Web Services Advanced Topics

Wokflows & Web Services
Kapitel 4

Coordination and Transactions

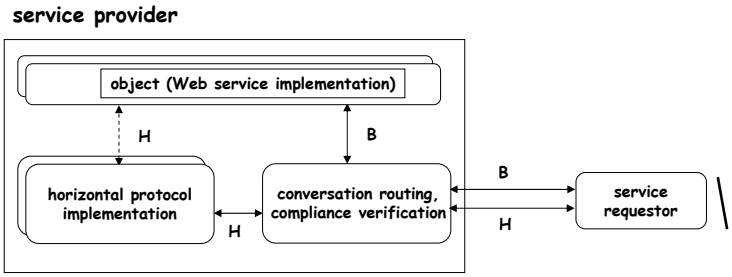
Coordination - Motivation

- Interactions are typically more complex than simple invocations
- Need to coordinate (sets of) activities or applications
  - Distributed
  - Running on different platforms using local coordinators
- Examples
  - Reach consistent agreement on the outcome of distributed transactions
    - Atomic transactions, 2PC
  - Coordinate auctioning activities
    - involves seller, auctioneer, buyers
  - Interactions between a customer and a supplier for ordering a product
- Interactions form a *conversation*
  - sequences of operations (message exchanges)
- Interactions adhere to a *coordination protocol*
  - specifies a set of correct/accepted conversations
  - *vertical* protocols: specific to business area (e.g., product ordering protocol)
  - *horizontal* protocols: define common infrastructure (e.g., transactions)

Infrastructure for Coordination Protocols

- Middleware support can be provided, with various degrees of automation
  - Conversation Controller
    - performs *conversation routing*
      - dispatch message to the appropriate "internal object"
        - one object for each instance of a conversation (e.g., an ordering session)
      - involves message correlation (conversation identifier), management of conversation context
        - example: session id
    - verifies *protocol compliance*
      - understand definition of the protocol
      - check if all messages adhere to the protocol definition
  - Generic Protocol Handlers
    - module that implements a specific coordination protocol
      - includes protocol-specific logic
      - processes and generates messages in accordance with the protocol rules
    - mostly applicable to horizontal protocols
      - example: transactions
    - forms of protocol execution support
      - handler realizes complete support, no intervention from the web service
      - handler and web service jointly realize the support
        - Example: atomic, distributed TAs

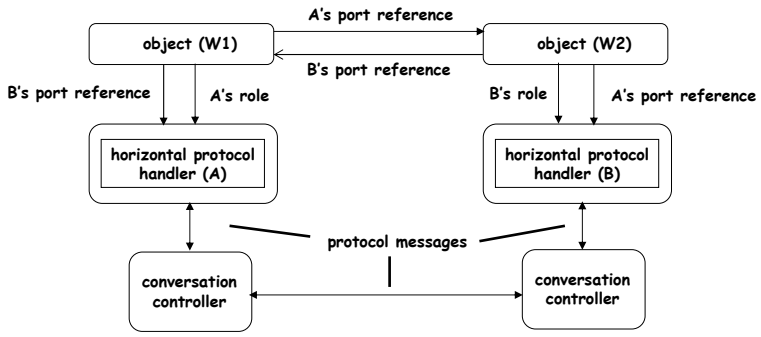
Implementing Horizontal Protocols



B: conversation compliant with a business protocol
H: conversation compliant with an horizontal protocol

source: Alonso et.al.: Web Services, Springer, 2003

Communicating Roles and Port References



source: Alonso et.al.: Web Services, Springer, 2003

Standardization

- Coordination infrastructure support for web services needs to be based on standards for
  1) generating and transporting unique conversation identifiers in SOAP headers
     - needed to map messages to conversations, and eventually to the objects handling them
  2) a framework and a set of (meta-) protocols for agreeing on which protocol is to be executed on how it is coordinated
  3) horizontal protocols
     - to separate horizontal protocol implementation from the inidividual web services
  4) protocol languages
     - to allow for protocol verification
- Web Services Coordination (WS-Coordination) Specification
  - standardizes 1), 2)
- Web Services Atomic Transaction (WS-AtomicTransaction) Specification
  - uses WS-Coordination framework to define coordination type for Atomic Transactions (i.e., it standardizes 3) for atomic TAs)
- Both specifications are no official standards yet
  - proposals by BEA, IBM, Microsoft

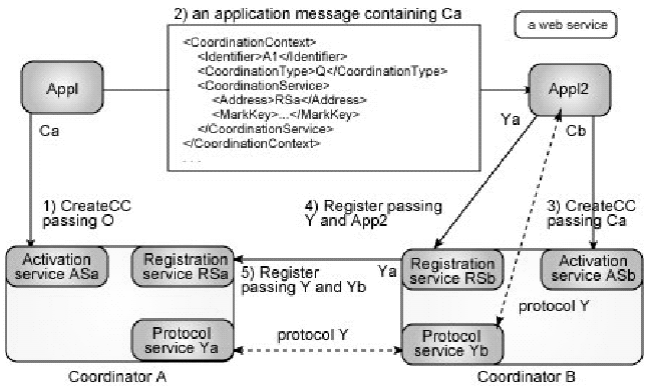
WS-Coordination

- Basic entities are **coordinators** and **participants** that wish to be coordinated
  - central coordination: all participants talk to a single coordinator
  - distributed coordination: each (or multiple) participants talk to its own coordinator
  - a participant can again be a (subordinate) coordinator
    - example: hierarchical 2PC
- Abstractions to describe the interactions between coordinator and participants
  - coordination protocol
    - set of rules governing the conversation
      - example: 2PC
  - coordination type
    - set of logically related protocols
      - example: atomic transactions
    - instance of a coordination type may involve several instances of the coordination protocols
- Coordination context
  - used to exchange coordination information among different parties
    - placed within messages exchanged between parties
    - contains coordination type, identifier of the coordination type instance

The Model

- Coordination service (coordinator) consists of
  - **Activation** service
    - Used by a participant to create coordination context (initiate instance of protocol type)
  - **Registration** service
    - Enable application to register for coordination protocols
  - (set of) **coordination protocols**
    - Specific to coordination type
- Extensibility
  - Publication of new coordination protocols
  - Definition of extension elements that can be added to protocols and messages
    - Exchange application-specific data on top of defined message flows

Distributed Coordination - Interactions



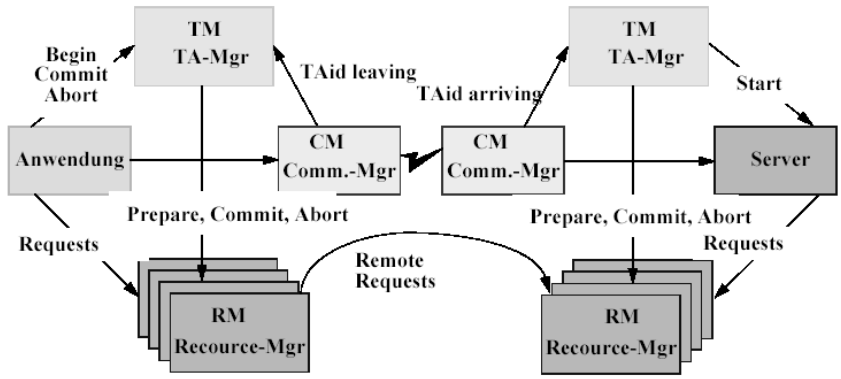
WS Atomic Transactions

- Atomic Transactions (TA) coordination type
  - Defines type-specific commit protocols
    - Completion: A participant (app creating the TA) registers so that it can tell the coordinator when/how to complete the TA (commit/abort)
    - 2PC: a resource manager (RM) registers for this protocol to be included in the commit/abort decision
      - Hierarchical 2PC (local coordinators can be interposed as subordinate coordinators)
    - Two variants of 2PC
      - volatile 2PC: a participant wants to be notified by the coordinator just before the 2PC begins
        - Example: participant caches, needs to communicate changes on cached data to DBMS before TA commits
      - durable 2PC: a participant manages durable resources
        - Example: DBMS
  - Participants can register for more than one protocol
  - Extension elements
    - Example: communicate isolation levels

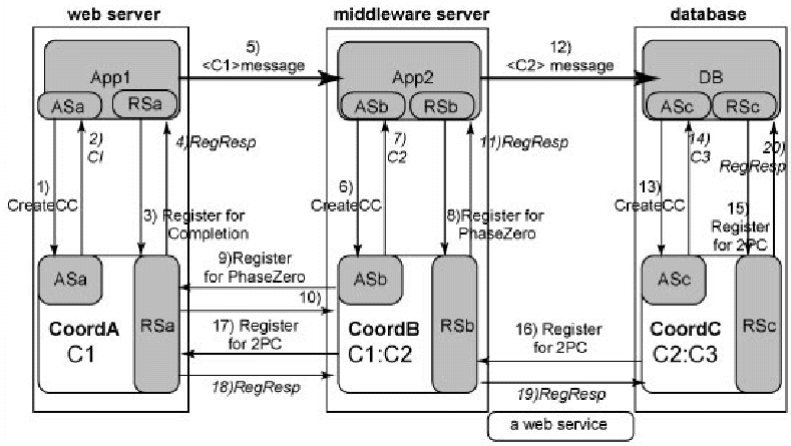
Atomic Transaction – Example

```
<?xml version="1.0" encoding="utf-8"?>
   <S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope“
      <S:Header>
         . . .
         <wscoor:CoordinationContext
         xmlns:wscoor=http://schemas.xmlsoap.org/ws/2002/08/wscoor
         xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
                  xmlns:myApp="http://www.w3.org/2002/08/myApp">
   <wsu:Identifier>http://foobaz.com/SS/1234</wsu:Identifier>
   <wsu:Expires>2002-08-31T13:20:00-05:00</wsu:Expires>
            <wscoor:CoordinationType>
         http://schemas.xmlsoap.org/ws/2002/08/wstx
            </wscoor:CoordinationType>
            <wscoor:RegistrationService>
                  <wsu:Address>
                  http://myservice.com/mycoordinationservice/registration
                  </wsu:Address>
                  <myApp:BetaMark> ... </myApp:BetaMark>
                  <myApp:EBDCode> ... </myApp:EBDCode>
            </wscoor:RegistrationService>
            <myApp:IsolationLevel>
                  RepeatableRead
            </myApp:IsolationLevel>
         </wscoor:CoordinationContext>
         . . .
      </S:Header>
      . . .
   </S:Envelope>
```

X/Open DTP revisited …



AT WS-Coordination Flow



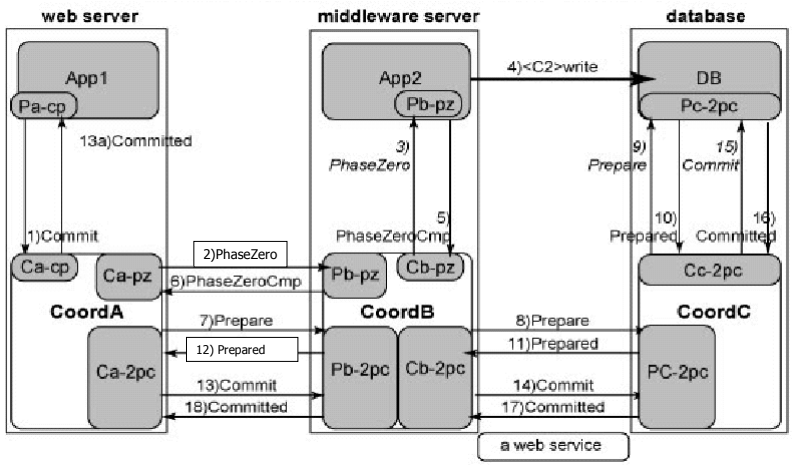
AT WS-Coordination Flow (cont.)

- App1:
  - sends a CreateCoordinationContext message (1) to its local coordinator's Activation service ASa
    - create an atomic transaction T1
    - gets back in a CreateCoordinationContextResponse message (2) a CoordinationContext C1 containing the transaction identifier T1, the atomic transaction coordination type and CoordA's Coordination PortReference RSa
  - sends a Register message (3) to RSa to register for the Completion protocol
    - gets back a RegisterResponse message (4), exchanging protocol service PortReferences for the coordinator and participant sides of the two-way protocol
  - sends an application message to App2 (5)
    - propagating the CoordinationContext C1 as a header in the message.
- App2:
  - decides to interpose local coordinator CoordB in front of CoordA
    - acts as a proxy to CoordA for App2
    - CoordA is the superior and CoordB is the subordinate
  - does this by sending a CreateCoordinationContext message (6) to the Activation service of CoordB (ASb) with C1 as input
    - getting back (7) a new CoordinationContext C2 that contains the same transaction identifier (T1) and coordination type, but has CoordB's Coordination PortReference RSb.
  - registers with CoordB for the PhaseZero (volatile 2PC) protocol (8 and 11)
    - CoordB registers with CoordA for the PhaseZero protocol (9 and 10)
  - sends a message to DB (12), propagating CoordinationContext C2

AT WS-Coordination Flow (cont.)

- DB:
  - decides to interpose its local coordinator CoordC by sending a CreateCoordinationContext message (13), further extending the superior-subordinate chain
    - gets back (14) a new CoordinationContext C3 that contains the same transaction identifier (T1) and coordination type, but CoordC's Registration service PortReference RSc
  - registers with CoordC for the 2PC protocol because it is a resource manager (15 and 20)
  - causes CoordC to register with CoordB for the 2PC protocol (16 and 19)
  - causes CoordB to register with CoordA for the 2PC protocol (17 and 18)

AT Coordination Protocol Flows

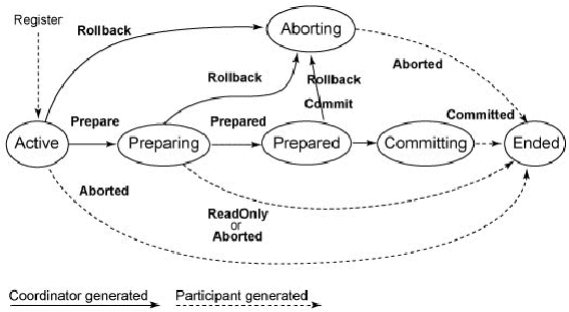


AT Coordination Protocol Flows (cont.)

- App1:
  - tries to commit the transaction using the Completion protocol (1)
- CoordA executes prepare-phase of **Volatile 2PC protocol**
  - has 1 participant registered for PhaseZero (CoordB), sends a Prepare message (2) to CoordB's PhaseZero Participant protocol service Pb-pz
  - CoordB relays Prepare message to App2 (3)
  - App2 sends its cached updates to DB
    - application message (4) propagates the CoordinationContext C2
    - sends a Prepared message (5) to CoordB
- CoordA executes prepare-phase of **durable 2PC protocol**
  - sends a Prepare message (7) to CoordB's 2PC Participant protocol service Pb-2pc
  - CoordB sends Prepare message (8) to CoordC's 2PC Participant protocol service Pc-2pc
  - CoordC tells DB to Prepare (9)
- CoordA commits
  - sends Commit message (13) to CoordB
    - Committed notification to App1 (13a) can also be sent
  - CoordB sends Commit message (14) to CoordC
  - CoordC tells DB to commit T1
    - DB receives the Commit message (15) and commits
  - Committed message returns (16, 17 and 18)

AT – 2PC Protocol

- Two-way protocol
  - Exchange of messages between coordinator and participant
- State Diagram
  - State reflects common knowledge of both parties



AT – 2PC Protocol (cont.)

- OnePhaseCommit
  - If only one participant has registered for 2PC, the commit/abort decision can be delegated to that participant
    - Send OnePhaseCommit message instead of Prepare message
  - Can be recursively applied by subordinate coordinator
- "Presumed abort" assumption
  - No knowledge of a transaction implies it is aborted
  - Allows for optimizations during commit phase
- "Read-only" optimization
  - After receiving a prepare message from the coordinator, participant can reply with a read-only message and skip the second phase
- Replay Message
  - Used by participant to solicit transaction outcome from coordinator after a failure

Security

Web Services Security

- Protect resources such that only appropriate "entities" can access them
  - Authorization: decide whether an identity can access a particulare resource
- Ensure the safety of information exchange among trading partners
  - Confidentiality: protection against eavesdroppers
  - Authentication: provide/verify proof of identity
  - Integrity: message was not modified accidentally or deliberately in transit
  - Non-repudiation: sender of message cannot deny he/she sent it
- Cryptography is used to protect the information exchange
  - Transport Security
    - Basic authentication, SSL/TLS
  - Web Service Security
    - Digital Signature, Encryption, …

Basic Cryptographic Concepts

- Encryption (-> confidentiality)
  - symmetric
    - same key is used for encryption and decryption
      - "shared secret"
  - asymmetric (public key cryptography)
    - public key, private key pairs
    - sender uses public key of the receiver to encrypt the message
    - receiver can decrypt the message only using the private key
    - computationally more expensive than symmetric encryption
  - often, asymetric encryption is only used for exchanging a symmetric key
- Message digest (-> integrity)
  - digest algorithm (similar to a hash function) is applied to data/message
  - produces a digest value (hash value) that depends on the original data
    - sent with the data
  - receiver can apply digest to the data and compare the result to the digest sent with the data
    - verify that data has not been augmented on the way
    - used in combination with digital signatures

Basic Cryptographic Concepts (cont.)

- Digital signature (-> integrity, authentication, non-repudiation)
  - The digest is encrypted with the private key of the signer, producing the signature
  - To verify the signature, anyone with access to the public key of the signer can
    - Decrypt the signature (original hash) using the public key
    - Apply the hash function to the original data
    - Compare the two hash values to make sure they are identical
  - Allows to make sure that
    - the data has not been modified
    - the data was actually sent by the owner of the public key
- Certificate
  - Data structure that holds at least the following information
    - identification (name, address, …) of the certificate owner (person, company)
    - public key of the certificate owner
  - issued by a certificate issuing authority
    - authority signs the certificate with its own private key

Transport Security

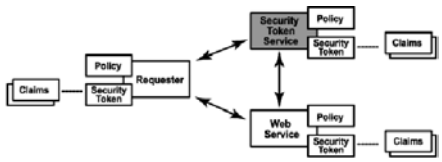
- HTTP Basic Authentication
  - UserID, Password authentication on the web
    - Initial HTTP request results in error "401 Unauthorized"
    - Browser opens dialog to request user, password info, resubmits the request
      - Userid/password are encoded in Base64, NOT encrypted
  - Web server verifies permissions based access control list (ACL)
- Secure Sockets Layer/Transport Layer Security (SSL/TLS)
  - Protocol for transmitting data in a secure way
    - point-to-point secure sessions
    - Can provide confidentiality, authentication, integrity
  - Located between application layer and transport layer (TCP)
    - Other protocols can be performed over SSL
      - HTTPS is HTTP over SSL
  - Supports server authentication and client authentication via certificates
    - The latter is rarely used, requires client to possess a certificate issued by a certificate authority
      - HTTP authentication frequently used here

Web Services and SSL/TLS

- SSL/TLS can be used for transmitting SOAP messages
  - SOAP/HTTPS
- Problems with SSL/TLS for SOAP messaging
  - SSL assumes that communication occurs directly between to parties
    - SOAP messaging may include third-party intermediaries
  - SSL encrypts the whole message
    - not possible to encrypt only parts of a SOAP message (e.g., the body)
  - SSL does not support digital signing of (parts of) the SOAP message

Web Services Security Model

- End-to-end security
- General Model
  - WS can, as part of its *policy*, require proof of a *set of claims* from a requester
    - name, key, permission, capability
  - A requestor can provide proof of claims with a message by attaching a *security token*
    - e.g., X.509 certificate, Kerberos ticket, …
  - Requestor may try to obtain required claims from *security token services*



Web Service Security

- Initially industry proposal, standardization by OASIS
- WS-Security
  - SOAP extensions (headers)
  - focus on WS integrity and confidentiality
    - pass security tokens, sign and encrypt messages
  - mechanisms to be used with other extensions, higher-level protocols for complete security solution
  - Leverages XML Encryption, XML Digital Signature, …
- WS-Security does not attempt to address interoperability across different security infrastructures and trust domains
  - how to make sure that partners understand and support each others security policies (e.g., which kind of security tokens are used, …)
  - this is left for other specifications to solve

SOAP Signature Details

- XML Digital Signature
  - Defines a Signature element with its descendents to store
    - Information about the hashing and encryption algorithms used
    - Signature itself
    - Public key to verify the signature
      - Or address of PK directory that includes the key
  - XML Canonicalization is used to produce canonical form before signing
- WS-Security specification
  - Defines how to embed the Signature element in a SOAP message as a header entry
  - Possible to sign whole message, parts of the message, attachements
    - Multiple signatures in the same SOAP message supported

SOAP Encryption

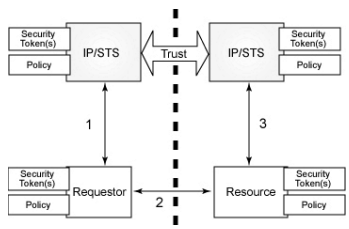
- XML Encryption
  - Defines EncryptedData element to hold
    - Information about the encryption method
    - Key information
      - Name of secret shared key, public key, …
    - Encrypted data
- WS Security
  - Defines Encryption element/header
    - Includes reference to encrypted data
    - Can be directed towards specific intermediary
  - Multiple encryption elements in the same SOAP message supported

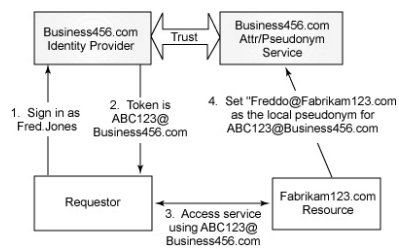
Policies

- Interoperability, step 1
  - ability to express how you implement security, what you expect from a service partner
- WS-Policy
  - express capabilities, characteristics of entities in a WS-based system
    - authentication scheme
    - transport protocol
    - privacy policy
    - Quality-of-Service characteristics
  - policy assertions, expressions, statements
  - allows senders, receivers to specify their security requirements and capabilities
- WS-PolicyAttachment
  - associate policy expressions with subjects
    - reference policies from WSDL definitions
    - associate policies with UDDI entities

Trust

- Interoperability, step 2
  - ability of a service partner to request from a recognized authority that a particular security token is exchanged for another
  - establish chain of trust
- WS-Trust
  - security token service (STS)
  - request/obtain security token
  - manage trusts, establish and assess trust relationships
    - build a chain of trust from recipient's trust authority to the sender authority
- WS-Federation
  - extends the WS-Trust model to allow attributes and pseudonyms to be integrated into the token issuance mechanism
    - provide federated identity mapping mechanisms
    - facilitate single sign-on





Additional Efforts

- WS-SecureConversation
  - describes how to manage and authenticate message exchanges between parties including security context exchange and establishing and deriving session keys
- Still to come as part of the web services security stack
  - WS-Privacy: will describe a model for how Web services and requesters state privacy preferences and organizational privacy practice statements
  - WS-Authorization: will describe how to manage authorization data and authorization policies
- XML Key Management Specification (XKMS)
  - Specifies protocols for distributing and registering public keys
- eXtensible Access Control Language (XACML)
  - Defines an XML Schema for an extensible access control policy language
- Security Assertion Markup Language (SAML)
  - XML security standard for exchanging authorization and authentication information

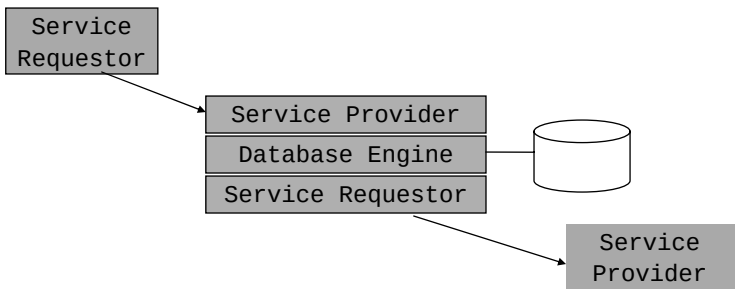
Security Assertions

- Security Assertion Markup Language (SAML)
  - XML standard for transporting security information between online commerce systems
    - Implement a single sign-on mechanism
  - Allows web sites and services to share information about a user
    - "entitlement" information
      - Credit limits, gold card profiles, …
    - Registration information
- Various security assertions
  - Authentication, attribute, decision
- Assertions are produced by their respective authorities
  - Example
    - Client sends request including userid and password to authority
    - Authority issues document containing authentication and attribute assertion (e.g., company ranking)
    - Client sends purchase order (request) to web service, attaching the security assertion
    - Service performs authorization, relying on the assertion

Databases and Web Services

Databases and Web Services

- Information Integration and dissemination
- Database as web service requestor
  - Invoking web services on my data
- Database as web service provider
  - Offering my data as service (making it easy)



Databases and Web Services

- DBMS as a web service provider
  - offer DB operations as web service
    - query, update, invoke a routine, …
  - "speak" XML
    - natively
    - translated
- DBMS as a web service consumer
  - invoke a WS through query/DML statement or as a side-effect of updates
  - process and analyze WS results inside query engine
  - provide integration services

SQL/XML

- Goal: standardization of interaction/integration of SQL and XML
  - how to represent SQL data (tables, results, ...) in XML (and vice versa)
  - how to map SQL metadata (information schema) to XML schema (and vice versa)
- Potential areas of use
  - "present" SQL data as XML
  - integration of XML data into SQL data bases
  - use XML for SQL data interchange
  - XML views over relational
    - possible foundation for XQuery
- Example
  - SQL table "EMPLOYEE"
  - XML document:

```
<EMPLOYEE>
  <row>
    <EMPNO>000010</EMPNO>
    <FIRSTNME>CHRISTINE</FIRSTNME>
    <LASTNAME>HAAS</LASTNAME>
    <BIRTHDATE>1933-08-
   24</BIRTHDATE>
    <SALARY>52750.00</SALARY>
  </row>
  <row>
    <EMPNO>000020</EMPNO>
    <FIRSTNME>MICHAEL</FIRSTNME>

   <LASTNAME>THOMPSON</LASTNAME>
    <BIRTHDATE>1948-02-
   02</BIRTHDATE>
    <SALARY>41250.00</SALARY>
  </row>
  …
</EMPLOYEE>
```

DBMS as a Web Service Provider

- Mapping for tables, schemas, catalogs to XML
  - no default mapping of arbitrary SQL query results in SQL standard
    - some work in the scope of JDBC web rowsets
- No standard way of publishing queries, routine invocations, etc. as a web service
  - left to tooling provided by DBMS vendors
  - SQL-based database web service
    - ability to send SQL to database and return results with default tagging (includes calls to stored procedures)
    - focus is data in and out of database rather than the format
  - XML-based database web service
    - Using DBMS-specific XML plug-ins engine support
    - Compose and decompose XML documents
- No standard set of web services for interacting with SQL or XML databases at the general API level
  - see ongoing work in data grid area

Example

- DB2 as an SQL-based web service provider

```
<?xml version="1.0" encoding="UTF-8"?>
<DADX xmlns=http://schemas.ibm.com/db2/dxx/dadx>
  <operation name="showemployees">
    <query>
      <SQL_query>SELECT * FROM EMPLOYEE</SQL_query>
    </query>
  </operation>
</DADx>
```

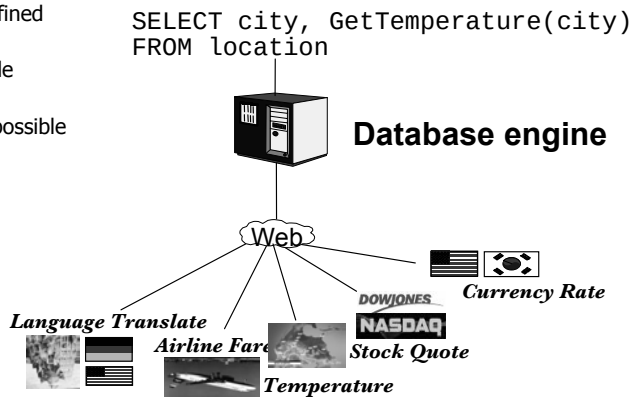
- DADx file (Document Access Definition Extension) contains definition of operations and corresponding data access statements to implement them
  - SQL, including stored procedure invocation
- WS tooling/runtime generates the corresponding web services, performs default tagging of results
- Can invoke DB2 XML extender functionality to perform composition/decomposition in a user-defined manner

DBMS as a Web Service Consumer

- Use SQL MED
  - web service result as one or more SQL tables
    - alternative: foreign routine
  - foreign data wrapper
    - invokes web service
    - maps (parts of) result from XML to SQL tables
    - challenge: support complex input parameters for WS
- Use SQL user-defined routines
  - web service as stored procedure
    - SP paradigm may not be adequate for further result processing
  - web service as user-defined (scalar or table) function
    - result is limited to a single value (chunk of XML) or a single table

Database – Web Service Requestor

- Web service invocation in engine
- Web Service UDF
  - SOAP User-defined Function
  - Scalar vs. Table Functions
  - Tool support possible



Grid Computing

Grid Computing

- Primary goal
  - computing as a utility
    - provide shared computing resources
    - hide details of components
      - location, management, …
    - virtualization of services
- Web Services
  - can be used in a Grid architecture to provide grid services
- Grid Computing and Databases
  - increased focus on data-intensive applications
    - significant processing on verly large amounts of data
      - collaboration
      - scalability
  - Grid for data access and integration

Global Grid Forum

- Open forum for standardizing grid interfaces
  - founded in 1998
  - produce technical specs that become grid recommendations
- Organized into topic areas, working/research groups
  - for example:
    - Architecture
      - Open Grid Services Architecture WG (OGSA)
      - Open Grid Services Infrastructure WG (OGSI)
    - Data
      - Data Access and Integration WG (DAIS)
      - OGSA Replication Services WG (OREP)
      - Data Format and Description Language WG (DFDL)
      - GridFTP WG (GridFTP)
      - Grid File System WG (GFS)

OGSA

- Identifies
  - the components that make up the infrastructure of a grid computing environment
    - described as services
  - the basic mechanisms which must be supported by grid components
    - expressed as web services
    - defined by OGSI
- Platform interfaces for
  - service groups and discovery, service domains, security, policy, messaging and queuing, events, distributed logging, metering and accounting, administration, transactions, orchestration
  - data management
    - access, replication, caching, metadata, schema transformation, storage

OGSI

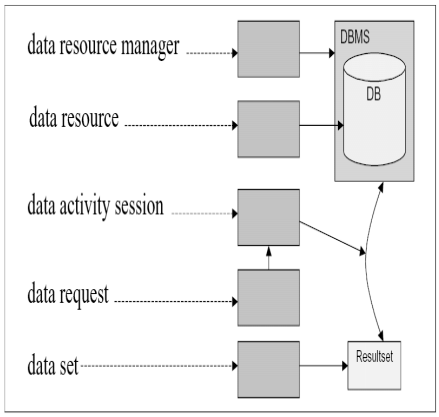
- Grid service must expose web service interfaces conforming to OGSI spec (e.g., factory)
- Grid services have state
  - long-term information to be maintained across client requests
- Conventions for performing service-related activities
  - handle: refers to an instance of a service
  - referring to collenctions of instances as a whole
  - factory: starting up service
  - service data: accessing a service state
  - state change notification
  - service lifetime management
  - inheritance support for grid services

DAIS – Data Access and Integration Services

- Service-based interface for accessing and integrating data on the grid
  - relational databases
  - XML databases
- Some features
  - naming results for subsequent use
  - multiple result formats
  - chunking large quantities of data
  - asynchronous result delivery
  - result delivery to third party
- Work in progress

DAIS – Main Constructs

- Services
  - Data Resource Manager
    - DBMS
  - Data Resource
    - database (tables or collections of XML)
  - Data Access Session
    - relationship between client and data resource
- Data Formats
  - Data Request
    - SQL, XPath, XQuery
  - Data Set
    - output result format



source: DAIS Grid Data Service Specification, June 2003

DAIS Topics

- DAIS model
  - see main constructs
- Transformations
  - transformation of results
- Stored Procedures
  - how parameters and result sets are handled
- Security
  - how database and grid security interact
- Transaction
  - transaction support in a grid environment
- Metadata
  - DBMS characteristics, database metadata, …