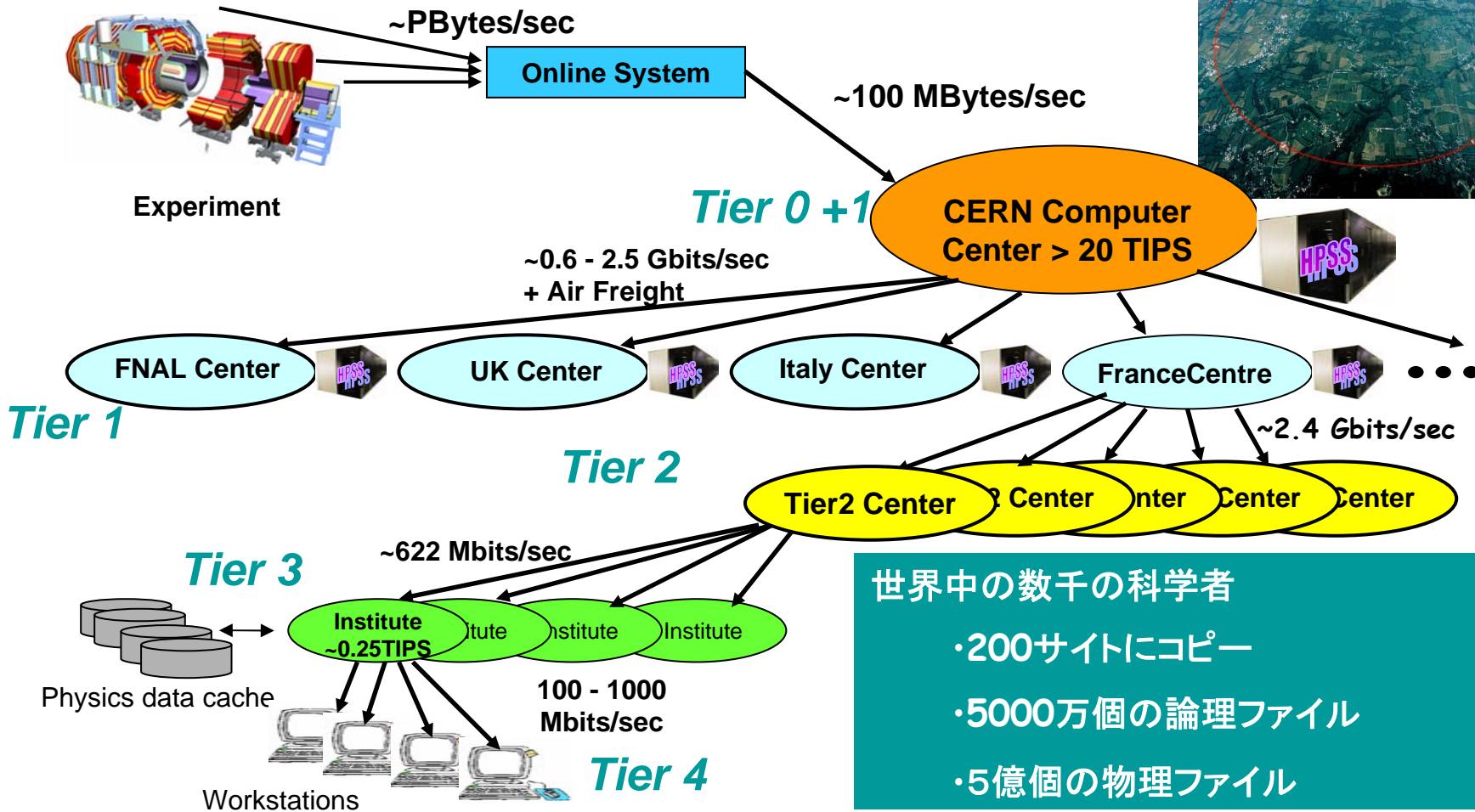
Webサービス＝サービス基盤上での構築が必須

データグリッド応用の種類 分散・大量データ処理(トップダウン的)

- CERN-LHC, 衛星データ, センサーネットワークなど
- **ひとつの情報リソースからのデータが膨大**
 - 単一のストレージに入らない
 - 同一タイプのデータソースが膨大にある
 - 分散した解析など、分散環境でデータを共有したい。
- **単一の管理ポリシーのデータをトップダウン的に分散させる技術。**

動機的な例 CERN LHC

100MB/s -> 360GB/H -> 8.64TB/Day -> 3PB/Year



(注: 2001 段階の図、、、2008年春に物理実験開始)

データグリッド応用の種類 情報統合(ボトムアップ的)

- 遺伝子データベースの統合と解析
- 仮想天文台など
- 分散した地域、組織などから、データがボトムアップ的に発生する。
 - 意味が似ているのに表現が違う。
 - 似たようなデータを持っている。
 - 組織独自のデータ管理ポリシーを維持したい
- 異なる組織などで別々に作られたデータを、仮想的にひとつのデータに統合する(情報統合)

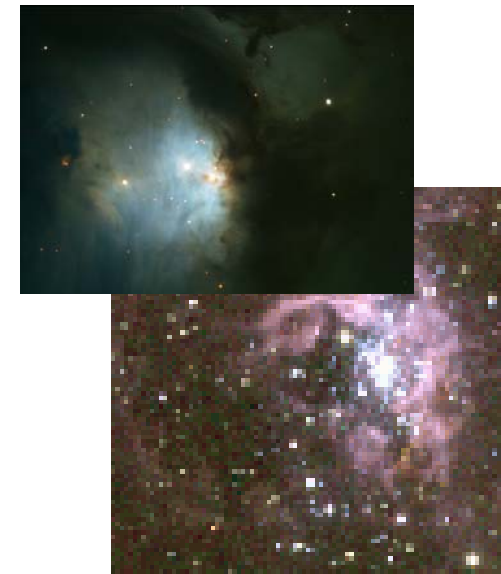
動機的な例2: 仮想天文台 & データベース天文学

望遠鏡データ: テラバイト〜ペタバイト

- **2MASS Project (2Micron All Sky Survey)**
 - 赤外線光での全天のサーベイ
 - 500万イメージ・10テラバイト
- **SDSS (Sloan Digital Sky Survey)**
 - 全天の4分の1 紫外〜近赤外
 - 40テラバイトの生データ
 - 600GBのカタログ
- 他、LIGO (重力波観測レーザー干渉計) など

Virtual Observatory

- 分散した複数のサーベイデータを統合して扱いたい。
- 多様な測定データを組み合わせて解析したい。
- 分散データ・マイニング
- 空間データ検索
(近傍のオブジェクトとか、移動オブジェクトの検索とか)



どう分散を吸収し、仮想化するか？

分散の吸収: 仮想化のレベルや応用が違う。(組み合わせることも可能)

- 分散ファイルシステム
 - Gfarmなど、単一の管理ドメインを広げる応用などに向く。
- 分散データベース
 - OGSA-DAIなど 特に、情報統合的な側面を持つ応用に有益

なぜか？

データの抽象表現 (テーブル) => 異なる情報や応用をつなぐための共通表現
物理表現/構造に独立 => 環境の違いを吸収できる。

抽象的なデータ表現(テーブル)によるデータ管理は、応用依存性が少なく有益。

★プログラム言語に独立、宣言的な検索言語(SQL等)は大量データ処理に有益

★ トランザクション処理、データの一貫性の保持、ユーザの権限管理によるデータ保護

その他

- Gridftp、ftpなどデータ転送ツール
- レプリカ管理など、分散に伴うデータの一貫性保持ツール

これらの機能群、ツール群を、よく連携させること。

目標と現実

- 目標:

- 異種のデータを無理なく統合したい。
 - 関係DBとXMLと、、、
- 製品の相違性を吸収したい
 - SQLの文法などの相違

- 現実:

- 遠隔アクセス、応用によるデータ統合は可能
 - 規格とそれを満足するミドルウェア
 - その上でのデータ処理プログラミング
- 汎用の異種データ統合などは実験研究段階

- サービスベースの分散データベースアクセス
 - Webサービスの規格を通して遠隔のデータベースをアクセスする。
RDB,XMLDB、OODBなど。

Webサービス経由のJDBCとかでいいのでは？

NO !

- 検索などデータの量が大量であること。
 - 結果をファイル転送(FTP)などで戻したい。
- 複数のサイトで分散処理
 - 分散問い合わせやデータ統合の実装
 - 第3者転送。(検索結果を別のサイトに送るなど)

サービス基盤におけるデータベース処理

- 単なる結果データの転送だけでないデータ処理
 - 統合を応用レベルで実現するための良いプログラミングの枠組み
 - 転送のついでにフォーマット変換とかしたいが、いちいちファイルに書くのは嫌だ。
 - 大量なので圧縮して戻したい。
- グリッドの他のツールとの親和性
 - GSI(グリッドにおける標準の認証)の支援: 計算処理との連携・融合
 - 管理ツールやインターフェイスなどのサービスレベルの互換性
- 異なるデータリソースの扱いを統合・連携したい
 - ファイル処理
 - WebDBなど

これらを満足するDB規格とミドルウェアがほしい

グリッド(OGSA)における規格とミドルウェア

- **OGSA(Open Grid Service Architecture)**
 - サービスベースのグリッドのアーキテクチャ
 - OGF(Open Grid Forum)で策定
- **OGSAに基づくデータベースアクセス規格**
 - **WS-DAI**
 - Web Service Database Access and Integration
 - OGF DAIS Working Groupで策定
 - OASISやW3Cにおける、WS-XXに対応
- **WSRF(GT4)に基づく、データベースミドルウェア**
 - **OGSA-DAI**
 - GT4(など)上でデータベースをアクセスするためのミドルウェア
- **特徴:この2つが密に連携している。**
 - WGのメンバーが開発に関与。
 - 開発のフィードバックを仕様へ反映、
 - OGSA-DAIがDAIS仕様の参照実装になる、などなど。
- **標準化活動との密接な関連 = 使える標準へ**

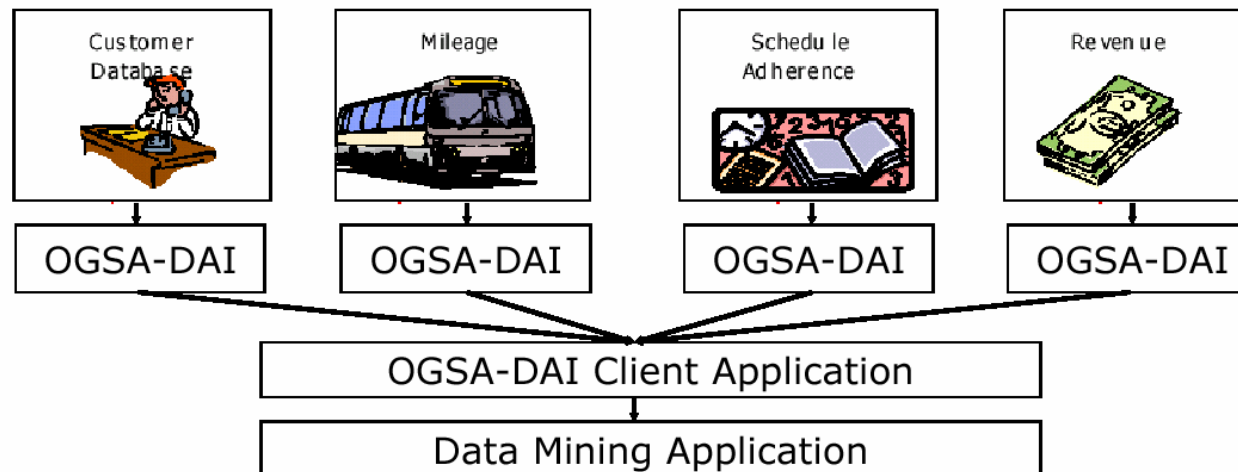
サービス基盤上のDBミドルウェア

- 現状：
 - グリッドの規格やミドルウェアと接続できるものが増加中。
- 相互接続性
 - OGFのWS-DAI規格を満足しつつある。
 - 特に関係データベース
- いろいろな課題
 - プロダクトの相違性吸収
 - 異種データ統合

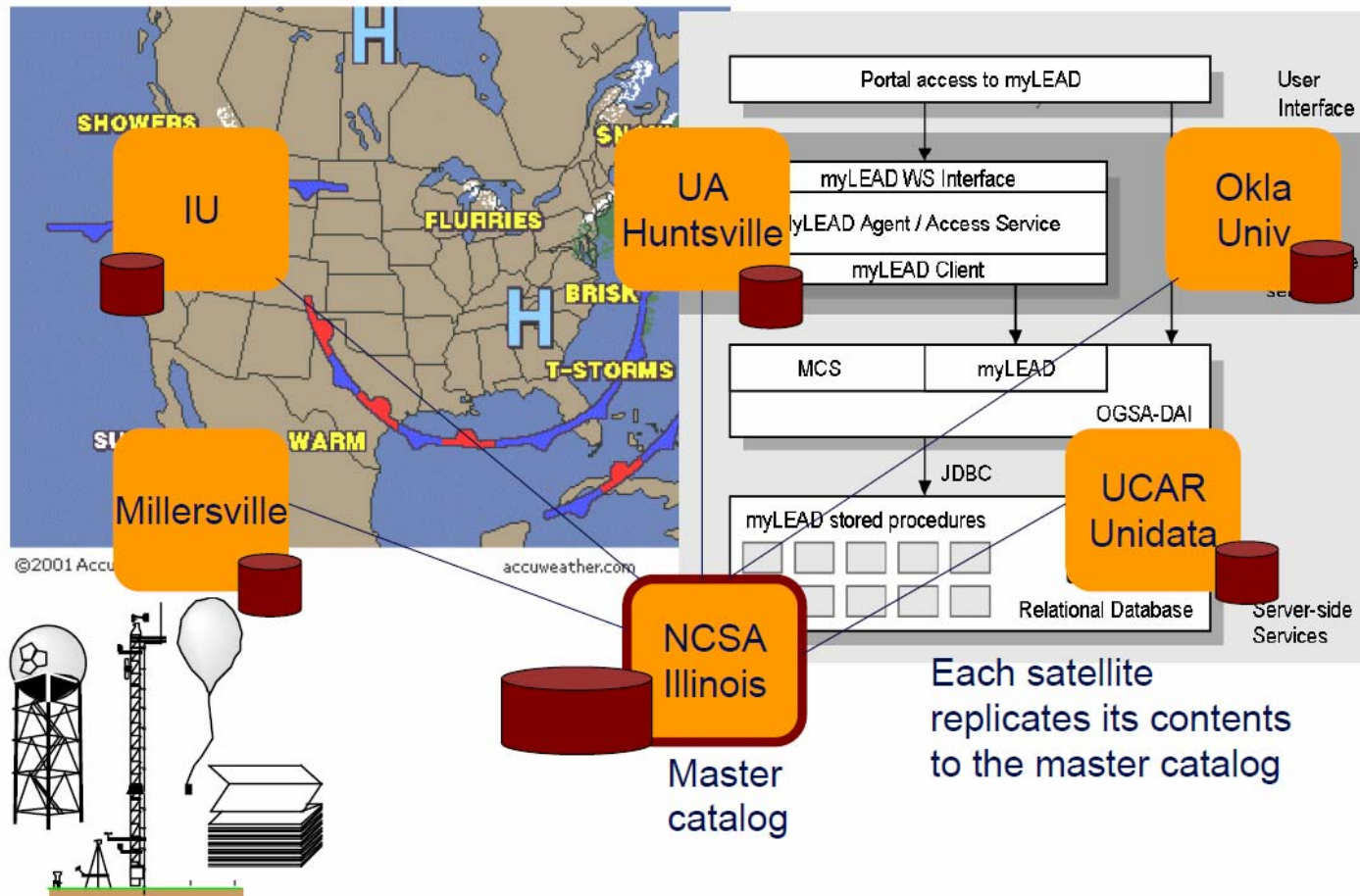
データ統合・分散データベースの応用

● 例: FirstDIG

- 商用(交通サービス、FirstTransportGroup)利用でのOGSA-DAI
- 複数のデータベースにまたがったデータマイニング
- バスが10分以上遅れると82%の確率で収入が10%下がる、とか



LEAD: OGSA-DAIによる気象衛星データの管理



最近の応用: BEinGRID

- Business Experience in Grid
 - EUのプロジェクト
 - ビジネスに対するグリッドの応用
 - 18個の実験プロジェクト
 - データベース系もいくつかある



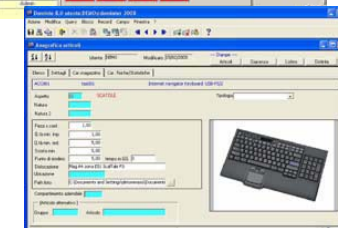
BE12: Sales Management System

- ENEA (Italian National Agency for New Technologies, Energy & Environment)
- CINECA (Italian supercomputing centre)
- Technocassa (cash register)
- Pizza New Spa (pizza take-away service)
- Domino Italia (business process management)

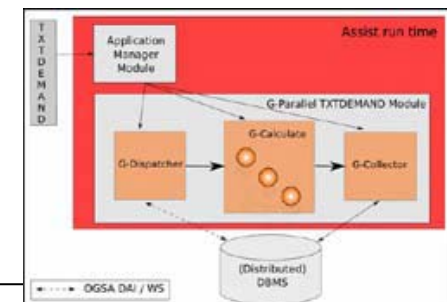


ピザチェーンにおける、売上管理の分散データベース

- データベース連携の実現: 主に関係データベース
 - DBアクセスの標準層の設定によるアプリケーションの効率的な構築
- セキュリティ(Gridの)によるデータの保護



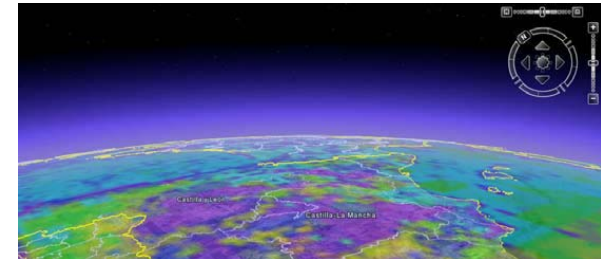
- BE5 (リテール管理) などでもデータベース統合が



BE7:Earth Observation

BE18: Seismic Simulation

- 地球観測衛星のデータ統合
 - 異種のデータの重ね合わせ
 - 複数の衛星
- 地質データのシミュレーションとデータ管理
 - 計算とデータの接続
 - 複数コミュニティでのデータ管理



★ 応用の特徴

- 計算と分散データベース処理の連携
 - 例: 分散遺伝子DBを統合してクラスタ上でBlastにかけるとか。
 - Grid のミドルウェアに基づいている (GT4, OMII) ことが有利
- セキュリティ基盤との連携
 - GSIに基づく、グリッドの認証
 - VOMS ?
- 第3者データ転送、大量データ転送
 - 大量の検索結果がクライアントに直接戻るとまずい
 - 検索結果を別のサイトに送ったり、
 - GridFTPなどと結びつけたり、

WS-DAI 規格

(Database Access and Integration Service)

DAIS: Database Access and Integration Service

- DAIS-WG: OGFにおける、データベースアクセス標準の策定のためのワーキンググループ
 - WS-DAI: DB実装に非依存なデータベースアクセス規格
 - WS-DAIR: 関係データベースの接続規格
 - WS-DAIX: XMLDBの接続規格

規格のドキュメントとしては終了。

現在は互換性テストのための実装待ち。各2つづつは必要。

XMLが1つ終了(OGSA-DAIグループ)

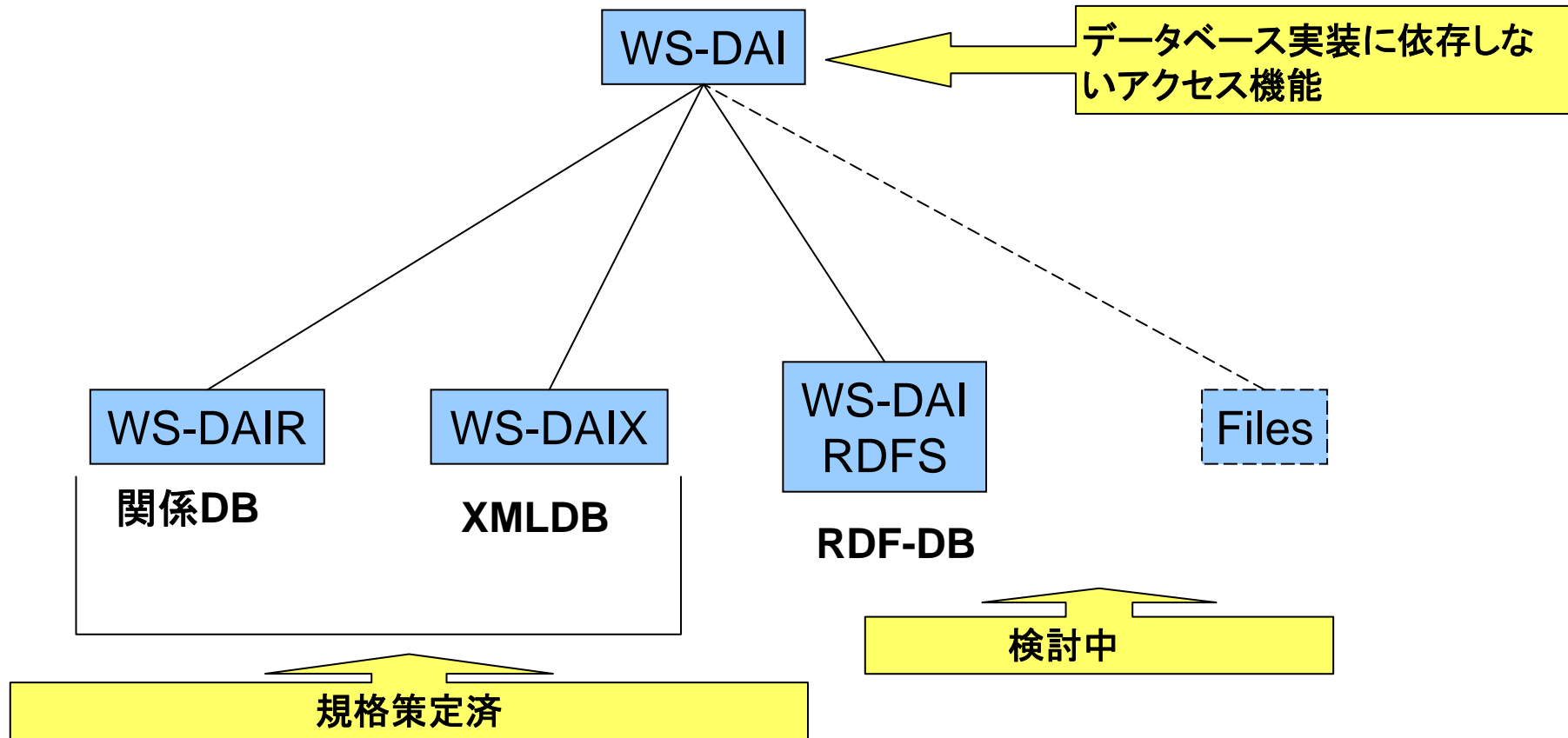
RDBが1つ終了(AMGA)

続々と増えている状況(特にRDB、後述) = 使える規格へ

WS-DAIの規格の動機と特徴

- Webサービスに基づいたデータベースアクセス規格
 - SQLを渡すにしても標準化されたインターフェイスが必要、、、
 - 意外とない。
- いろいろなデータベースをなるべく統一的に扱いたい
 - RDB, XMLDB, RDF, ファイル、などなど、、
- 大量データを扱えるようにしたい。
 - 検索結果が直接クライアントに戻っては困る場合がある。

DAIS 規格の構造

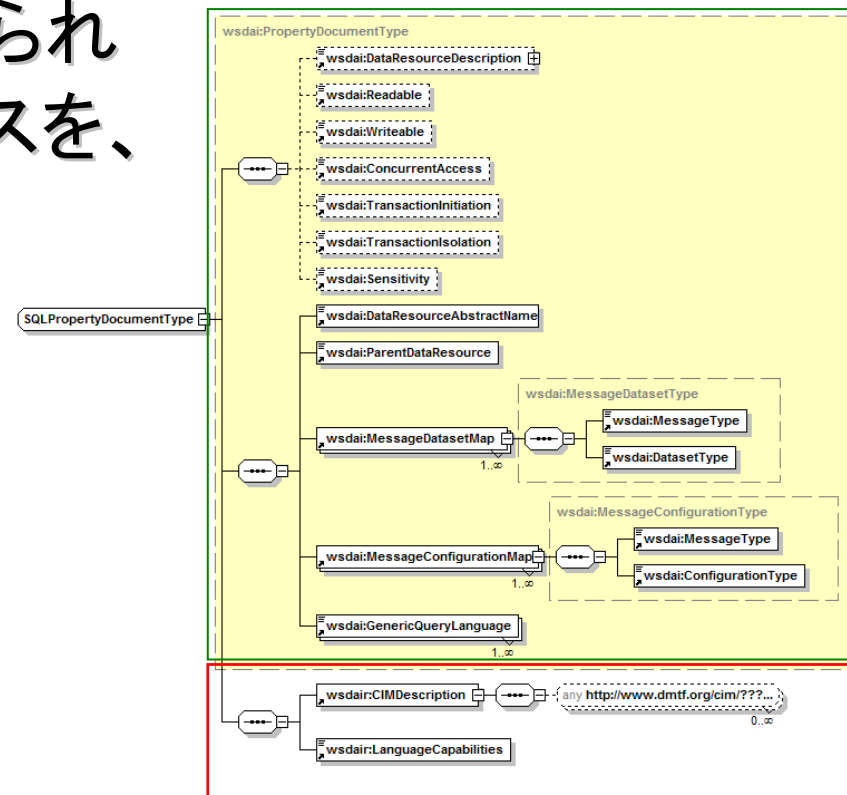


WS-DAIと WS-DAIR,X など下位規格との関係

- 基本的に、WS-DAIで定められたメッセージインターフェイスを、

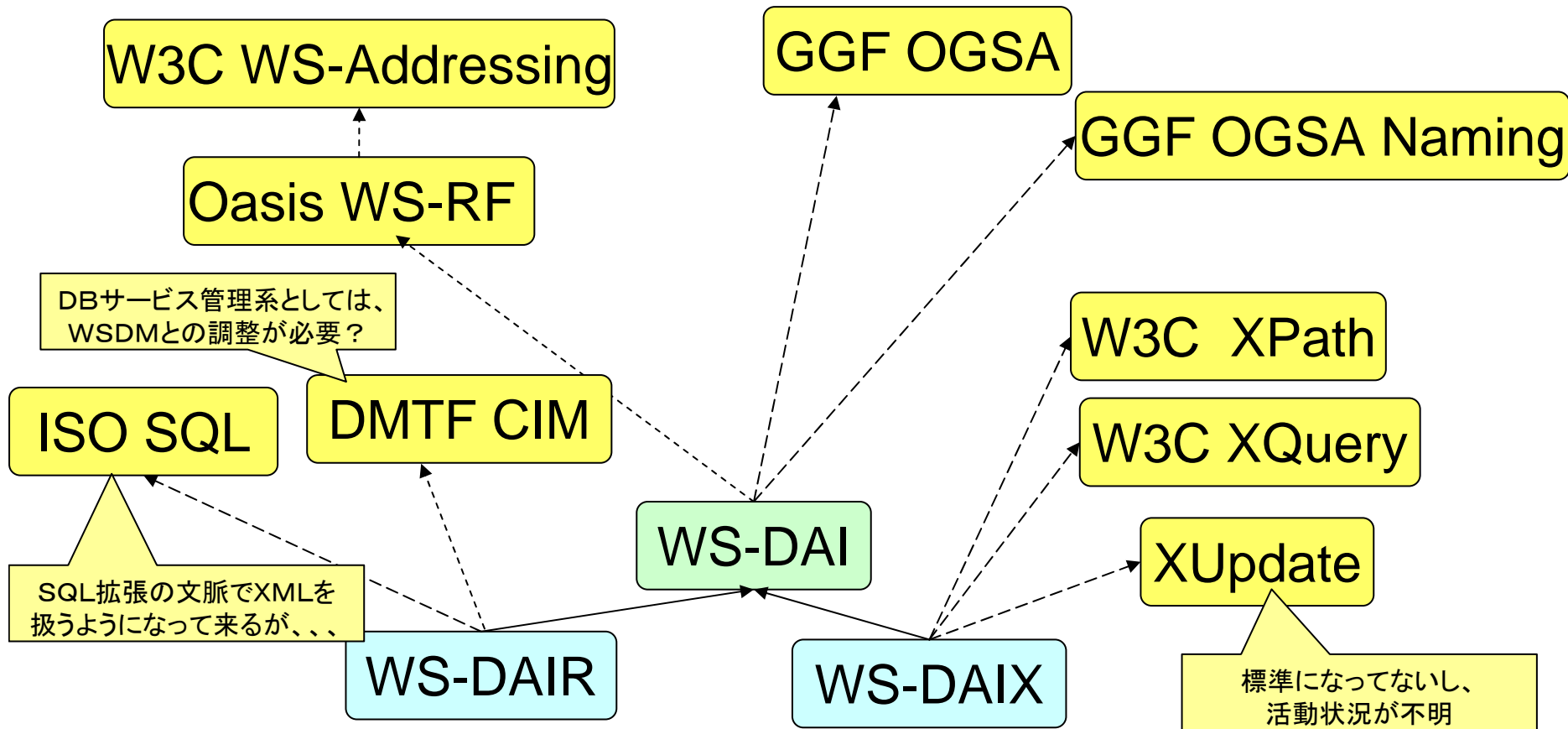
個々のDB実装に対応して
拡張あるいは詳細化する。

WS-DAIで決まる規格



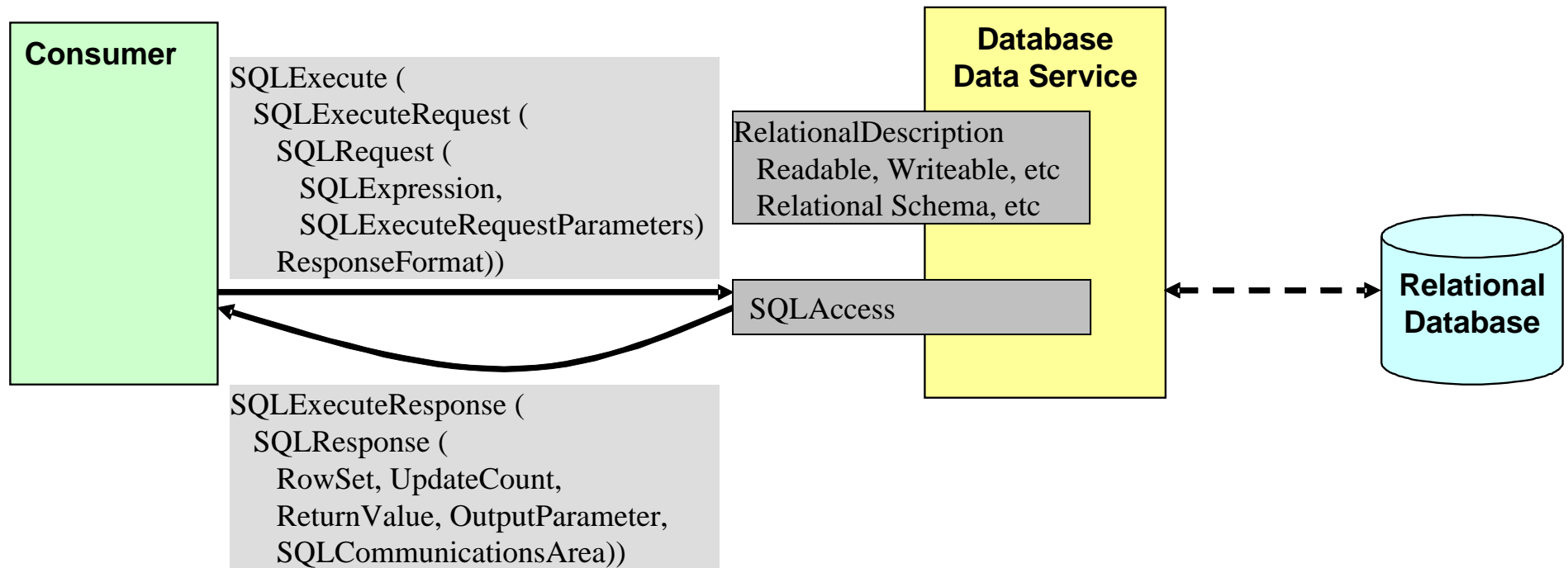
WS-DAIRなど、下位の規格の拡張

関連する規格

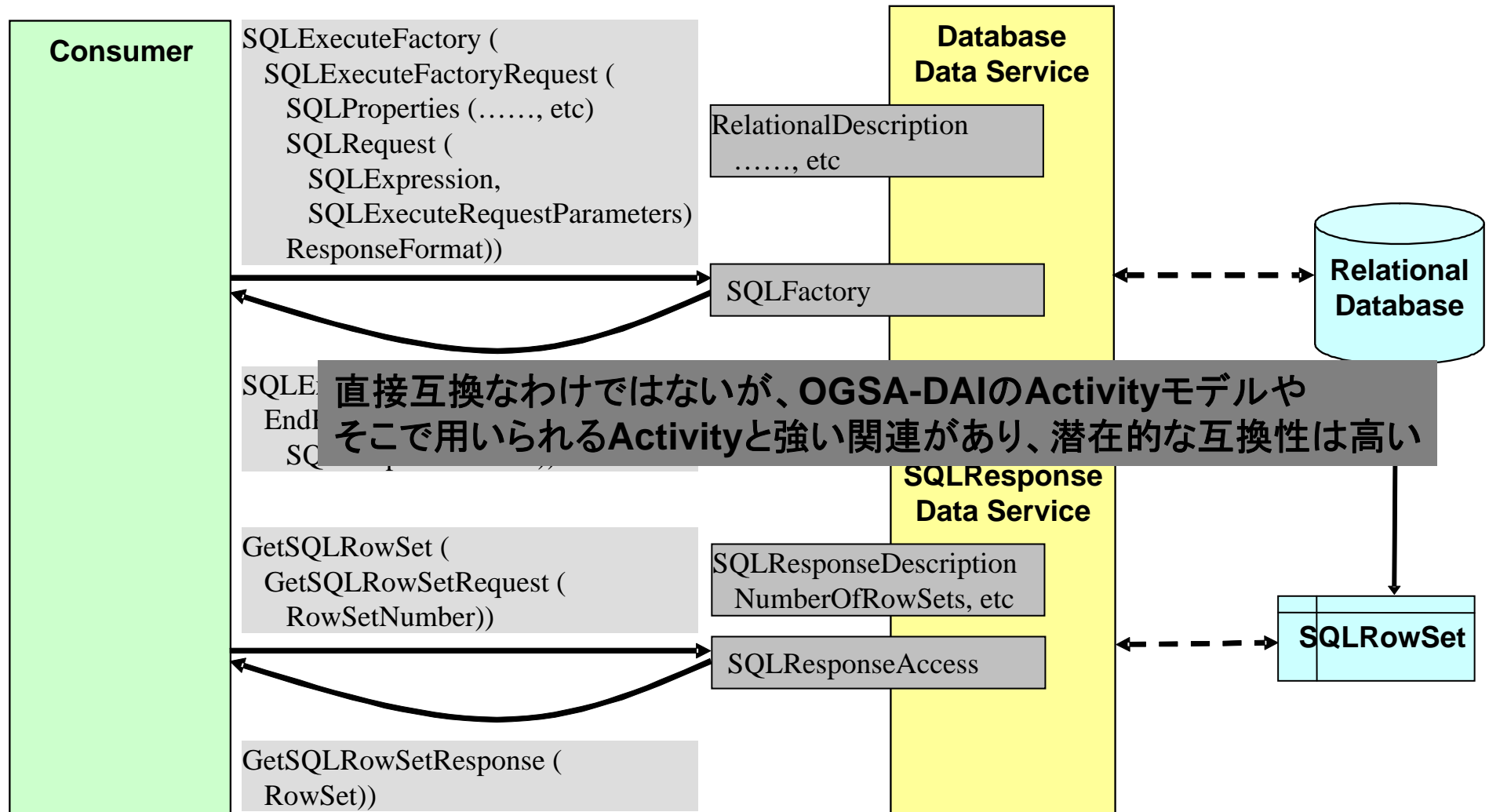


DAISアクセスパターン1: 直接配送

普通のクライアントサーバにおけるSQL処理



DAIS 2:間接配送:データ量が多いときなど、検索結果を集合として、別のサービスから提供する



★ DAISのモデルの特徴(まとめ)

- 結果の保持と別サービスによる提供
 - 大量の結果データへの対応
 - 第三者転送への対応
- 複数のデータベース(RDB, XML等)に対応する共通モデル。
 - どこに何があるかとかは、知りやすい。

WS-DAI規格の拡張性

- WS-DAIR(Relational), DAIX(XML)は、コアなWS-DAI規格の拡張である。
- 他のデータベースについても容易に拡張できる。
 - Files:
 - Object指向DB: ラフな提案が出たが活動中断状態
 - 標準言語の欠如、利用人口やマーケットの小ささなど。
 - RDF DB: 産総研とEu Ontogridが共同で提案/作業中
 - W3C SPARQLなどをベースにグリッド向け拡張
 - Ontology Primitivesの提供
 - ユースケース & ボランティア募集中

OGSA – DAI

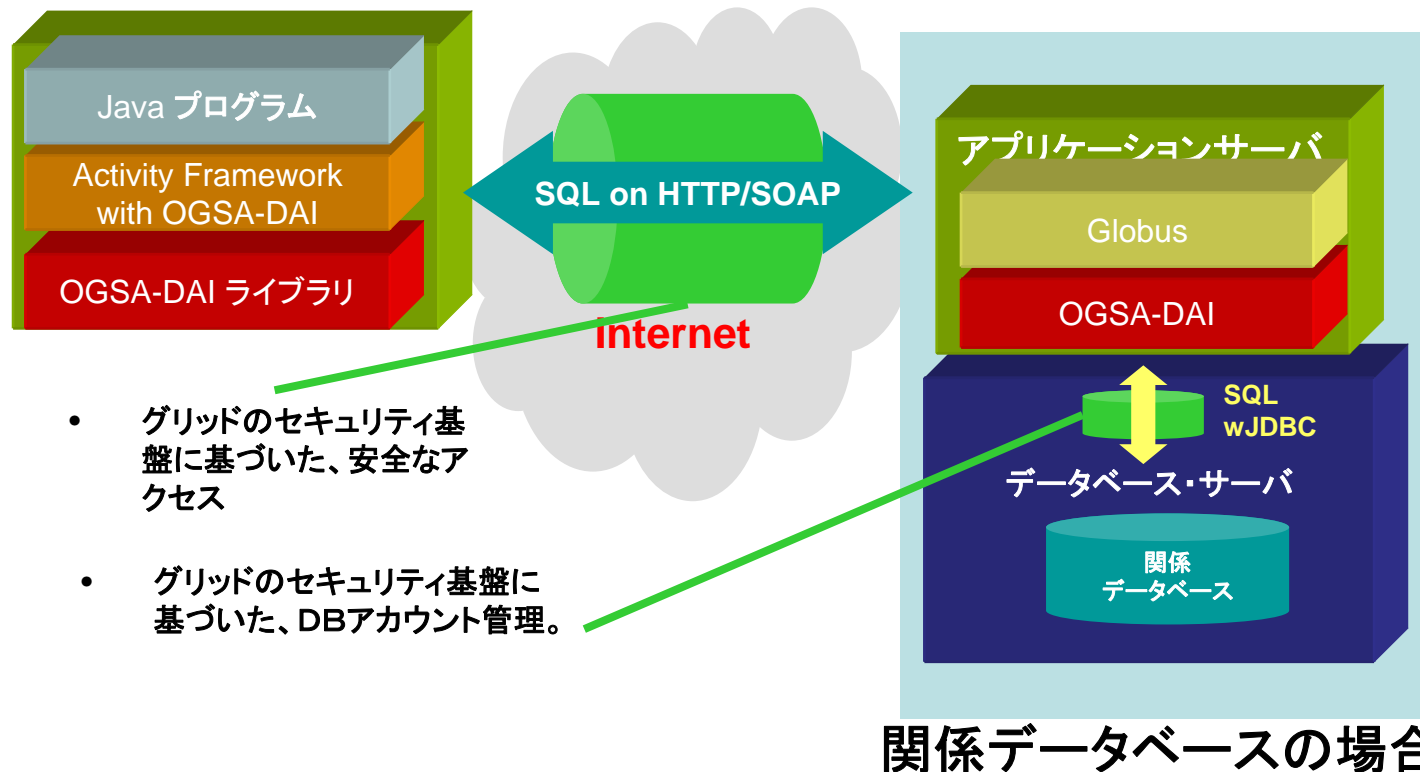
OGSA-DAIとは

- Webサービスを基礎とした、データベースアクセスのミドルウェア
 - OGSA(Open Grid Service Architecture)
 - DAI(Database Access and Integration)

単純には、リモートのDBをWebサービス経由でアクセスする
ミドルウェア(もちろんそれだけではないが)

- 英国 OMII (Open Middleware Infrastructure Institute)の1プロジェクト
(<http://www.omii.ac.uk>)
- OMII グリッドのミドルウェア
 - OGSA-DAI データベース
 - myGrid/Taverna セマンティックWeb・ワークフロー

- リモートのDBをWebサービス経由でアクセスするミドルウェア
(もちろんそれだけではないが)
 - ファイアーウォールの内側のDBMSにSQLなどで直接アクセスできる



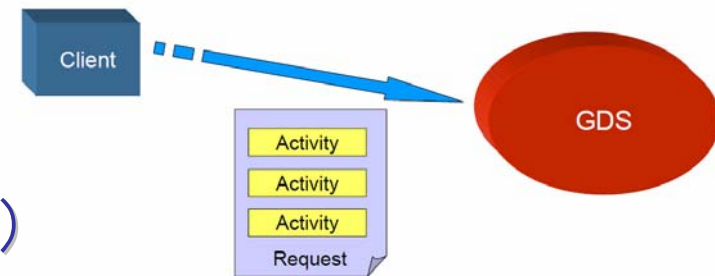
- グリッドのセキュリティ基盤に基づいた、安全なアクセス
- グリッドのセキュリティ基盤に基づいた、DBアカウント管理。

- 「リモートのDBをアクセス」=データベース統合がそのままできるわけではない。
 - 分散SQLとかが実装されているわけではない(関連ソフト:OGSA-DQPを使う)
 - その代わりに、分散データ統合の処理を書くための基本部品は揃っている(後述)

続き

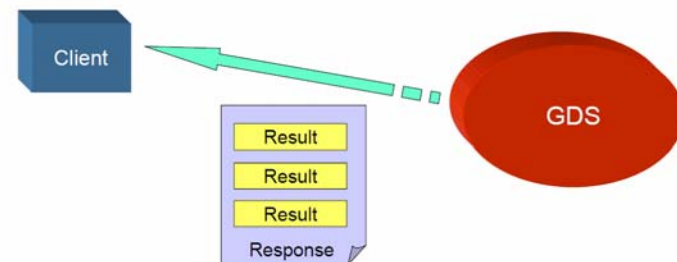
●リモート・遠隔地のデータベースシステムが、Webサービス(データ要求実行サービス)として提供され、グリッドのミドルウェア(Globus Toolkit)上のプログラム(client)から使える。

- 一般的なデータベースプログラミング
(プログラム内からSQLなどを投げる)



●アクティビティという処理モデル(後述)

- 一連の処理をまとめて投げ、
- 処理結果をまとめてもらう。



いいところ

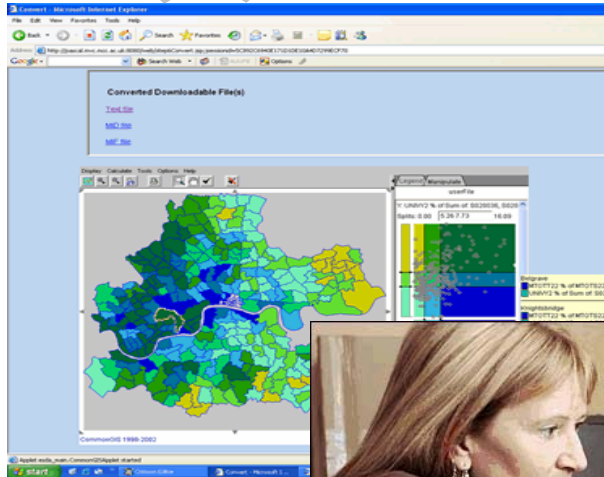
単なるSQLのリモートアクセスとどう違うか？

- サービスベース
 - HTTPポートの開いているところでは、どこでもOK
- GSIをサポート
 - GSIとDB(例えばOracle)のアカウントのマッピング
- 結果のデータ変換をサポート
 - XSLTを支援、検索結果の加工など。
- 第3者転送・大量データ処理をサポート
 - FTP・GFTPなどにより、結果を第3者サイトに転送できる

サポート・プラットフォーム

- Webサービス基盤
 - Globus Toolkit 4
 - Axis
 - OMII
- データベース
 - 関係DB
 - MySQL, PostgreSQL, Oracle, DB2, SQLServer
 - XMLDB
 - eXist, Xindice,

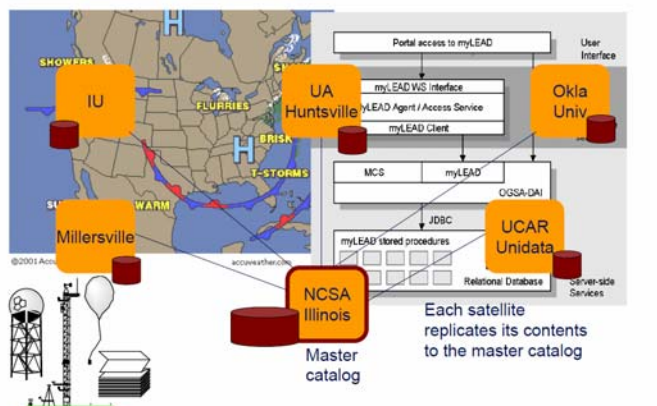
アプリケーション (Success Story等より)



Uk-e-social science:
分散した社会科学データベースの統合環境
異組織のもつ異なるDBプロダクトの統合



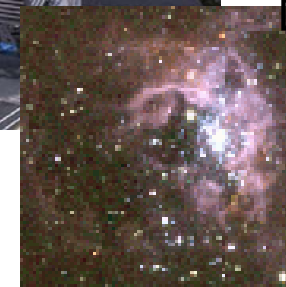
eDiamond:
分散した医療情報・X線画像
DBの統合



LEAD:
全米規模での気象情報の分散DB統合

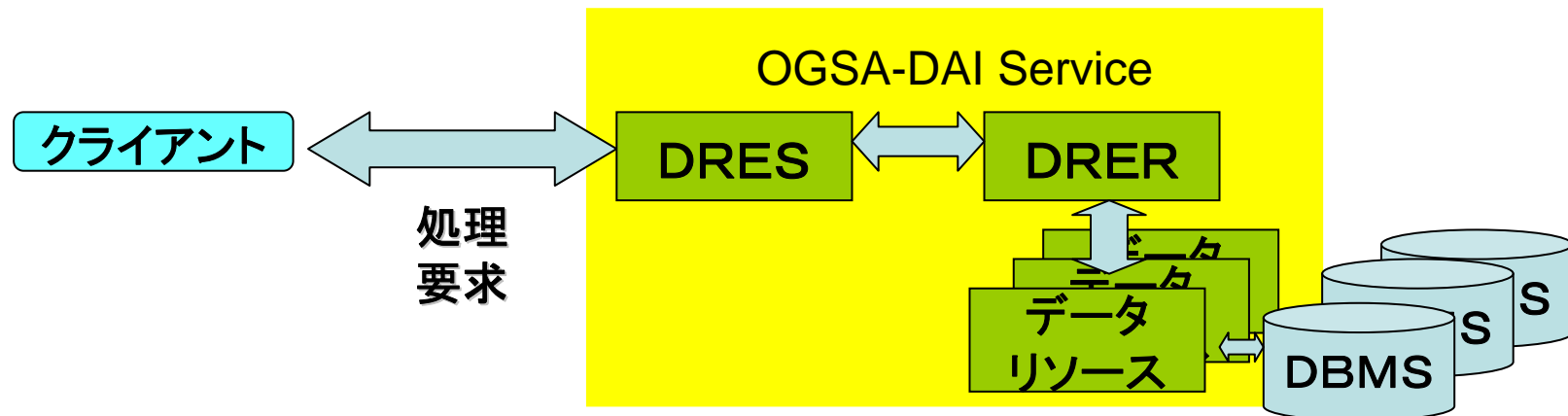


AstroGrid:
組織、地理的に
分散した望遠鏡画像
DBの統合。



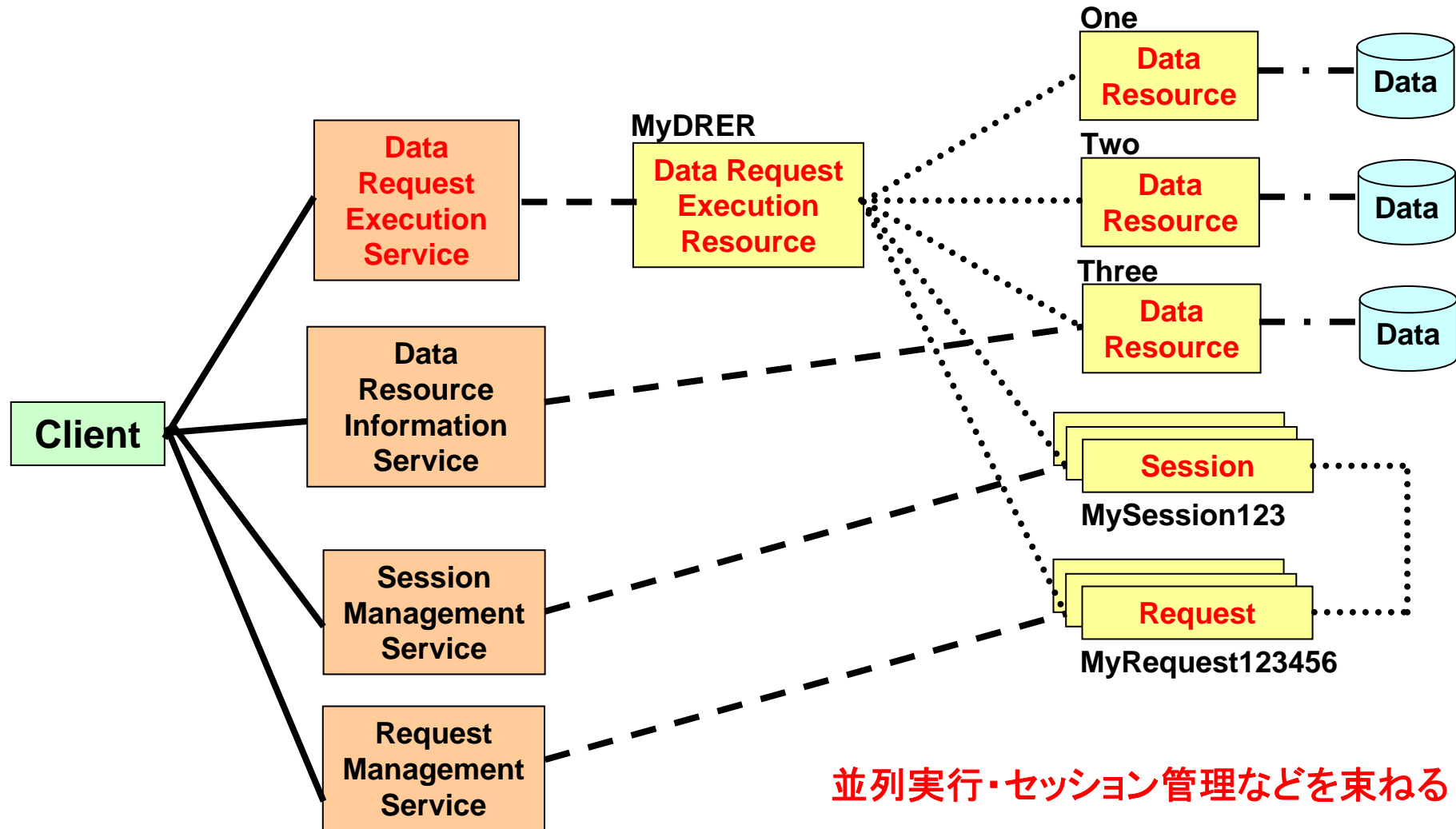
OGSA-DAIの実行モデル

- データ要求実行サービス(Data request execution service)
 - Webサービス。
 - サービスとしては、このサービスだけが見える。
- データ要求実行リソース(Data request execution resource)
 - 要求実行サービスが見せる唯一のリソース。要求を実行するリソース。
 - 実際には、その下にデータリソース(複数)が接続されている。



- Webサービスへなげる処理要求はSQL(とかXQueryとか)か？
- そうではない。

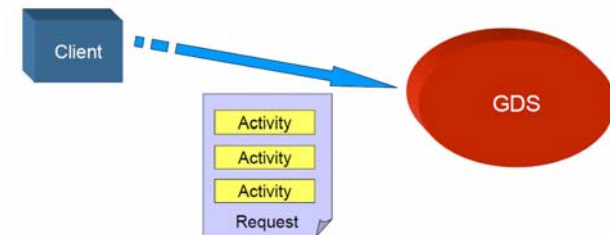
要求実行サービスの位置づけ



並列実行・セッション管理などを束ねる

OGSA-DAIにおける処理要求

- **Activity & Workflow**
- **Activityとは？:OGSA-DAIにおける基本実行要素**
 - たとえば、SQLの問い合わせ1つとか。
 - 検索結果のファイルへの書き込み、とか
 - データの圧縮とか、
 - データの転送(FTP処理)とか、
 - これらを**Activity**という単位で実装している。



OGSA-DAI=たくさんのアクティビティの集まり

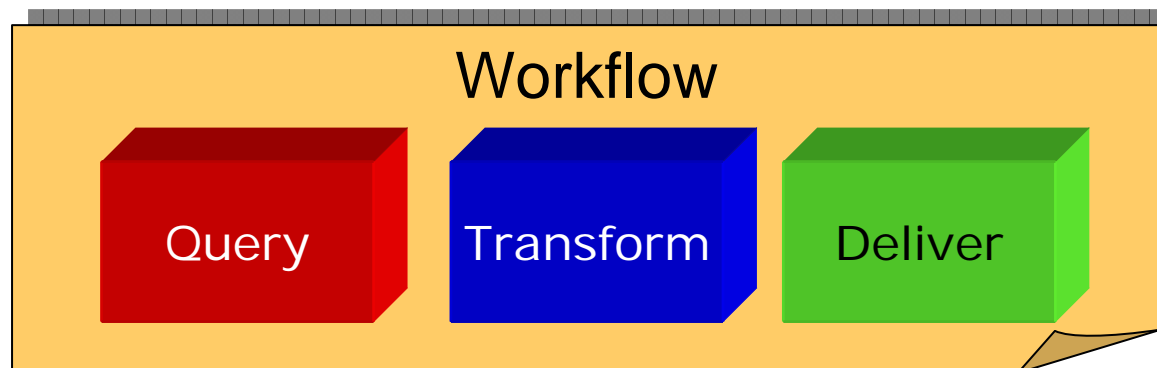
- データベース検索(SQL、XQuery)だけでなく、その周辺のデータ処理の集まったアプリケーションの構築環境である。
- ActivityそのものはWebサービスではない。

OGSA-DAIのWorkflowとは

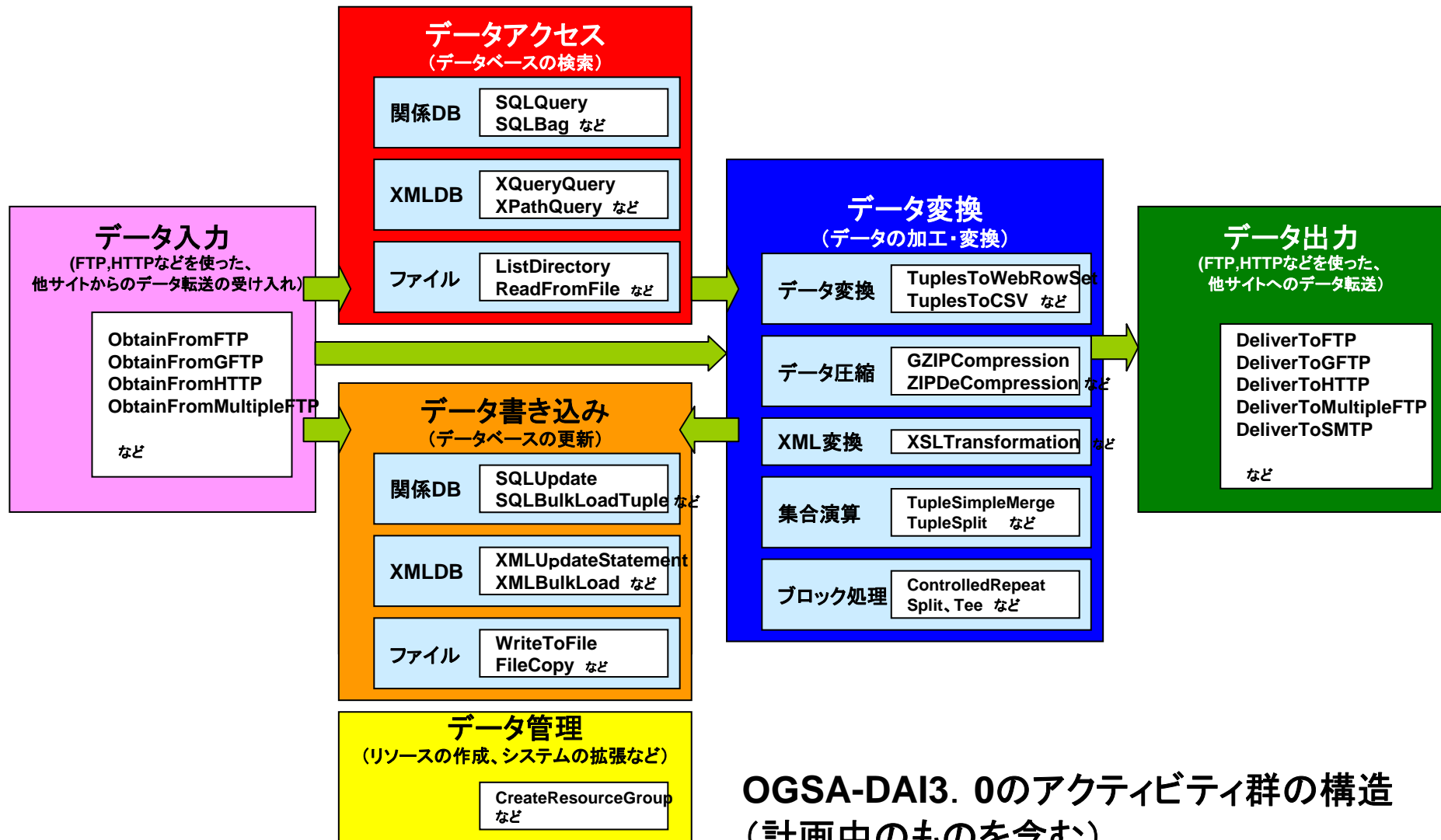
- Workflow: 複数のActivityがつながって、一連のデータ処理を記述するもの



1. SQLで検索をして、(SQL問い合わせのアクティビティ)
2. 結果をCSVに変換して、(データ変換のアクティビティ)
3. そのデータをリモートに転送しよう、



WorkflowがWebサービス(要求実行サービス)に対する一つの呼び出しになっている。



何でこんな構造なのか？

- 一つのサイトでやるデータ処理は、問い合わせだけではない。
 - 一般には、加工して、別のサイトに転送して、、、といった一連の処理が必要。
- これら小さな処理の連携・接続をWebサービス同士の連携でやるのは非効率的
 - 一つのサイトやコンピュータの中なら、もっと処理同士の連携は簡便かつ効率的にできるはず。
 - データ処理の単位はActivityとして定め、Activityが組合わさったワークフローをWebサービスの入力とする。
 - Activity間の連携は、パイプ・ストリームのような簡便かつ効果的な実装を使う。

1. 一つのデータサービス内でのワークフロー

- Activityの連携でワークフローを記述、処理
 - ひとつのサービス内でできることを高度化

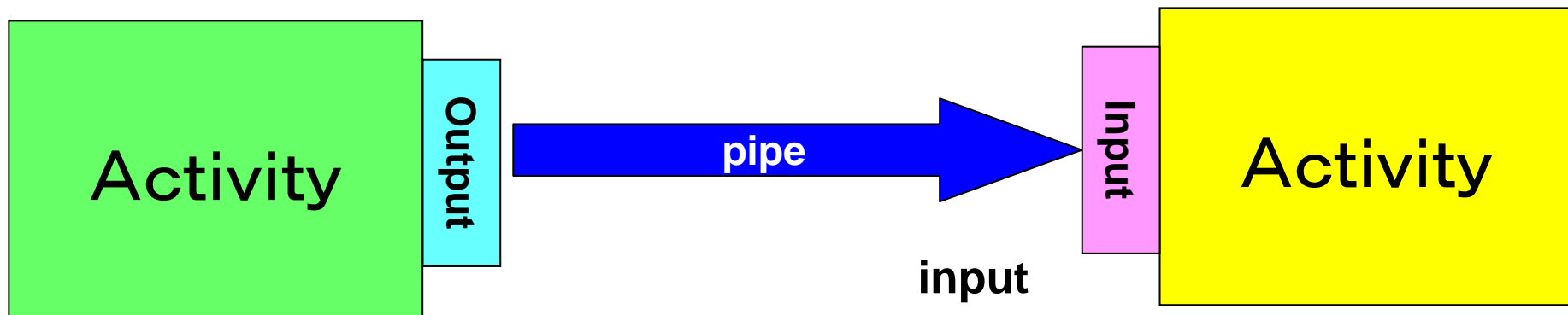
2. 複数のサービスにおける連携

- データベース処理を含む汎用ワークフロー
 - BPELやTavernaなど、汎用ワークフローエンジンと組み合わせよ。
 - 該当モジュールの提供
 - WEEPなどDAI向きBPELエンジンの提供
- 分散データベース処理に基づくワークフロー
 - 分散問い合わせ処理のミドルウェア+ α の提供
 - OGSA-DQP
 - 問い合わせ処理を最適化する。

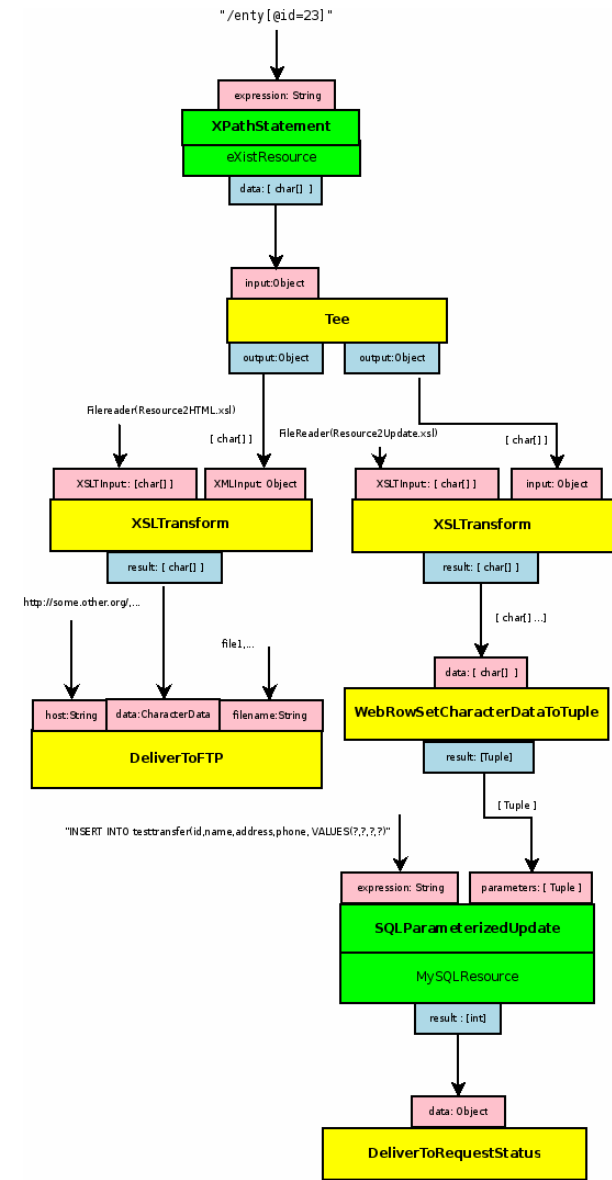
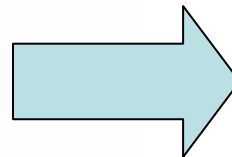
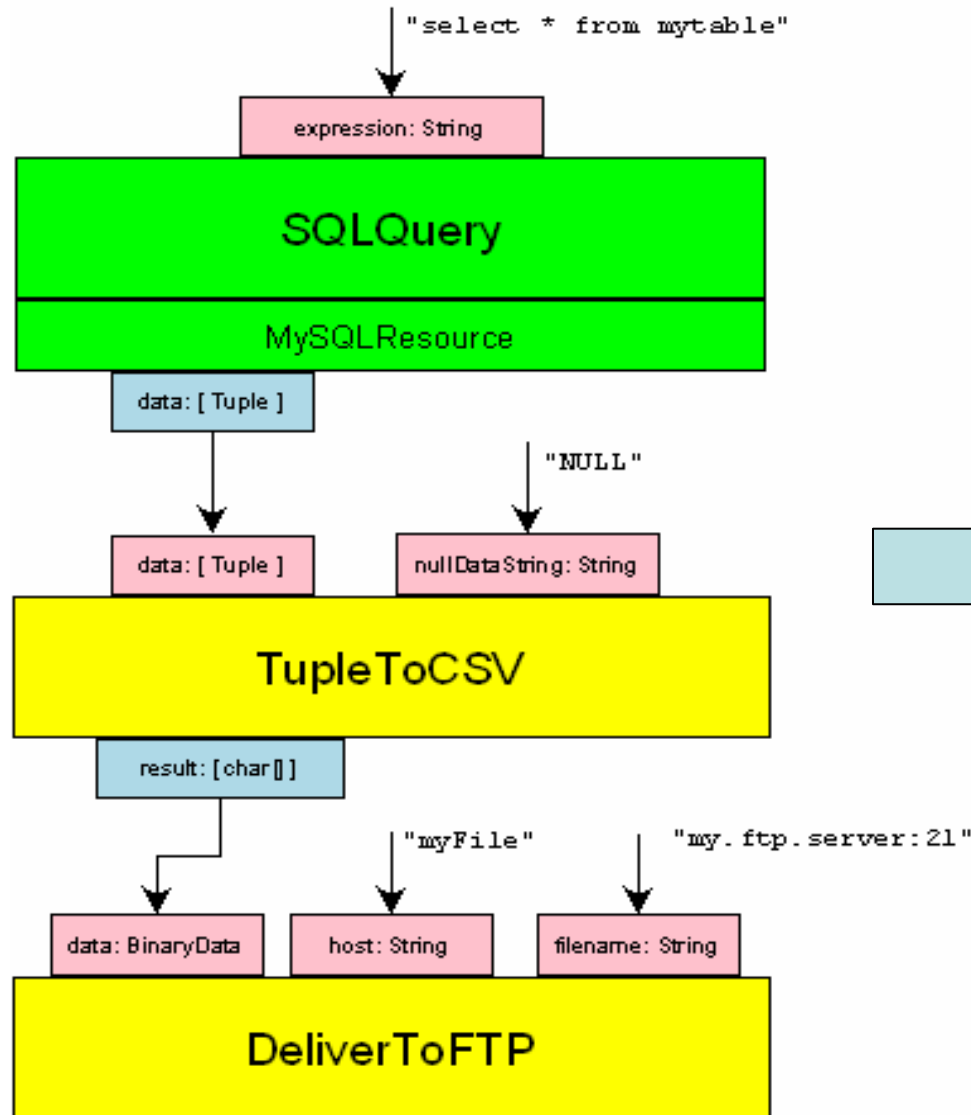
Activity Workflowの作り方。

1. Webサービスに投げるXMLとして
2. Javaのクライアントプログラムで

基本: pipeline状に順番にIOをつなぐ(単純)



複雑なものもOK



XML表現におけるワークフローの構築 (概念的な説明)

```
<workflow>
```

```
  <pipeline>
```

```
    <activity SQLquery>.....
```

```
      <outputStream pipe="pipe1">
```

```
    </activity>
```

```
    <activity TupleToCSV>,,,
```

```
      <inputStream pipe="pipe1">
```

```
    </activity>
```

```
  </pipeline>
```

```
</workflow>
```

アクティビティを順番に記述

2つのアクティビティの入出力を、pipe1でつなぐ

Javaにおけるワークフローの構築 (概念的な説明)

```
SQLQuery query = new SQLQuery();
    query.setResourceID(dataResourceID);
    query.addExpression("SELECT * FROM sample;");
SQLBulkLoadTuple loadTuples = new SQLBulkLoadTuple();
    loadTuples.setResourceID(dataResourceID);
    loadTuples.addTableName("result");
    loadTuples.connectDataInput(query.getDataOutput());
DeliverToRequestStatus deliver = new DeliverToRequestStatus();
    deliver.connectInput(loadTuples.getDataOutput());
PipelineWorkflow workflow = new PipelineWorkflow();
    workflow.add(query);
    workflow.add(loadTuples);
    workflow.add(deliver);
```

各アクティビティ
の
定義

ワークフローへの
Activityの追加

ワークフローの
作成

まとめ

- **OGSA-DAI**
 - サービスベースのデータベースアクセスミドルウェア
 - RDB(MySQL,Oracle,,) & XML(eXist,,)
 - **Activity Framework**
 - Query(SQL,XQuery,XPath)
 - Convert
 - Transfer
 - etc
 - **Activity Workflowのモデル**
 - XML
 - Java

★ OGSA-DAIにおける考え方。

1. 一つのデータサービスでの処理を高度化する。

- Activityの連携でワークフローを記述、処理
 - ひとつのサービス内でできることを高度化

2. 複数のサービスにおける連携(後述)

- データベース処理を含む汎用ワークフロー
 - BPELやTavernaなど、汎用ワークフローエンジンと組み合わせよ。
 - 該当モジュールの提供
 - WEEPなどDAI向きBPELエンジンの提供
- 分散データベース処理に基づくワークフロー
 - 分散問い合わせ処理のミドルウェア+ α の提供
 - OGSA-DQP
 - 問い合わせ処理を最適化する。

いいところ

単なるSQLのリモートアクセスとどう違うか？

- サービスベース
 - HTTPポートの開いているところでは、どこでもOK
- GSIをサポート
 - GSIとDB(例えばOracle)のアカウントのマッピング
- **結果のデータ変換をサポート**
 - XSLTを支援、検索結果の加工など。
- **第3者転送・大量データ処理をサポート**
 - FTP・GFTPなどにより、結果を第3者サイトに転送できる

実装上の制限

- **分散問い合わせやサービス連携:別のミドルの分担**
 - 関係DBについての分散問い合わせができる。
 - OGSA-DQP(最適化も含め実行計画を立ててくれる)
 - XMLの分散問い合わせはない。
 - それ以上の処理はワークフローエンジンと組み合わせる
 - サービス連携
- **ミドルウェア→プロダクト依存性がある**
 - 問い合わせを投げ、結果をもらうためのミドルウェア
 - XMLDBとRDBを問い合わせレベルでは統合できない。
 - RDBのプロダクト固有の機能については、連携・統合できない。

拡張性

- **OGSA-DAI=Activityの集合**
Activity=XMLSchemaで書かれたIOインターフェイスと、実装(クラス)の記述からなる。

OGSA-DAIの構成ファイルは、Activityの定義の集まり。

Activityは拡張・変更できる

- 新しいインターフェイスと実装を追加する。
- 既存のインターフェイスを改造する。
- 実装を差し替える、
- などなど、、、

Activity定義・拡張の例

- GEODE project: (産業医学・健康管理)

- DB周辺処理のアクティビティ化
 - ルール処理
 - メタデータ登録・管理など
- マクロ的・典型的な処理のアクティビティ化

- 産総研 OGSA-WebDB,OGSA-DAI-RDF

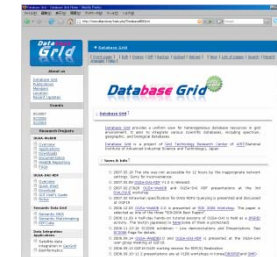
<http://www.dbgrid.org/>

- OGSA-WebDB:

- Webデータベースを仮想的に関係DBとしてみせるActivity

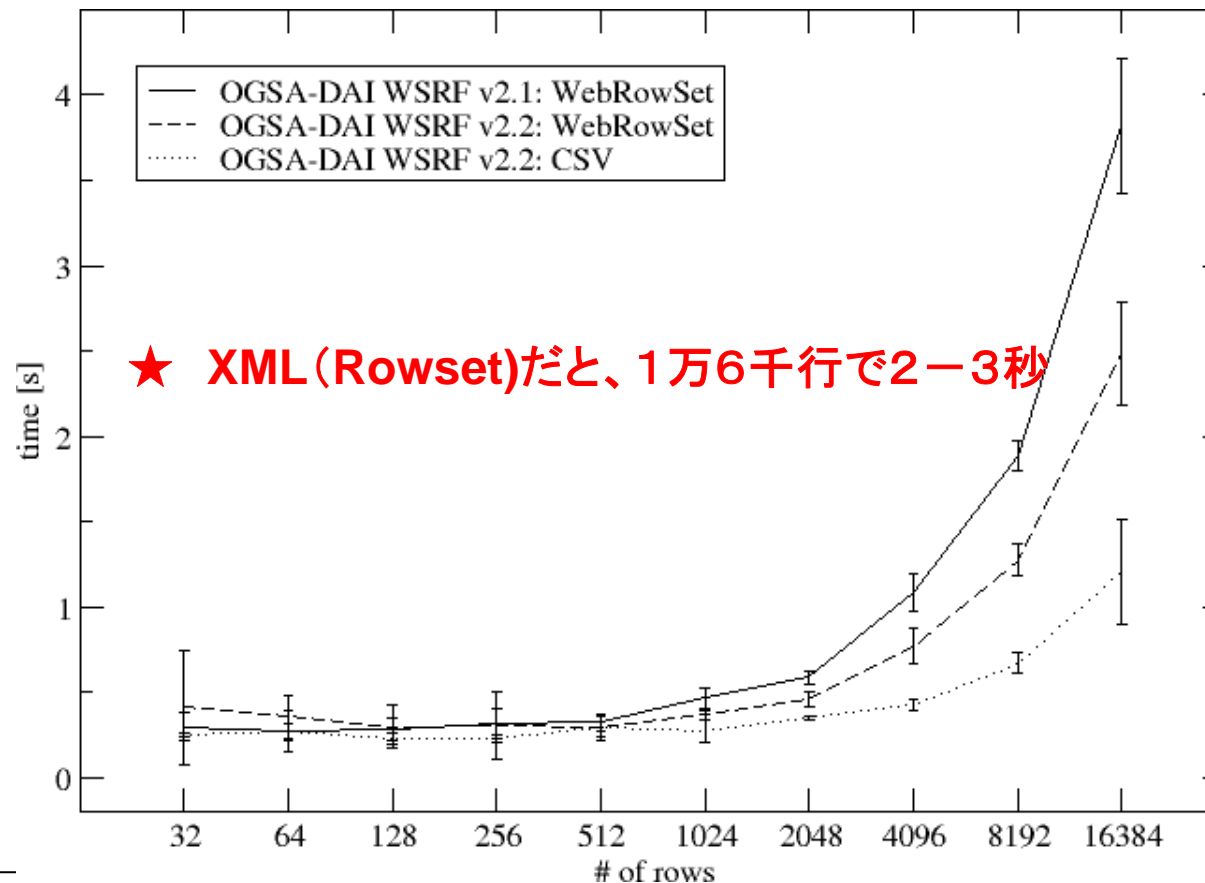
- OGSA-DAI-RDF:

- Jena,SesameといったRDFのデータベース処理のActivity



OGSA-DAIの性能

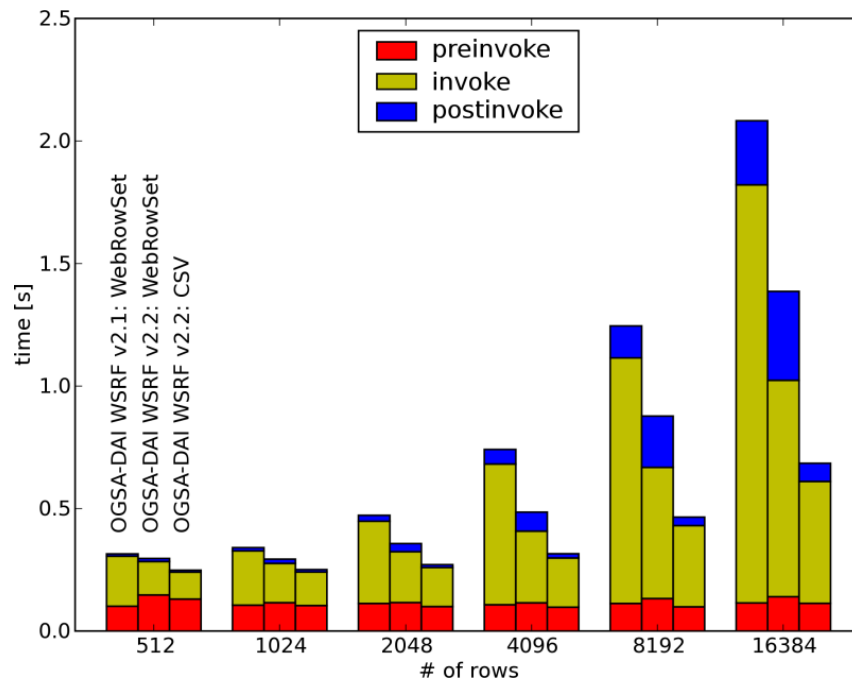
- サービスベース & XMLベースの転送
 - データ量が大きくなるに従って増加



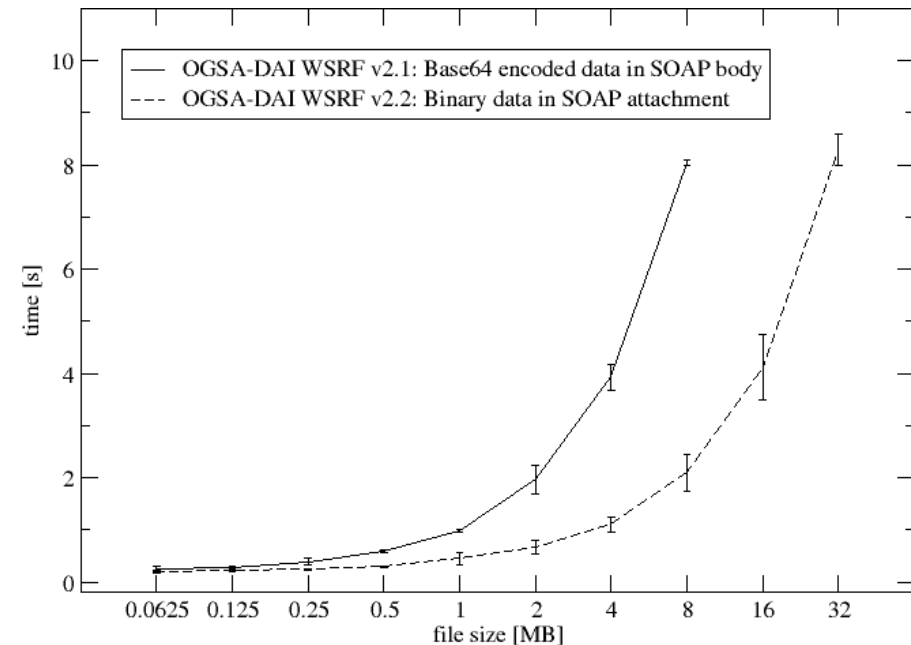
継続的な改良

- サーバソフトの改良

- バイナリ表現の利用による転送データサイズの縮小



★ 緑色がサーバの処理時間

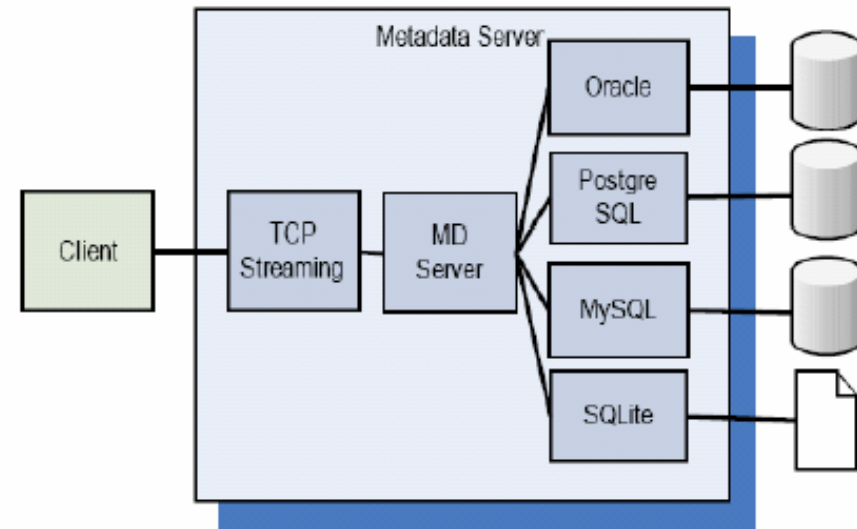


★ 8MBの結果データで8秒→2秒

他のミドルウェアの紹介
(OGSA-DAI以外のものが増えている)

AMGA

1. メタデータカタログ
2. SQL「風」言語による
プロダクト依存性の解消



Examples

```
createdir /jobs
addattr /jobs jobStatus int
addentry /jobs/job1 jobStatus 0
updateattr /jobs jobStatus 1 jobID>100
selectattr /DLibrary:FileName /DLAudio:Author /DLAudio:Album
'/DLibrary:FILE=/DLAudio:FILE and like(/DLibrary:FileName, "%.mp3")'
```

3. 高速
4. WS-DAIの支援

CORAL

ベンダ独立な
問い合わせ言語APIの提供

“SQL-free” API CERN IT Department

Example 1: Table creation

```
coral::ISchema& schema = session.nominalSchema();
coral::TableDescription tableDescription;
tableDescription.setName( "T_t" );
tableDescription.insertColumn( "I", "long long" );
tableDescription.insertColumn( "X", "double" );
schema.createTable( tableDescription );
```

Oracle

MySQL

```
CREATE TABLE "T_t" ( I NUMBER(20),
X BINARY_DOUBLE)
```

```
CREATE TABLE T_t ( I BIGINT,
X DOUBLE PRECISION)
```

Vendor independent “SQL-free” API CERN IT Department

Example 2: Issuing a query

```
coral::ITable& table = schema.tableHandle( "T_t" );
coral::IQuery* query = table.newQuery();
query->addToOutputList( "X" );
query->addToOrderList( "I" );
query->limitReturnedRows( 5 );
coral::ICursor& cursor = query->execute();
```

Oracle

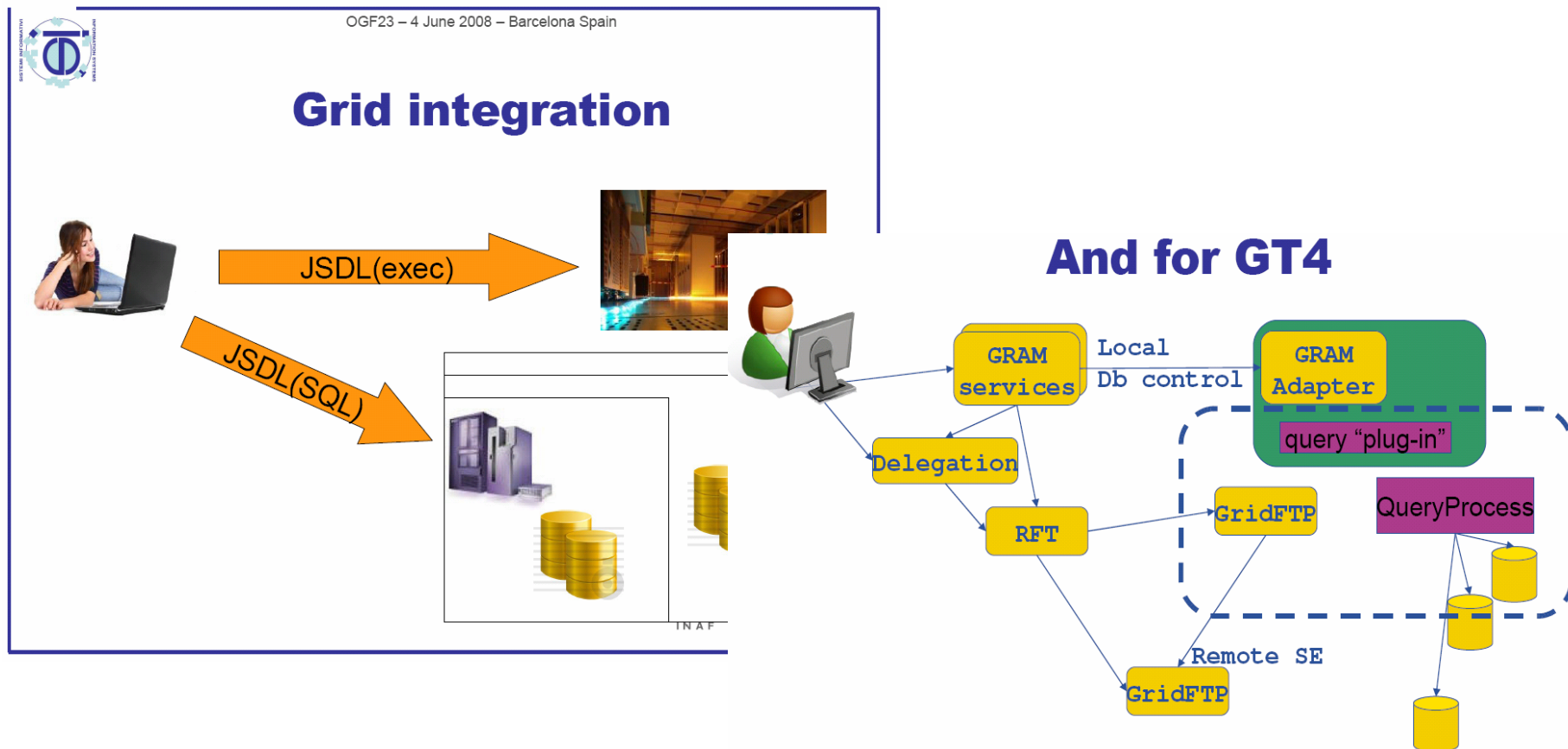
MySQL

```
SELECT * FROM
( SELECT X FROM "T_t" ORDER BY I )
WHERE ROWNUM < 6
```

```
SELECT X FROM T_t ORDER BY I LIMIT 5
```

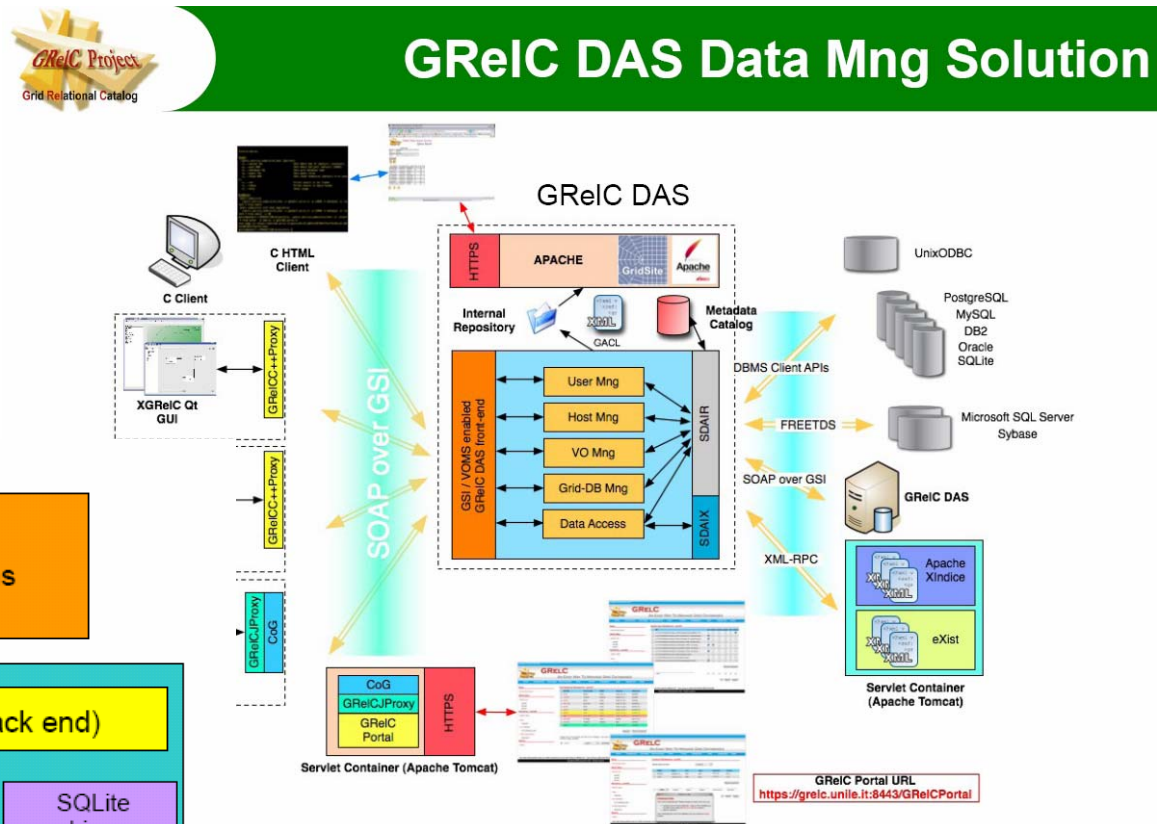
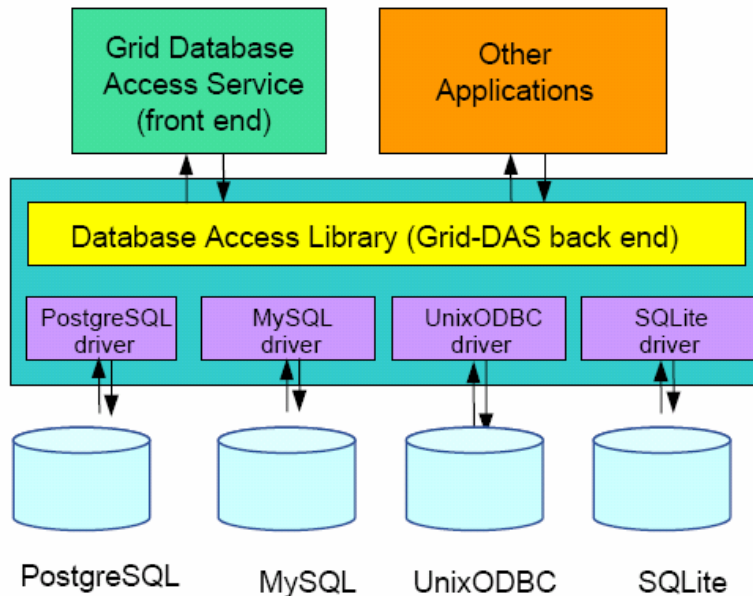
G-DSE

JSDL=グリッドのジョブ実行言語でSQLを投げる
 計算とデータベース処理が同一枠組みで実行




GReIC

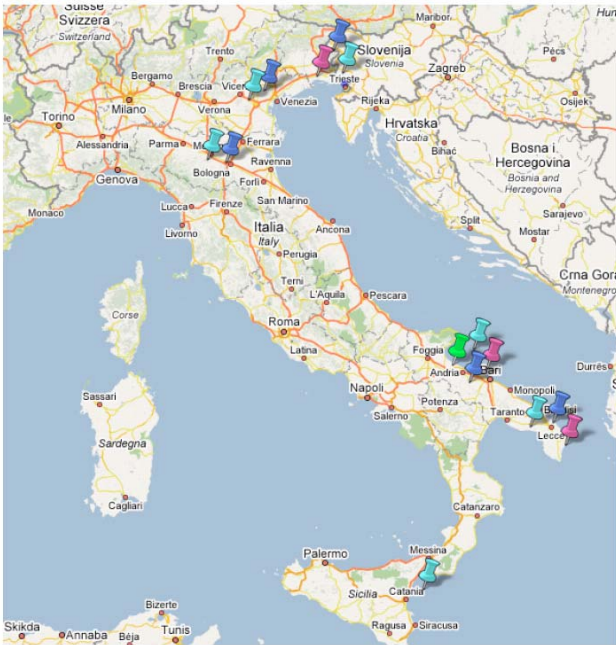
- OGSA-DAIと類似の構造を持つミドルウェア
- 互換性はない
- WS-DAIを支援予定



性能評価



TESTBED

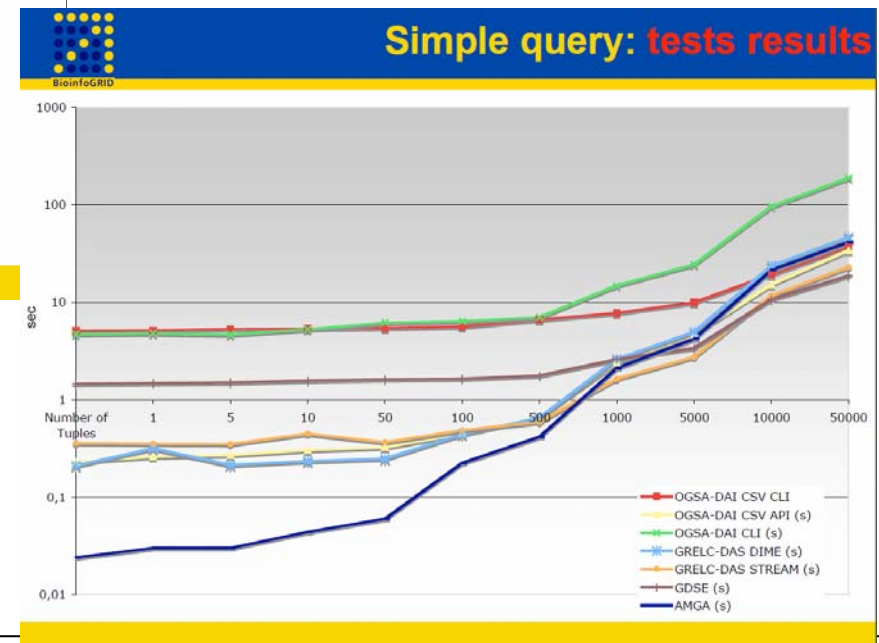


- 📌 OGSA-DAI
- 📌 G-DSE
- 📌 AMGA
- 📌 GRELC-DAS

Test Database: Bioinformatics database containing just a "molecule table" with about 500.000 tuples (350MB, PostgreSQL).

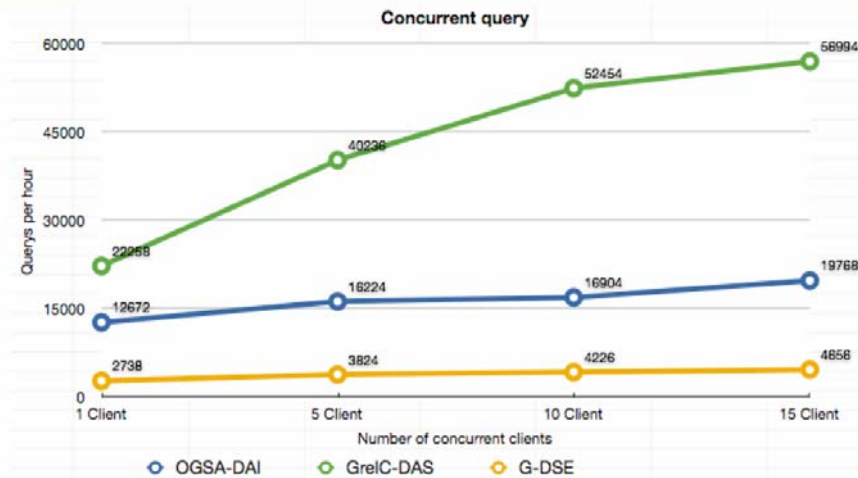
Other Databases:

- Sakila (MySQL 23 tables)
- World (MySQL 6 tables)
- Dellstore (PostgreSQL 8 tables)
- uniutrdb_test (MySQL 35 tables)
- go_5_06 (MySQL 18 tables)
- homo_sapiens (MySQL 74 tables)
- 2MASS (PostgreSQL)
- Population db (PostgreSQL)

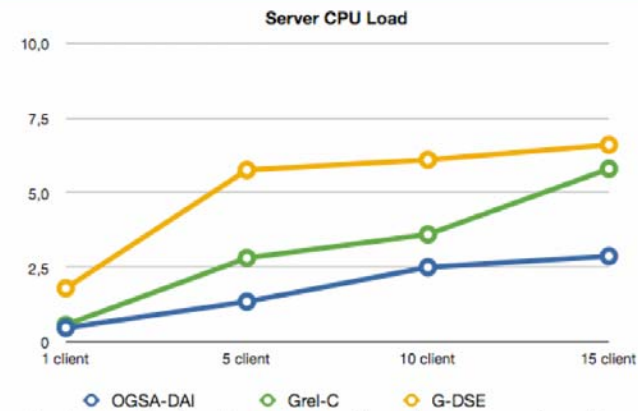


Performance

Concurrent test: number of query



Concurrent test: server load



- This graph shows the load average of each server when the concurrent test is executed:
 - It seems evident that for OGSA-DAI the number of client needed to saturate the machine is not reached yet... maybe we can achieve a better results increasing the number of concurrent client

OGSA-DQPとは

分散処理へのアプローチ

- やりたいこと

- 分散データベース処理を含む分散ワークフロー
 - サービス連携
 - SOAベースの応用開発

ナイーブに応用を作ると性能が悪くなりそうだ。

OGSA-DAIにおける考え方。

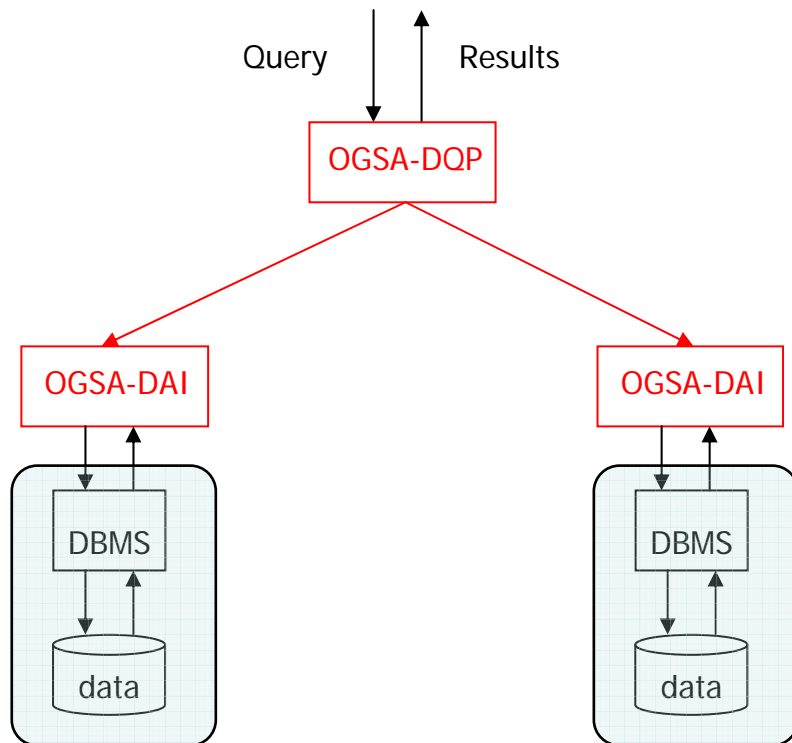
1. 一つのデータサービス内でのワークフロー

- Activityの連携でワークフローを記述、処理
 - ひとつのサービス内でできることを高度化

2. 複数のサービスにおける連携

- データベース処理を含む汎用ワークフロー
 - BPELやTavernaなど、汎用ワークフローエンジンと組み合わせよ。
 - 該当モジュールの提供
 - WEEPなどDAI向きBPELエンジンの提供
- 分散データベース処理に基づくワークフロー
 - 分散問い合わせ処理のミドルウェア+ α の提供
 - OGSA-DQP
 - 問い合わせ処理を最適化する。

OGSA-DQP (Distributed Query Processing) とは



- OGSA-DAIの関連プロジェクト.
- 分散SQLのミドルウェア
- 複数のOGSA-DAIを連携、統合。
 - ただしSQLのみ
 - Webサービスを呼べる
- DQP自身もOGSA-DAI互換サービスとして実装。
- 問い合わせ最適化
 - スケジューリング
 - 並行実行
 - データ転送

SQLの例

- Given two DBMSs and one analysis tool (i.e., a Web service):
 - **goTerm** : MySQLで作られたGene Ontology のデータベース。OGSA-DAI が実装されている。ice
 - **protein** : 別のDBの protein sequence DB, これも OGSA-DAI で実装。service
 - **Blast** (Web サービス経由で実行できるものとする);
- 分散した複数のDBを結合して、その結果をBlastにかける。

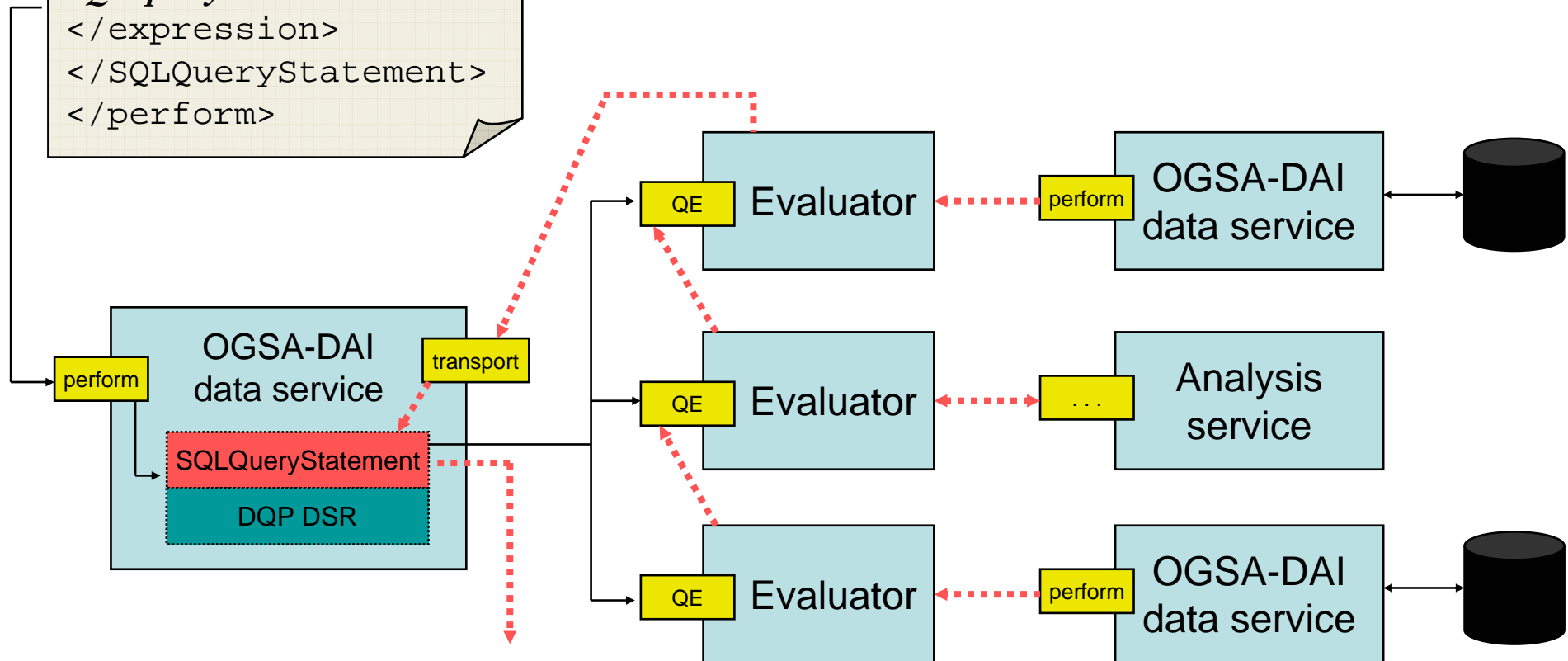
```
select p.proteinId, Blast(p.sequence)
from protein p, goTerm t
where t.termId = 'GO:0005942' and p.proteinId=t.proteinId
```

DQPの処理概要

Activityの連携(各サイトにおける処理とデータ転送の指定)を、うまく使って処理を分割・指示できる。

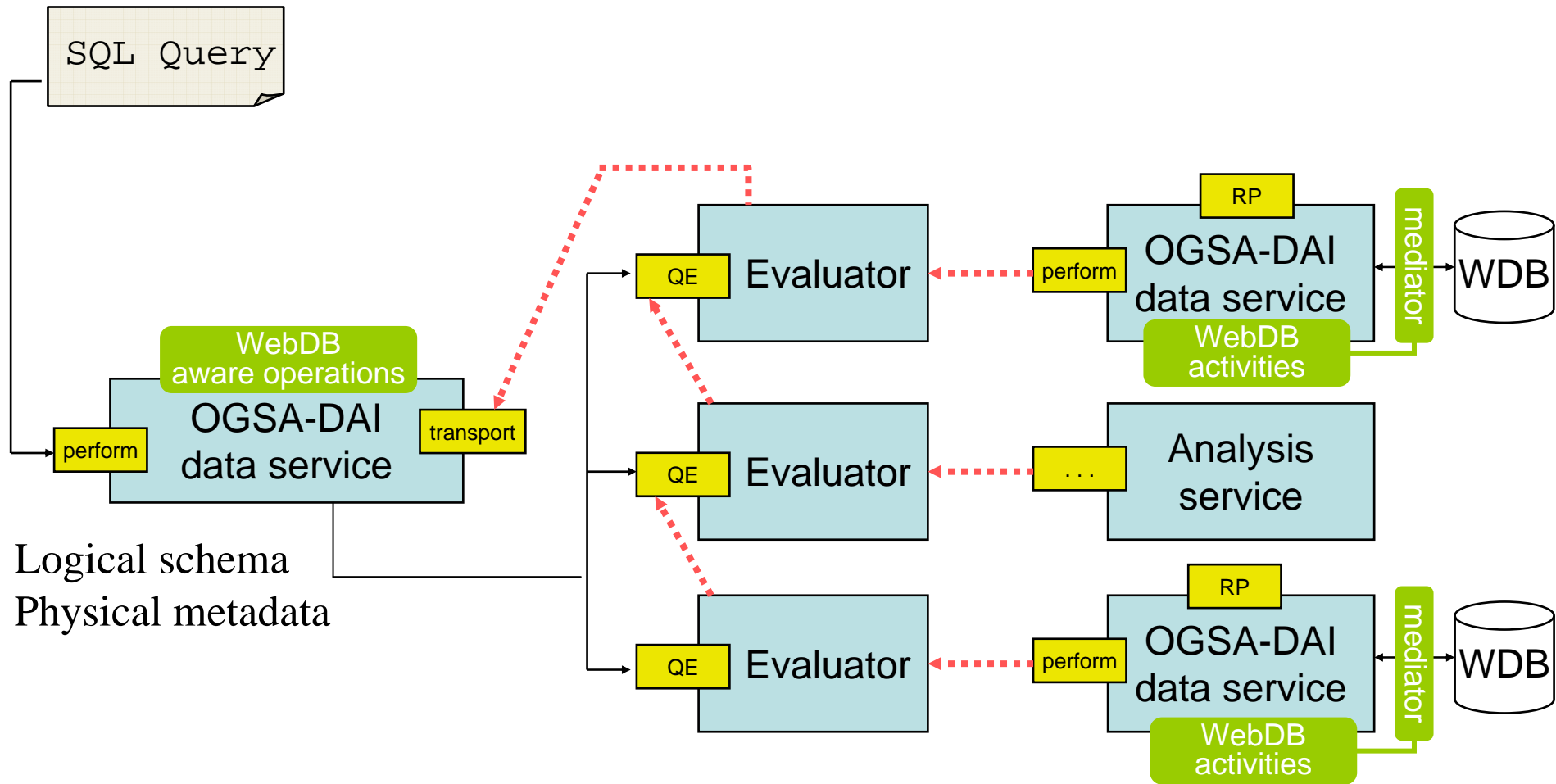
```

<perform>
<SQLQueryStatement>
<expression>
SQL query
</expression>
</SQLQueryStatement>
</perform>
    
```



Result: WebRowSet XML Stream

Web DB Integration



XML とRDBの統合 (SQLの拡張)

```
select XMLGen(  
    '<person>  
    <name>{$p.name}</name>  
    <address>{$p.address}</address>  
    </person>'  
) from person as p
```

Extract data from a relational table, construct an XML result using a template

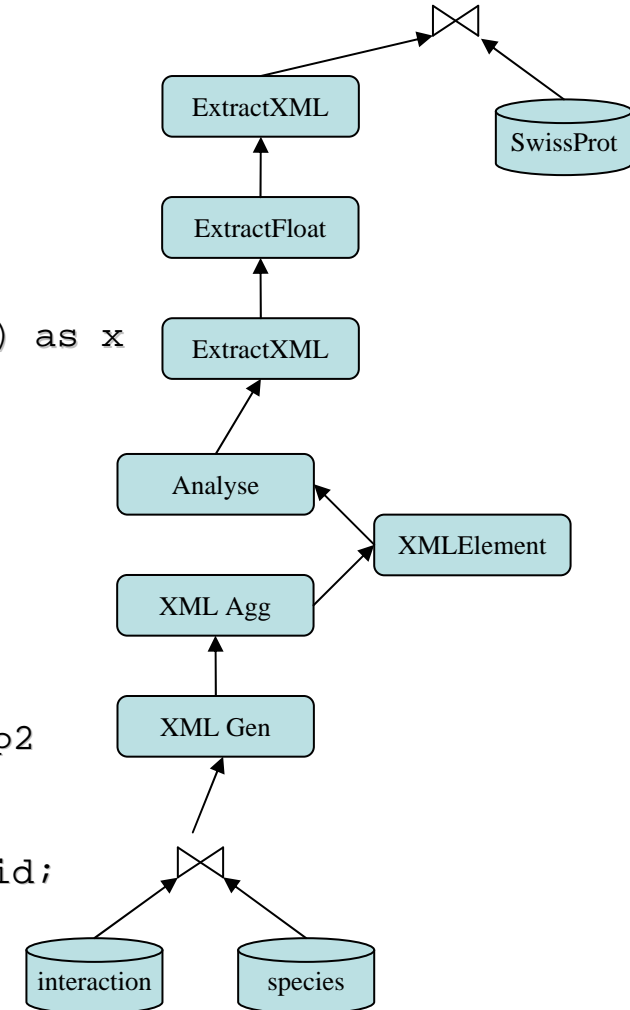
```
select XMLStringValue(p, '//name')  
from ExtractXML(people, '//person') as p
```

Create a result with a single String column. Data is extracted from an XML collection

```

select EntryName, GeneName, Taxonomy
from
  (select ExtractXML(x,'//result') as result
   from
     (select analyse(XMLAgg(
       XMLGen(<interaction>
         <node1>{$p1}</node1>
         <node2>{$p2}</node2>
       </interaction> ) as i
      from
        (select protein1 as p1, protein2 as p2
         from interaction, species
         where species='Homo sapiens'
          and interaction.organism=species.id;
        )
      )
     )
   )
  , SwissProt
where ExtractFloat(result,'//score') > 0.9
  and SwissProt.Accession_Number = ExtractXML(result,'//name');

```



まとめ

- **WS-DAI & 関連ミドルウェア**
 - WS-DAI: 実装が増えて規格として利用されつつある。
 - ミドルウェア: OGSA-DAI以外のものが増えてきた。
- **応用: e-Scienceからビジネスまで**
 - 実地的な応用実験へ適用されつつある。
- **研究開発**
 - 異種データ統合など

Questions?