Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

Training

ANSAwise - Distributed Databases and Distributed Systems

Rob van der Linden

Abstract

The remote database access (RDA) paradigm, based on remote query language access to data, and the remote procedure call (RPC) paradigm, based on remote calls to predefined procedures, have become increasingly important in recent years. Each has unique advantages and disadvantages. Unfortunately, they are currently supported by different protocols and different client-server infrastructures, which makes it very difficult in practice to take advantage of their combined capabilities.

This talk describes how the object based concepts of the ANSA/ODP computational model for distributed computing can be used to align these two paradigms, thereby supporting mixed paradigm programming. The model is applicable to current generation relational databases and SQL, and it lays a solid foundation for future object databases and object query languages.

An acompanying paper (APM.1138) was written by Gomer Thomas and Rob van der Linden and submitted as a research paper to the 20th International Conference on Very Large Data Bases, to be held in Santiago, Chile, on September12-15, 1994. The paper was not accepted.

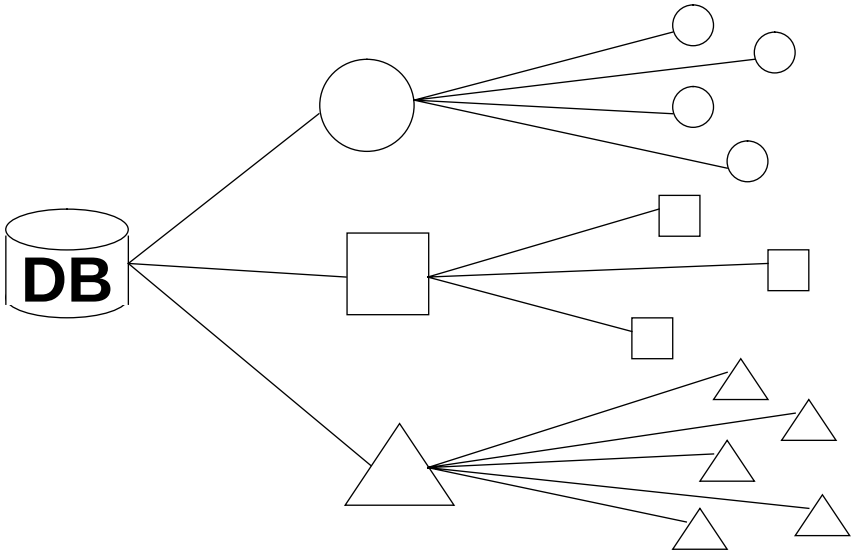
Thanks to Gomer Thomas of Bellcore, who laid the foundations for these slides.

APM.1332.01 | **Approved** | 24th October 1994

Briefing Note

Distribution:
Supersedes:
Superseded by:

Distributed Databases and Distributed Systems



In this session

- **Explain the Remote Data Access (RDA) paradigm**
- **Explain the Remote Procedure Call (RPC) paradigm**
- **Show how they can be unified**
- **Give an indication of partial solutions**

Two worlds

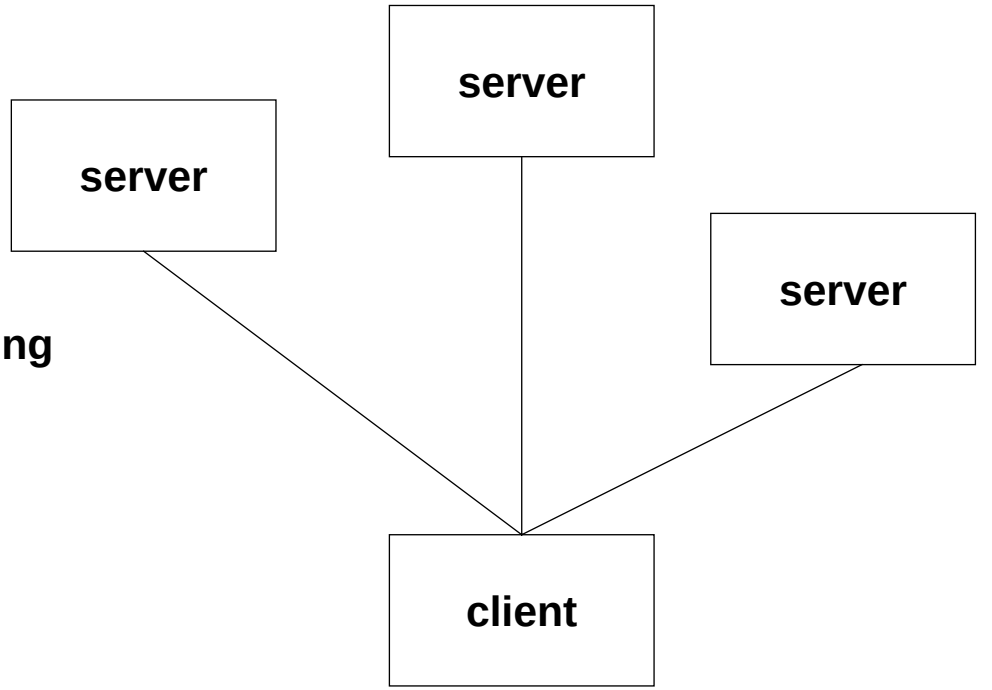
- Two approaches to distributed systems are predominant
  - Remote Data Access (RDA), generally used for databases
  - Remote Procedure Call (RPC), generally used for distributed computing
- RDA is based on the SQL structured query language
  - usually to access a relational database
- RPC is based on IDL interface definition languages
  - usually to access an application server
- Both are ‘client-server approaches’...
- ...Each has unique advantages
  - How can applications take advantage of both?

Remote Data Access (RDA) Overview

- Application interacts with database using SQL language statements
- Statements can retrieve, update, delete, or insert records:
  - a single record
  - a set of records at once
  - a set of records, one at a time
- Statements can be grouped into transactions
  - a transaction succeeds or fails as a whole
- The location of the remote database is transparent

RDA Paradigm

- **Basic Features**
  - **Transparent remote SQL**
  - **Dialog interaction**
  - **Dynamic SQL**
- **Enhancements**
  - **Distributed transaction processing**
  - **Distributed query processing**
  - **Static SQL**
  - **Stored procedures**



RDA Advantages

- **Flexibility for clients to define application-specific views and high level operations**

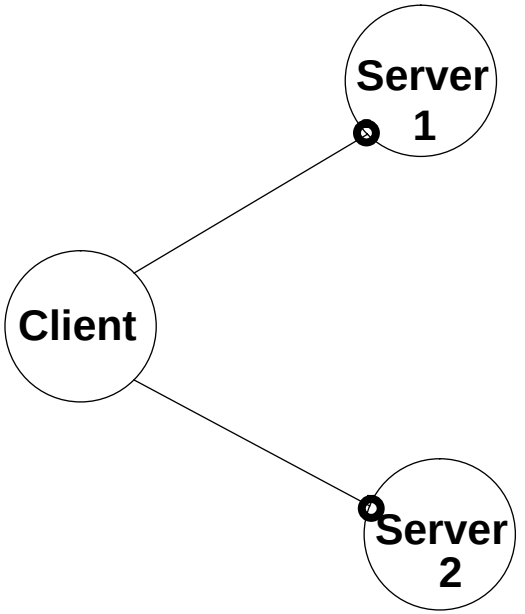
- **Vendor-supported interfaces to application development tools**
  - **data browsers**
  - **report writers**
  - **4GLs**
  - **spread sheets**
  - **graphing packages**
  - **etc.**

Remote Procedure Call (RPC) Overview

- **Application interacts with object using a remote procedure call defined in IDL**
- **Statements invoke individual operations defined in IDL**
- **Remote procedure call can do anything an ordinary procedure call could**
- **Transaction services can be used if available**
- **The location of the remote object is transparent**

RPC Paradigm

- **Basic features**
  - **transparent remote procedures**
  - **“one-shot” interaction**
  - **static procedures**
- **Enhancements**
  - **“transactional RPC” (dialog interaction)**
  - **distributed transaction processing**
  - **dynamic procedure definition**
  - **object-orientation**



RPC Advantages

- **High level of abstraction**
  - **client application programmer convenience**
  - **semantic integrity protection**
  - **low communications traffic**
- **Static compilation**
  - **efficiency**
  - **automated type checking**
- **Applicability to non-database services**
  - **for example, device control**

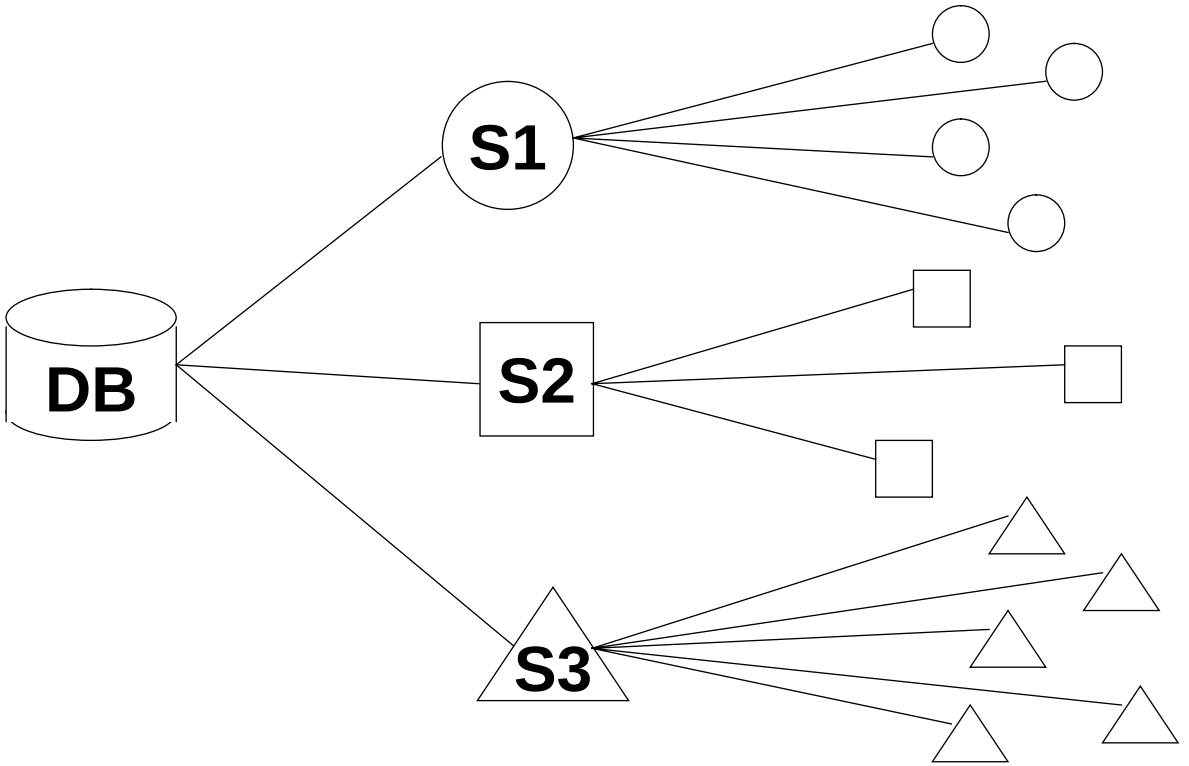
Unifying the RDA and RPC paradigms

- **The RDA and RPC paradigms work in different ways**
  - **RDA operates on sets; RPC operates on objects**
  - **RDA supports ‘connection-oriented’ client-server operations; RPC only supports ‘connectionless’ one-shot client-server operations**
  - **RDA supports a predefined query language; RPC can do anything**
- **To support general-purpose RPC access to an RDA database, it is necessary to model general-purpose mappings between them**

RPC-RDA Conceptual Model

- **Model for database**
- **Model for client-server interaction**
- **Model for RDA query language operations**
  - **Retrievals**
  - **Updates/deletes**
  - **Inserts**

Conceptual Model - Database

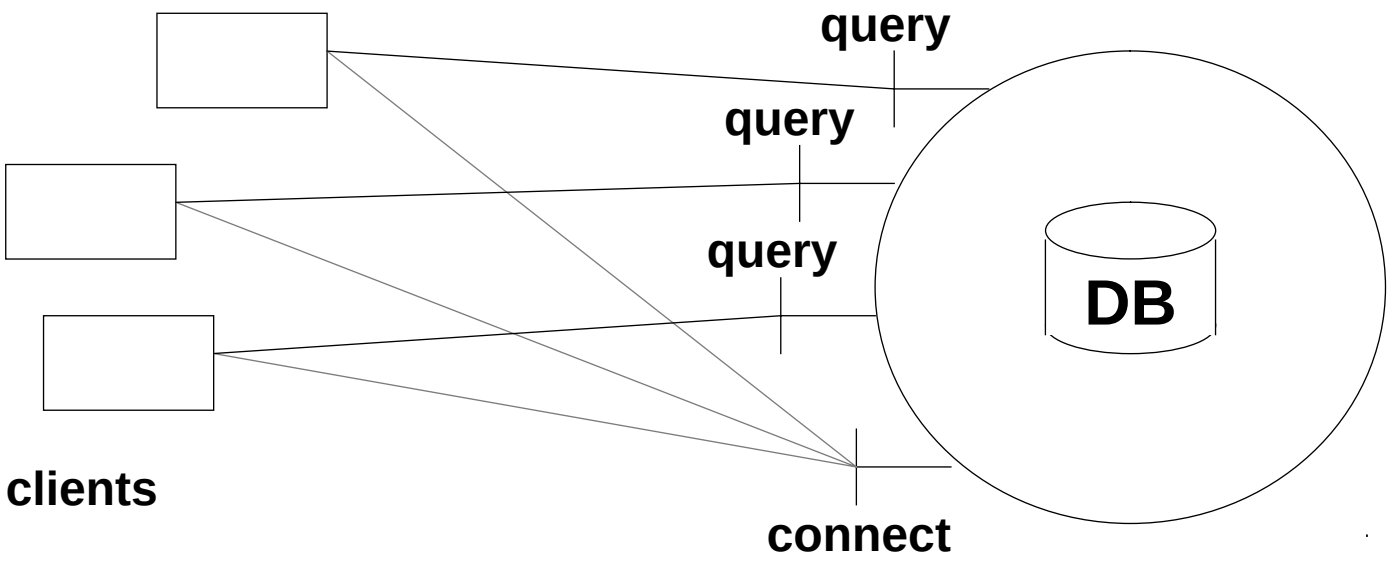


Conceptual Model - Database

- A set object is a collection of references to entity objects, all of same type
- A database object is a collection of references to set objects
- Each level has its own operations, which invoke operations on lower levels as needed
- No assumption is made about the locations of any of the objects

Conceptual Model - Client/Server Interaction

- **“Connection” interface (obtained by client from trader)**
- **“Query” Interface instances (created for individual clients on connection request)**



Conceptual Model - Retrievals

- A query statement is an operation constructor
  - defines operation type (signature), given schema
  - defines result value, given DB state
- Step 1: Invoke selection to generate result
  - input is query string and parameters appearing in query string, if any
  - output is set of entities
- Step 2: Invoke operations on result (typically including fetches)
  - input to fetch is reference to set of entities
  - output of fetch is (reference to) individual entity from set

Conceptual Model - Updates and Deletes

- **For searched update/delete - query statement is operation constructor**
  - **input is query string and parameters appearing in query string, if any**
  - **no output (except possibly status code)**
- **For positioned update/delete - treat as operations on fetched object**

Conceptual Model - Inserts

- **Single object insert - create object, include reference in set**
  - **straightforward enough**
- **Computed set insert - this requires two steps**
  - **Step 1: Generate result set to be inserted, via select**
  - **Step 2: Merge into target set**

Conceptual Model - Infrastructure Support for Queries

- **In all cases one wants to view a query statement as a constructor**
  - **a constructor which *dynamically* defines an operation**
- **The RPC (or object operation invocation) infrastructure needs to support this sort of dynamic operation definition**
- **Dynamic operation definition raises the issue of type checking**

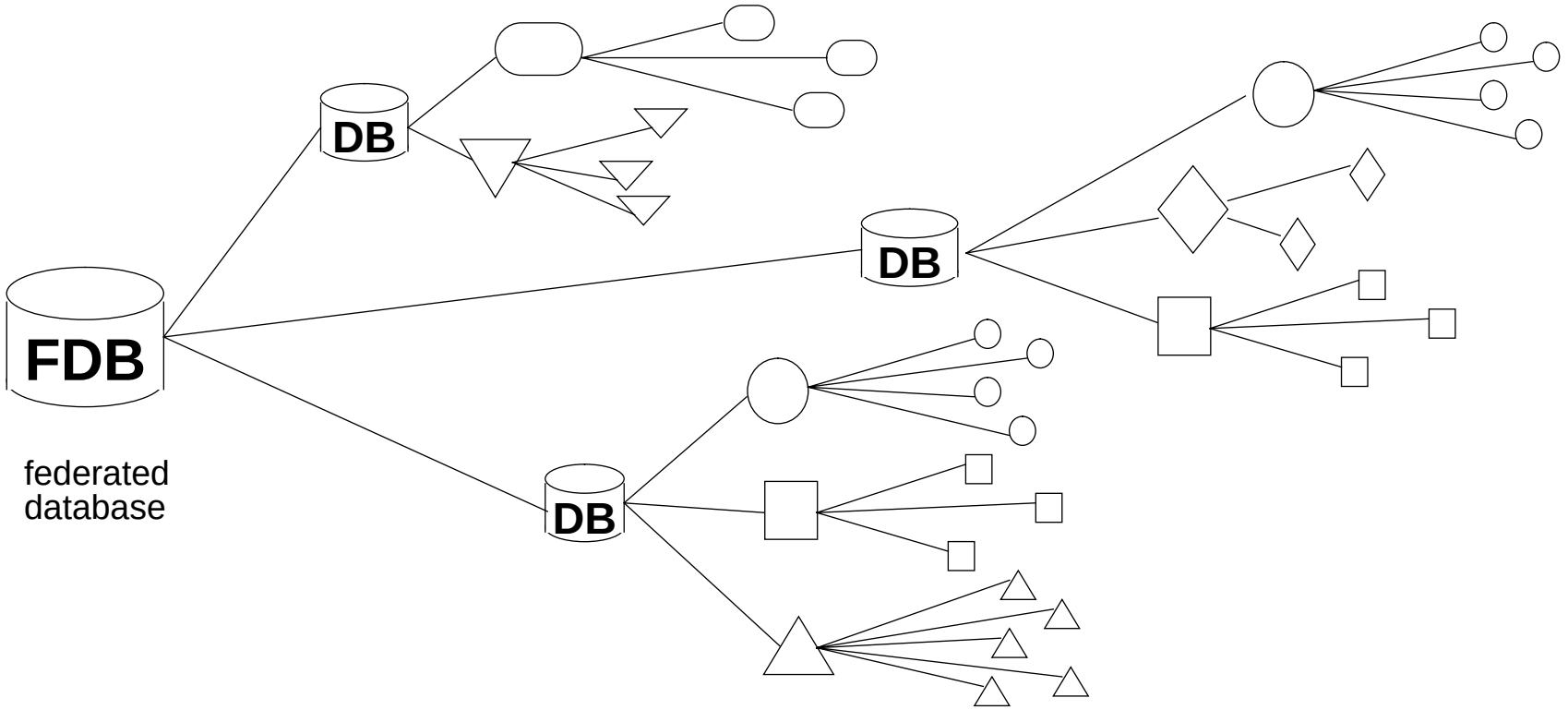
Type Checking

- **The problem:**
  - **the type of the operation is dynamically defined by query statement (in the context of its database schema).**
- **The solution:**
  - **the database should support a “describe” operation**
  - **... Input = query statement**
  - **...Output = type descriptor**

Trading for query interfaces - sample possibilities

- **Turn the trader into a full-blown data dictionary, containing**
  - **data models (with descriptions of semantics)**
  - **mappings to schemas**
  - **references to “connect” interfaces**
- **Stay using a minimal trader, containing**
  - **database names**
  - **references to “connect” interfaces**

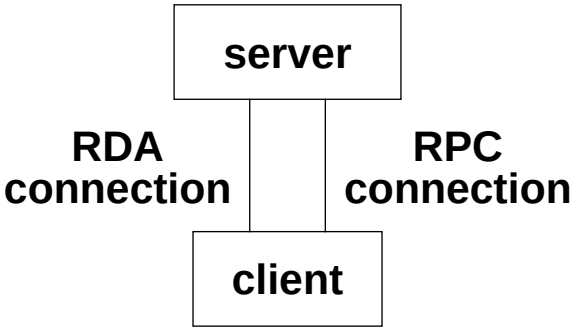
Distributed or Federated Database



Interim solutions - (1) Partial RDA/RPC Integration

- **Stored procedures**
  - **not suitable for non-database processes or services**
  - **not standardised (so far)**

- **RDA + RPC in tandem**
- **difficult to coordinate RDA thread and RPC thread from same client application at same server**



Interim Solutions - (2) RDA over RPC

- **Several possibilities...**
- **...RDA over RPC as transport protocol**
  - **Short term advantage: easy to adapt current products**
  - **Long term disadvantage: awkward and not well adapted to future object databases**
- **...RDA over dynamic RPC (as suggested by model presented here)**
  - **Architecturally “clean” and well adapted to future object databases**
- **...Hybrid (e.g., map X/Open CLI API to RPC)**
  - **All the disadvantages and none of the advantages of the above two approaches**

Summary

- **Use an object-based conceptual model for remote query language access**
  - **An elegant unification of RDA and RPC paradigms**
  - **A natural interpretation in context of relational databases and SQL**
  - **A solid foundation for future object databases**
- **A prototype has been built (as part of an enhanced trading system)**
- **For more information**
  - **on unifying RDA and RPC, see *Remote Database Queries in Object-Oriented Distributed Systems* (APM.1138)**