Database Connectivity

Considering Factors in DBC

- DBC Concept can extend to Current Situation as follows;
  - Data vs Contents
    - Translation of contents or data
  - API vs Service funtionality
    - Service request for server like DML or service funtionality
  - DBMS vs Service SW
    - Service processing engine like Query processing engine in DBMS
  - Database vs Meta-Database
    - How can handle the Meta-data for next generation
    - How can manage the media database for multimedia contents service
      - Query processing for semantic oriented query
      - Its based on ontology concept

Introduction of DBC

- Provide services that are covered by the translation layer
  - DB gateway: provide communication services

- A trade of middleware
  - extremely useful tools
  - enables easier access to data in heterogeneous sources
  - in future, combining with MOM or DCE to extend the functionality and retain market share

- Three key areas
  - the feasibility of comprehensive translation across diverse DBMSs
  - the performance problems associated with dynamic SQL
  - whether alternative approaches may offer greater flexibility and higher performance

Definition of DBC

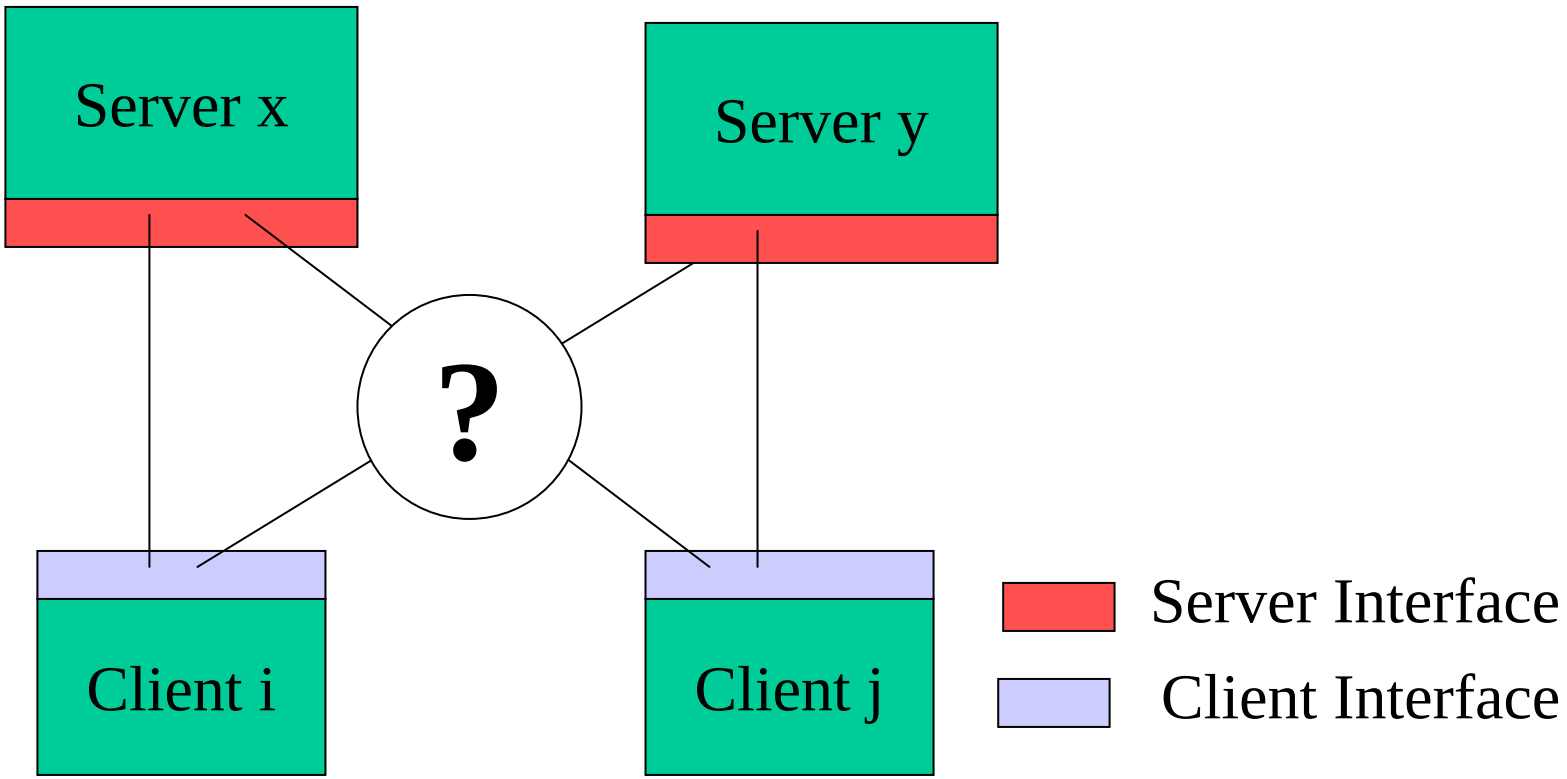
- Definition
  - an insulating layer between the application and the DBMS
  - uses one standard language to access all the DBMSs
    - translate the standard langauge to the various languages used by the target DBMSs
- Common application programming interface
  - the API of one of the DBMSs
    - adv: API is likely to be workable and pratical
    - dis: API depends on the DBMS vendor
  - a standard
    - adv: vendor-netutral
    - dis: API may not be updated with DBMS developement
  - a proprietary, purpose-built API developed by the middleware vendor
    - adv: the vendor can devise a language to handle all types DBMS
    - dis: depends on th middleware vendor’s product

Why is DBC SW needed

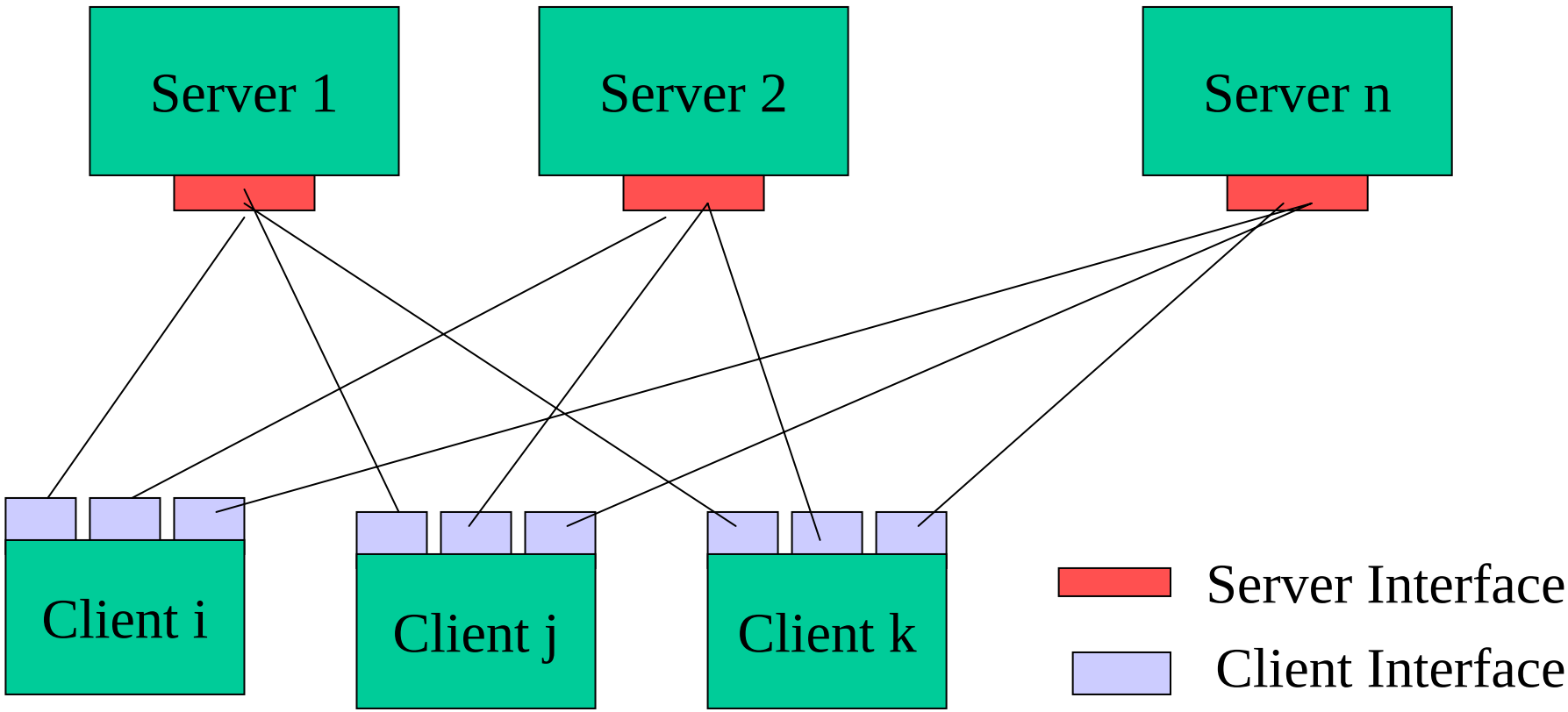
- Problems with Proprietary Interfaces
  - see Next two slides
  - Same problems are occurred in mobile environments

- Heterogeneous DB Access Approaches
  - see Next three slides

Problems with Proprietary Interfaces (1)



Problems with Proprietary Interfaces (2)

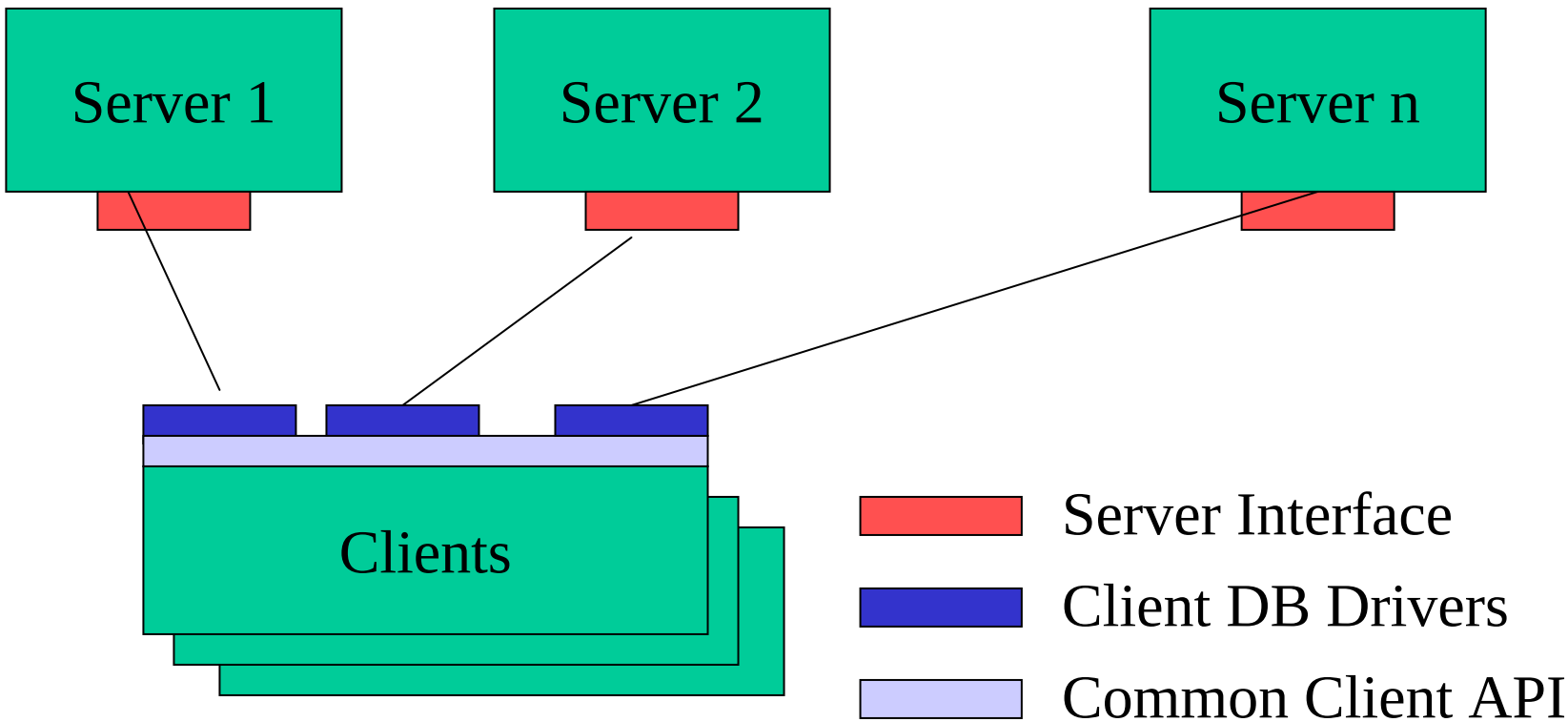


Connection Types = Client Types * Server Types

⇒ Leads to an impossible development/maintenance

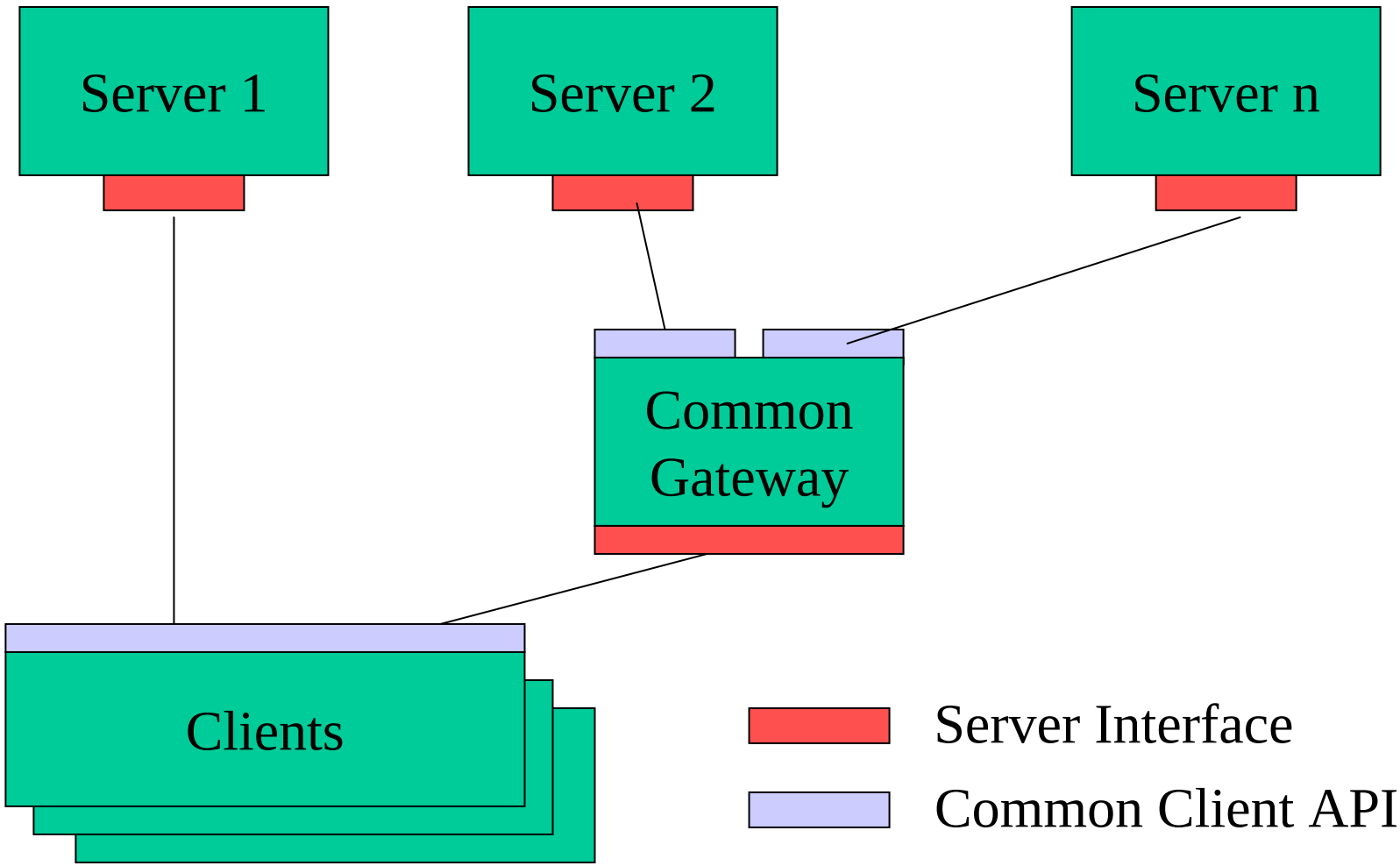
Heterogeneous DB Access Approaches (1)

1. Common Interface Architecture



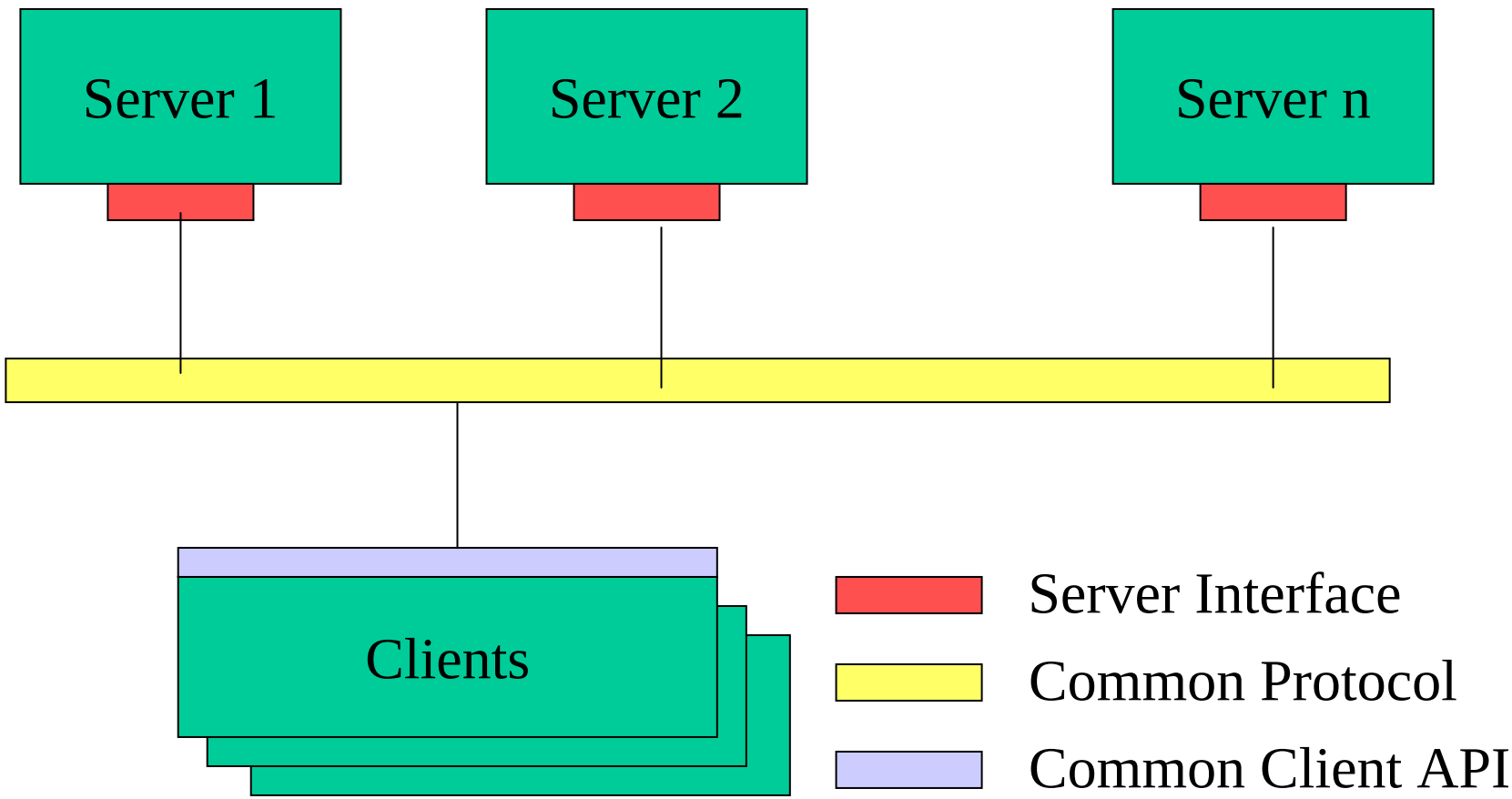
Heterogeneous DB Access Approaches (2)

2. Common Gateway Architecture



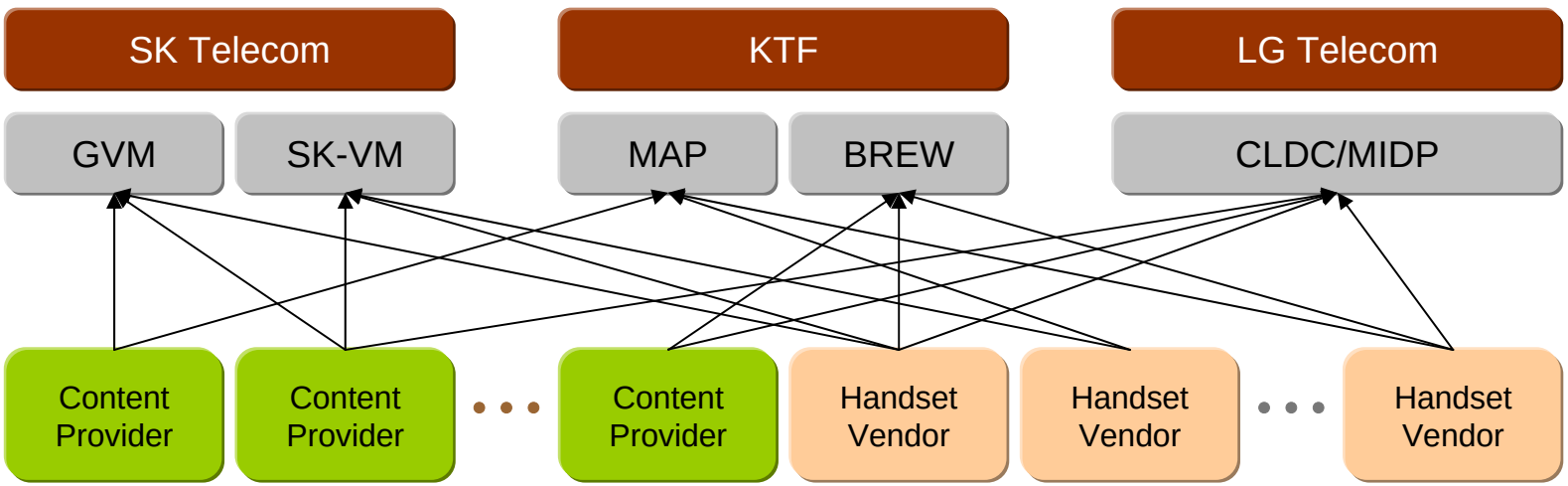
Heterogeneous DB Access Approaches (3)

3. Common Protocol Architecture



Mobile 환경의 다양한 플랫폼의 문제점

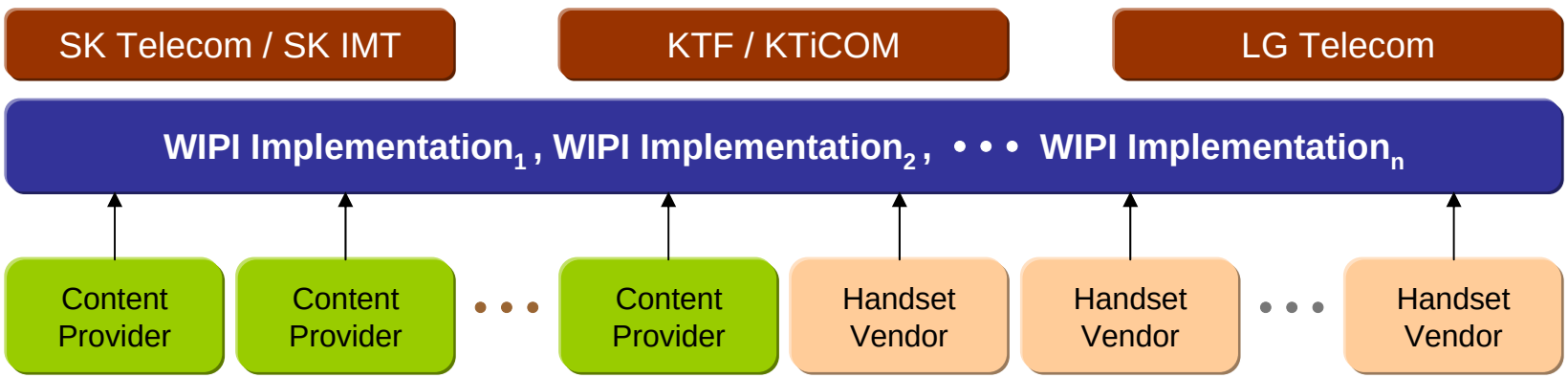
- 산업의 발전을 가로막는 장애요인
- 표준화 필요성 대두



응용 프로그램 실행 환경들이 혼재된 복잡한 서비스 형태

WIPI 의 주요 기능 (선택)

- 동적 API 및 Component 관리 기능
  - 다운로드를 통해 동적으로 API 및 Component(DLL)를 추가/갱신할 수 있는 기능
  - 다운로드 되면 바로 플랫폼에서 DLL를 활성화시킴으로써 즉시 적용되며, DLL 삭제 시 원상태로 회복되는 기능



WIPI를 적용한 환경에서의 서비스 형태

How does DBC middleware help?

- DBC MW
  - an insulating layer between the application and the DBMS
  - provides one standard API to access all DBMSs
  - the programmers only have to learn one DML

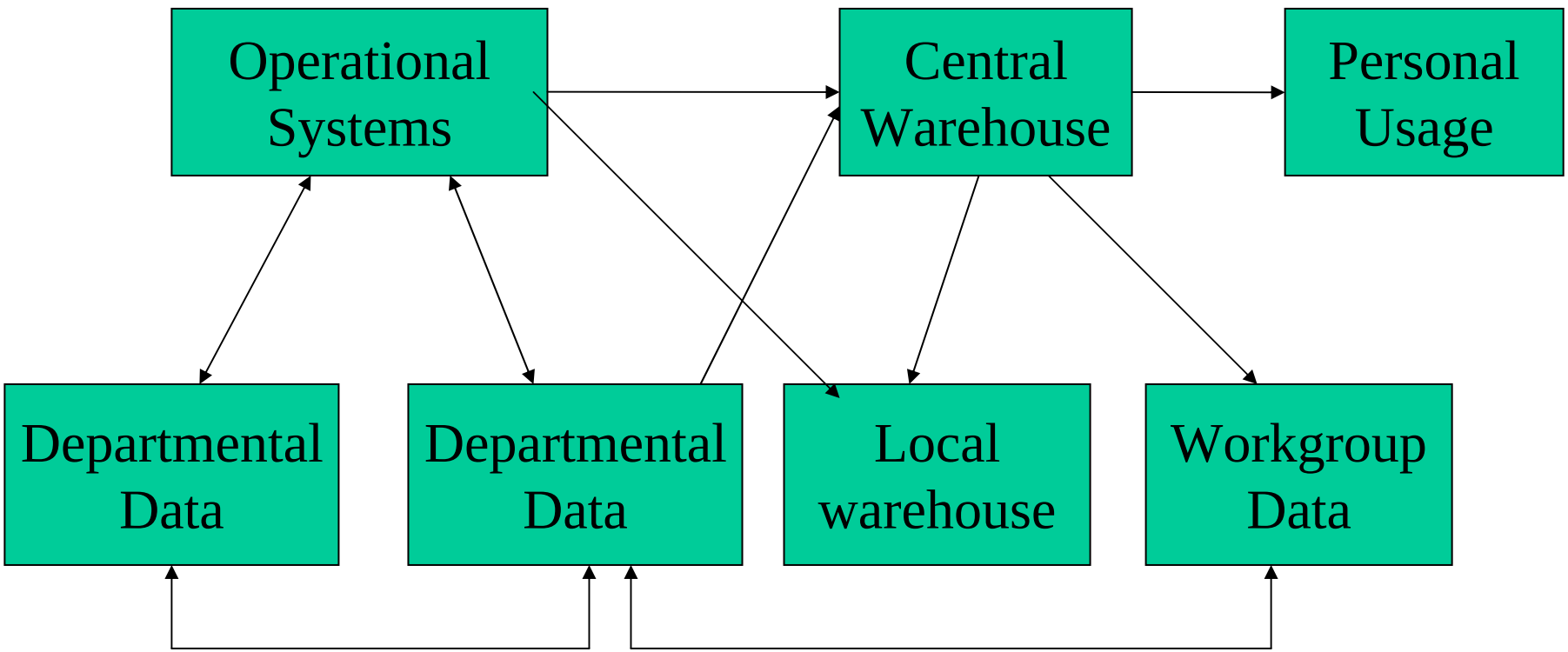
- Benefits of DBC (One API)
  - development times are reduced
    - » complexity and learning time are reduced
  - quality is improved
    - » less to make mistakes by one interface
  - costs can be reduced
    - » errors and learning time are reduced
  - systems are more adaptable
    - » the applications will require little change

DB gateways and drivers

- DB Gateway and drivers
  - see Next slides
    - moving problems
    - query drivers

Data Movement Problem

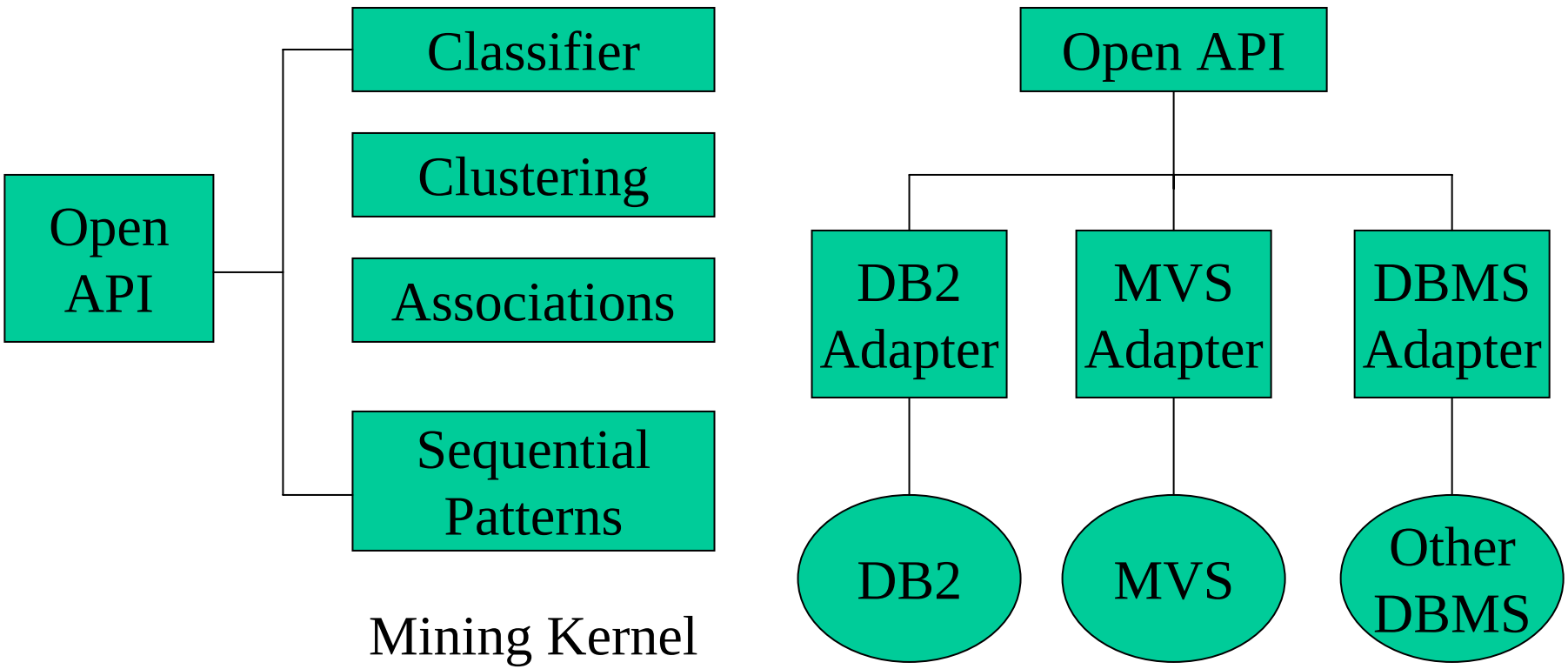
- Data is stored in multiple locations.
  - Client/server means data is generated everywhere
  - Demand for anywhere, anytime data
  - Limited window for data movement



Database Gateway Functions

DBC

- A DBMS gateway
  - identifies and translates DBMS differences
    - because of each DBMS has a proprietary technique to house data



Background to development of DBC

- Late 1980s
  - DBC appeared
- At 1990s
  - prompted for client/server computing
    - the number of types DBMS
    - proprietary languages
    - accessing to heterogeneous data
- Different types of DBMS
  - hierarchical model
  - network model
  - relational model
  - object-oriented model
  - extended relational model

The challenge of DBC (1)

- Two part to the process of translation
  - the translation of the names
    - usually handled via a directory or dictionary that contains the names
  - the translation of the DML
    - may seem a simple, but the complexities involved
      - the differences between the underlying models of DBMS
      - various approached to implement the fundamental run-time services
      - the difference in dialect between the supposed standards, SQL
- Translation middleware
  - cannot do all of these things
  - can never provide a truly transparent interface to all the data in all the DBMSs

The challenge of DBC (2)

- Translation between different DBMS models
  - based on different concepts
  - use a different DML and DDL
    - translating queries
    - translating updates

- Translation between the flavors of SQL and other standards
  - different methods of structuring commands
  - variations in syntax and semantics
  - data types
  - status codes and messages
  - collating sequences
  - stored procedures
  - explicit and implicit commit and rollback
  - nested transactions

The challenge of DBC (3)

- Translation between the ways of achieving the same services
  - a set of development services
    - to create schemata, allocate physical storage, allocate views and develop their application
  - a set of run-time services
    - ensure the data lose or corruption, the integrity, etc
      - integrity checks
      - locking and locking level
      - locking duration
      - deadlock detection
      - data dictionary services and system catalog
      - security services
      - global optimization
      - access across heterogeneous DBMSs

Standards (1)

- Standard can mean **a de jure** or **a de factor**
  - Microsoft's ODBC (Open Database Connectivity)
  - IBM's DRDA (Distributed Relational Database Architecture)
  - ISO/ANSI's RDA (Remote Data Access) standard
  - SQL Access Group's RDA
  - Borland's IDAPI (Integrated Database API) Architecture
  - Apple's DAL (Data Access Language) standard

Standards (2)

- Microsoft's ODBC (Open Database Connectivity)
  - see next 5 slides (22-26)

- IBM's DRDA (Distributed Relational Database Architecture)
  - specifically designed to link together IBM DB platforms
    - 4 different relational DBMSs
      - SQL/DS (DB2/VM, DB2/VSE), DB2, DB2/400 and DBM
    - one hierarchical IMS DBMS

ODBC (1)

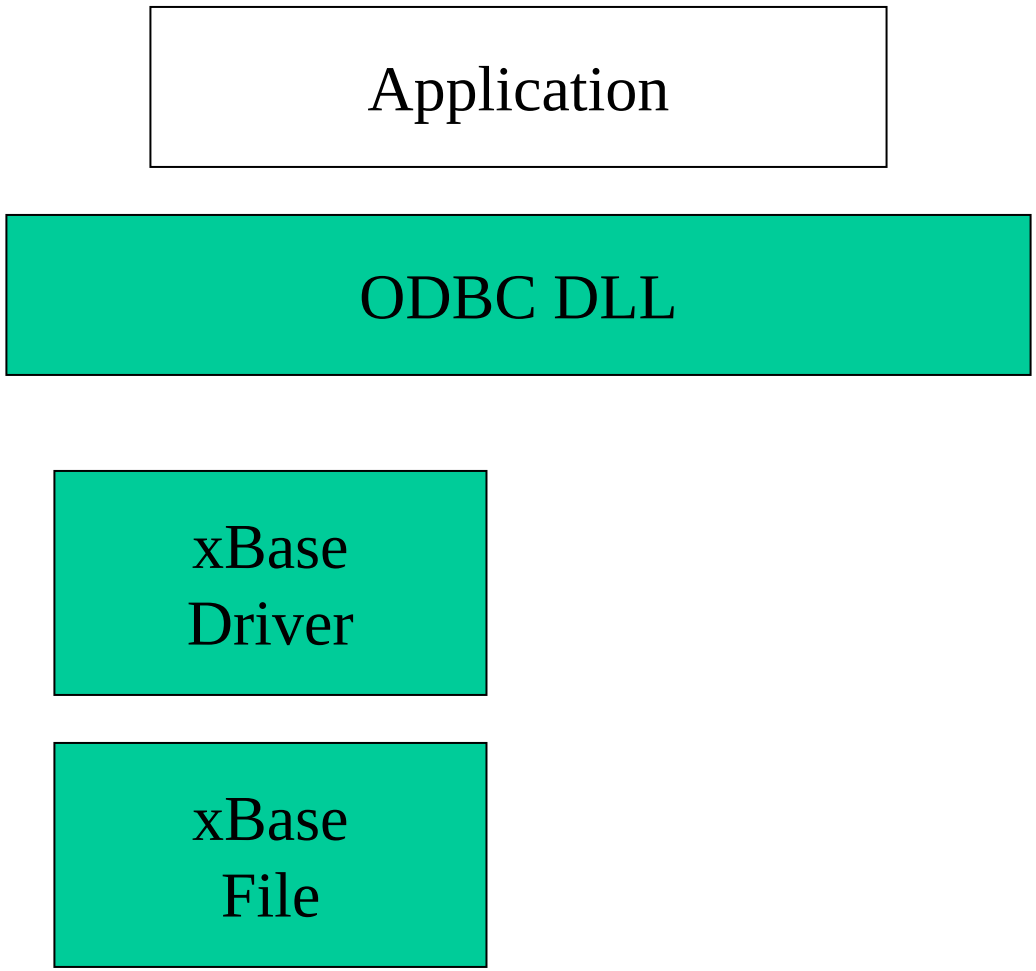
- What is ODBC (Open DataBase Connectivity)
  - a common interface approach to database connectivity implemented as a Call Level Interface
    - API for connecting to databases
  - compatible with other database connectivity approaches
    - broad industry support
    - eg) data access API in a version of Microsoft Windows
  - complements embedded SQL API
    - based on SQL access group (SAG) Common Language Inteface
  - initiating de jure standards process via ANSI

ODBC (2)

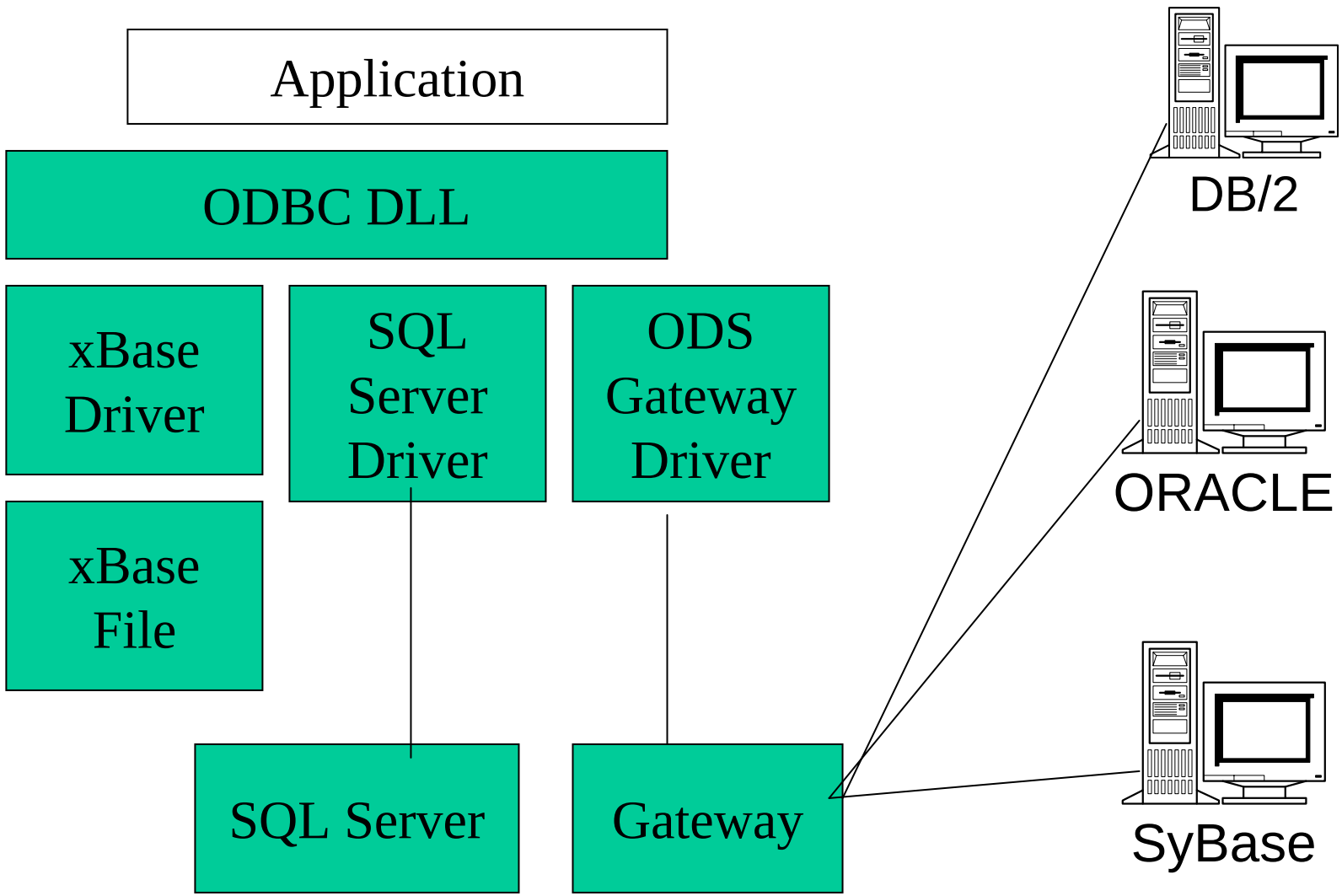
- Benefits of a Standard API
  - Universal database access
  - eases development burden
  - broadens application support for databases
  - Simplifies API confusion
  - Built-in scalability for applications

- Three tier Driver Model
  - One tier drivers
    - intelligent
    - typically for file access (dBase, Paradox)
    - “Engine in a driver”
  - Two and three tier drivers
    - modular
    - typically for DB servers, intelligent backends, gateways
    - “Passthru” logis

One-Tier ODBC Drivers

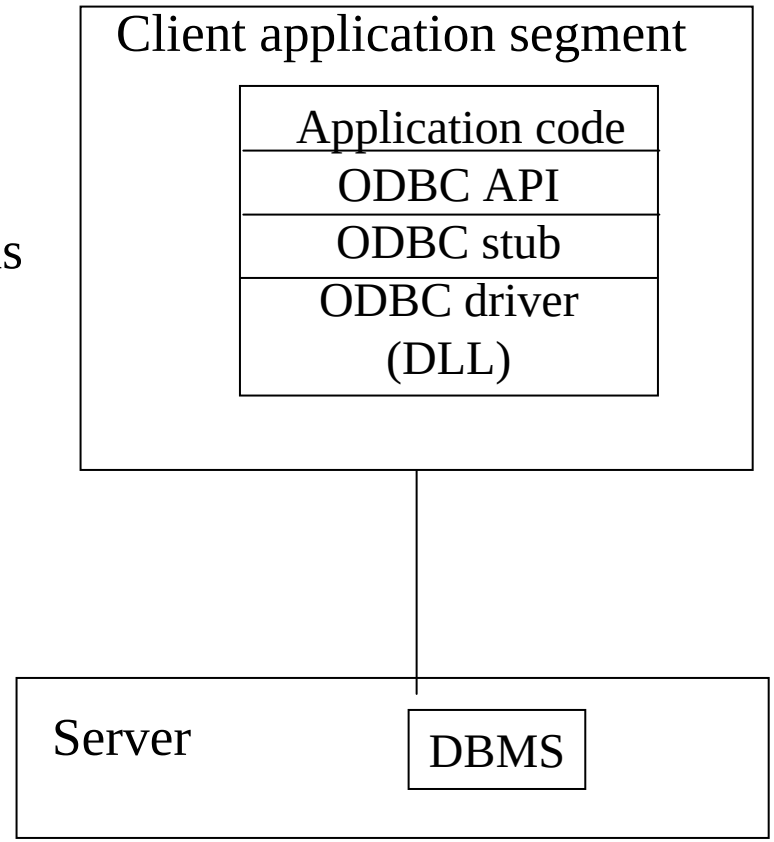


3-Tier ODBC Drivers



Middleware: ODBC

- ODBC API
  - Calls generated by application writer
  - Driver writer may provide API functions
- Driver is dependent on server DBMS
  - SQL server driver
  - Oracle driver
  - ODBC-compiler DBMS



Standards (3)

- ISO/ANSI's RDA (Remote Data Access) standard

  – initiated as an ISO/IEC standards effort of the WG3 in 1985

  – became an ISO draft international standard in 1991

  – is a full standard in now

    $\Rightarrow$ a multi-purpose standard

  – consists of two parts

    - the generic RDA (ISO/IEC DIS 9579-1)
    - the SQL specification (ISO/IEC DIS 9579-2)

    $\Rightarrow$ not only supports application to DBC, but also supports DB-to-DB connectivity

Standards (4)

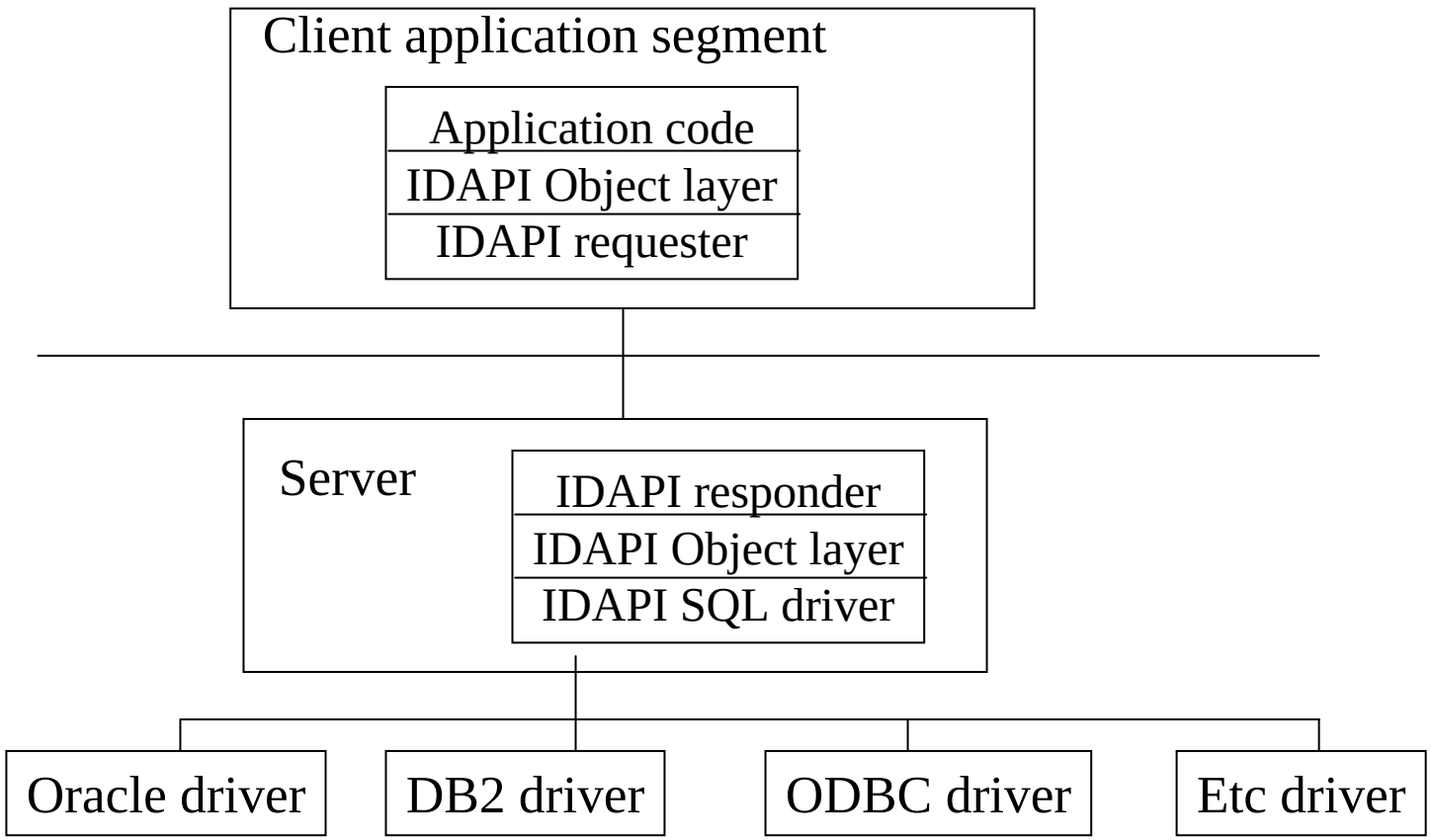
- SQL Access Group's RDA
  - SQL Access Group: founded in 1989
    - Digital, HP, Informix, Ingres, Oracle, Sun , Tandem, Teradata, and Unify
  - define and prototype DB interoperability and portability specifications
    - ISO's RDA and ANSI's SQL/SQL2
    - X/Open in 1994

- Borland's IDAPI (Integrated Database API) Architecture
  - developed by Borland, IBM, Novell and wordPerfect
  - based on SQL Access Group's CLI
  - see Next Slide

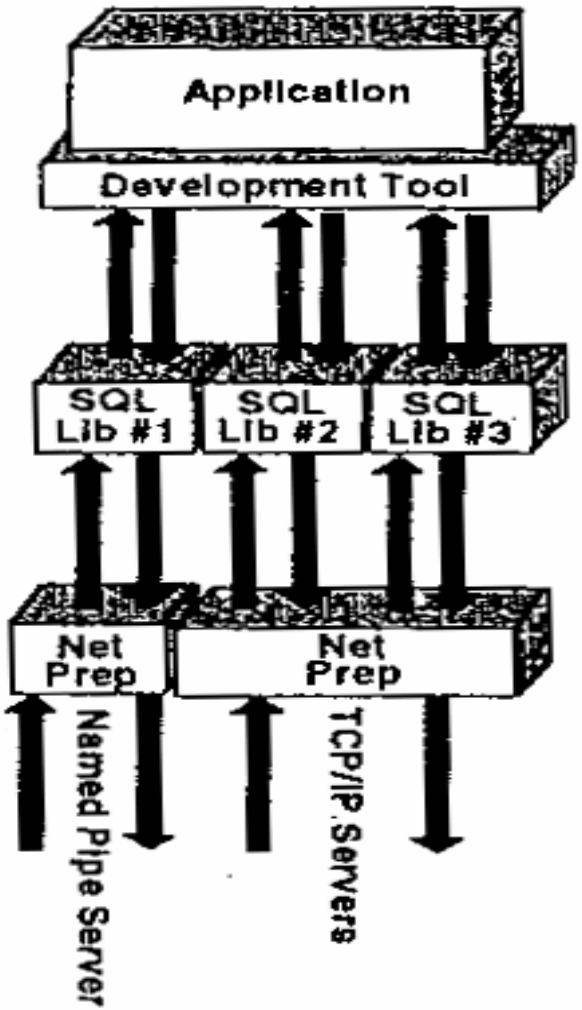
- Apple's DAL (Data Access Language) standard
  - an early pioneer of DBC and its Data Access Language
  - based on SQL Access Group's CLI

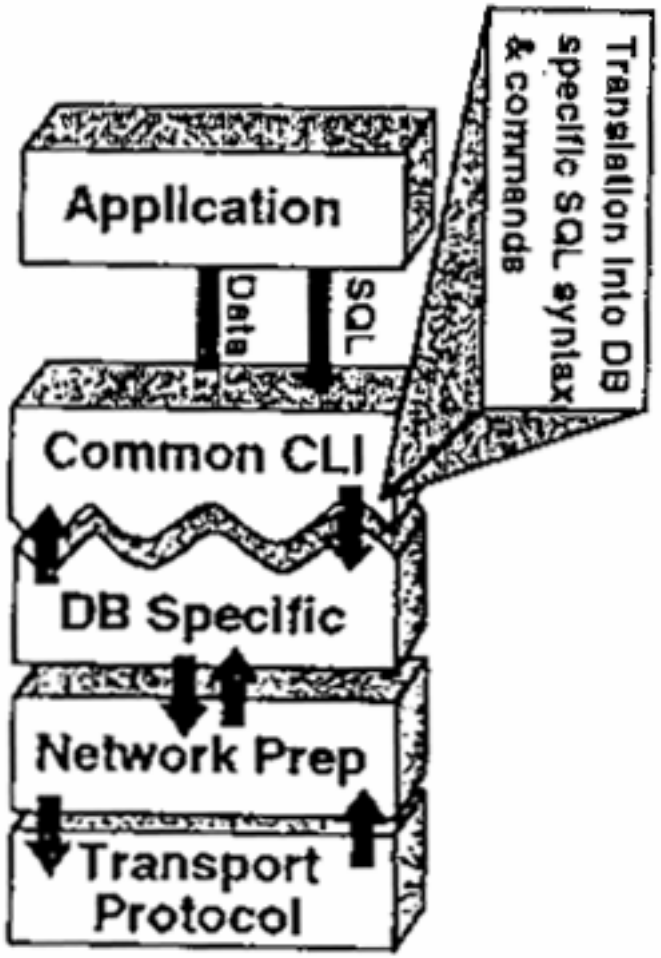
Middle-ware: IDAPI

- IDAPI: Integrated Database Application Programming Interface
  - Developed by Borland in 1992
    - based on SQL Access Group's CLI
  - A direct competitor to ODBC

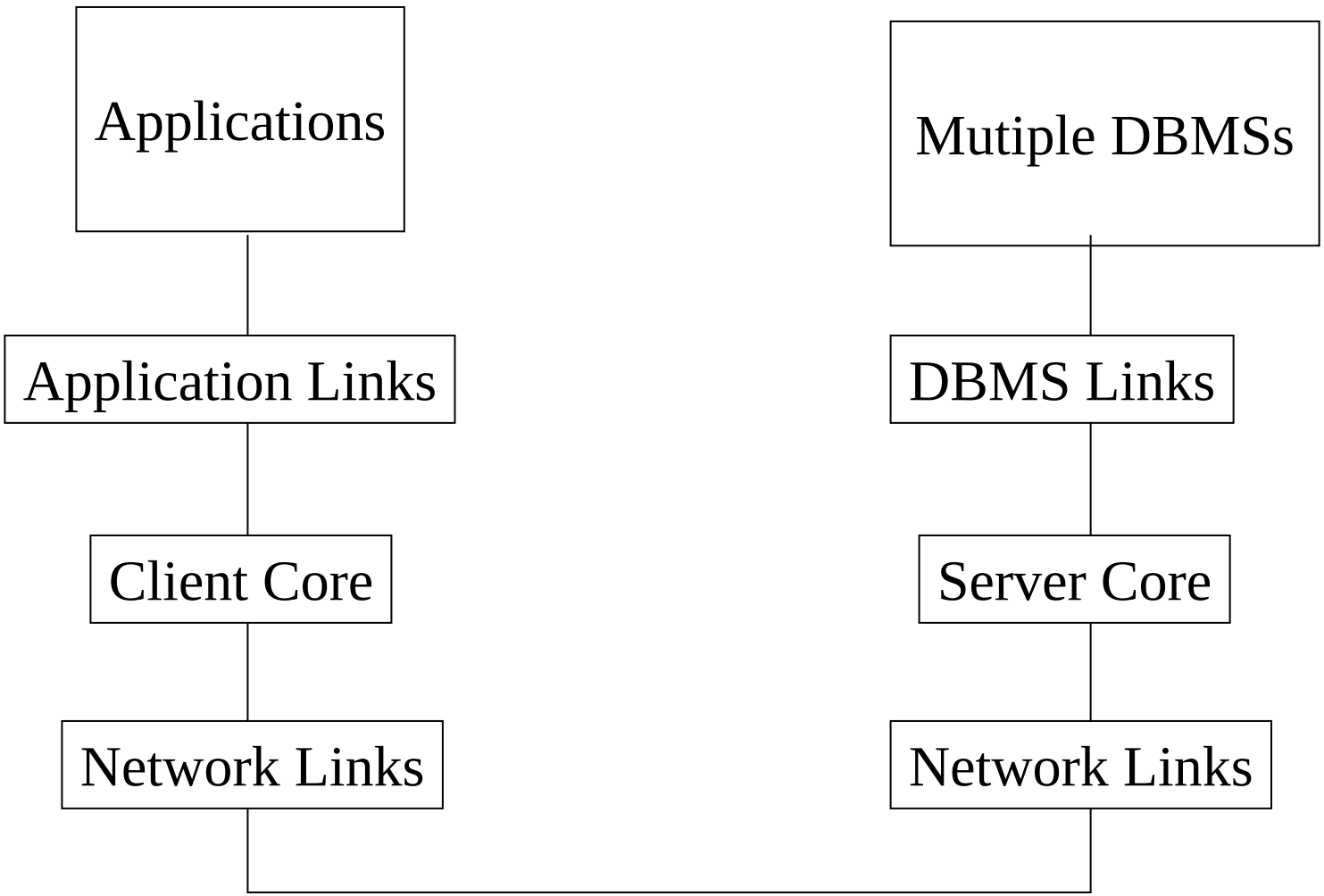


Middleware


Application
Development Tool
SQL Lib #1
SQL Lib #2
SQL Lib #3
Net Prep
Net Prep
Named Pipe Server
TCP/IP Servers


Application
Data
SQL
Translation into DB specific SQL syntax & commands
Common CLI
DB Specific
Network Prep
Transport Protocol

Flow of Data accessing Architecture



Types of product (1)

- Three types
  - database drivers
  - database gateway
  - distributed database environments

- Database drivers
  - translate from the API used by the user to the API of the target DBMS
    - don’t consider the communication over network
      - must be provided by other middleware or by the programmer
  - provide co-operating drivers
    - a developer can access multiple DBMSs simultaneously and therefore multi-DBMS joins
  - adv: a cheaper and more simple architecture
  - dis-adv.: don’t provide complete connectivity
      - need addtitional middleware

Types of product (2)

- Database gateway
  - be responsible for both translation and communication
    - adv: communication function
    - dis-adv.: do not cover all protocols of the underlying networks
  - resides on both the client and the server
  - three types of gateway
    - point-to-point gateways
      - are like a driver with its communication layer added
      - enable the developer to access one DBMS using standard DML
        - » example: accessed using ODBC
    - SQL gateways
      - enable to access to multiple R-DBMS via a single API
        - » this API may be standard or proprietary and provide for heterogeneous joins
    - universal gateways
      - provide access to all types of DBMS via a single API

Types of product (3)

- Distributed DB environments
  - based on ORBs, DCE, MOM, or other communication layer product
  - support translation and transport to and form the DBMSs
  - support management services such as guaranteed delivery, time services and security services

  - two pioneers in using ORB-type technology
    - Micrsoft: OLE custom object (OLX)
      - provide an OLE wrapping around DB drivers and support connection to data sources that ODBC does not support
    - Oracle: Object
      - is a similar product being planned by Oracle
      - provide an ORB-type management layer around its gateways and driver products

Adv. And Disadv. With DBC (1)

- Advantages
  - improve the quality of applications
  - save effort and money

- Considering some issues
  - Transparent translation
  - Performance
  - Solving the wrong problem
  - Other solutions to give to transparency

- Is transparent translation feasible?
  - DB translation
    - the complexity of translation between DBMSs with different philosophical roots
      - non-procedural or navigational
    - the difficulties of dealing with stored procedures
    - the complexity of dealing with new sorts of data
    - locking and deadlock protection

Adv. And Disadv. With DBC (2)

- Performance is an issue
  - Dynamic: translation at run-time may slow processing
    - automatically handles as long as the correct version
  - Static: translation at compiling time may fast and efficient
    - any changes may not be reflected in all the applications
  - performance degradation
    - in dynamic case, large DBs is not trivial
    - in static case, limited to only certain environment

- Are we solving the worng problem?
  - The developer will still need to spend the time trying to find the data
  - Middleware will only save fractions of tuime

Adv. And Disadv. With DBC (3)

- Other solutions
  - can be based on RPC/DCE or MOM
    - calls the sub-routines and receives the requested data in reply
    - ignore the use of tools ; GUI builders and 4GLs
  - advantages
    - performance can be tuned
    - database change does not affect the application
    - change to code need only be made once
    - the code can be located near the data it accesses, reducing network traffic
    - the programmer does not need to know where the data is
    - the access will benefit from functions of RPC and MOM
      - management servicesL guaranteed delivery, security, etc
  - disadvantages
    - specialisy knowledge of the DML is still needed
    - the specialists can become a bottleneck unleess their time for mananing

When to use (1)

- Should be used when:
  - you have data spread across many DBs and DBMSs to need to access **using a single API**
  - the accesses you require **do not have critical response times**
  - the accesses required **do not require complex** joins accessing many DBMS and fast response times
  - the DBs **contain normal structured information** in field; do not contain images or other BLOBs
  - the information is reasonably **well structured** with minimal duplication, clear naming of fields, easy-to-understand records or tables
  - you **have the staff to test** the middleware to make sure it works effectively
  - you recognize that **total data transparency is not possible**

When to use (2)

- Should not use DBC for:
  - heavy-duty updates using transactions and requiring the use of locking
  - very large databases
  - applications that require large cross-database joins
  - applications that make extensive use of stored procedures
  - applications requiring fast response times

$\Rightarrow$ may use RPC or MOM based middleware to support these cases

Product Examples

- Common Interface
  - Apple : Data Access Language
  - Borland: SQL-Link
  - IBM: EDA/SQL
  - Lotus: DataLens
  - Oracle: SQLnet
- Common Gateway
  - Microsoft: Open Database Server
  - MDI: Database Gateway
- Common Protocol
  - IBM: Distributed Relational DB Arch
  - ISO/ANSI: Remote Database Access