Database cooperation: classification and middleware tools

Paolo Atzeni
Universita' di Roma Tre, Italy

in cooperation with
Luca Cabibbo and **Gianni Mecca**

[atzeni,cabibbo,mecca]@inf.uniroma3.it
http://poincare.inf.uniroma3.it:8080

Outline

- introduction
- classification criteria
- classification
- architectural techniques
- architectures and middleware tools
- conclusion and discussion

- introduction
- classification criteria
- classification
- architectural techniques
- architectures and middleware tools
- conclusion and discussion

Focus

- database cooperation
- bridge the gap between methodologies and tools
- classification of applications
- implementation architectures

Specific framework

- contract with AIPA (National Authority in Italy for Computing in the Public Administration)
- AIPA is promoting a National network: "the Internet of the Italian government and administration offices"
- AIPA wanted to give guidelines to the various branches on how to exploit the network and promote cooperation

General framework

- the need for integrating applications arises everywhere, stimulated by the development of networks
  - integration (or interaction) of components developed separately, for technical, organizational, or temporal reasons
  - cooperation of previously separate processes (process reengineering)
  - cooperation of independent companies (or merges thereof)

Levels of interaction

- **connectivity**: systems and networks exchange packets of information (in Internet, with TCP/IP)
- **interoperability**: systems and networks interact by means of standard services (in Internet, standard protocols above TCP/IP)
- **cooperation**: applications over different systems interact with one another; at the extreme level, distributed applications coordinate existing local applications

Interoperability services

- file transfer (ftp)
- virtual terminal (telnet)
- electronic mail (X.400 or ESMTP/MIME)
- directory service (X.500)
- WWW (http)

Cooperation of Information Systems

- requires that the cooperating systems offer application services
- it happens when systems make use of application services offered by other systems
- cooperating systems "share goals"
- a major feature: "continuous change"

- introduction
- $\boxed{\text{classification criteria}}$
- classification
- architectural techniques
- architectures and middleware tools
- conclusion and discussion

First level classification

two forms of cooperation (no clear cut, indeed)

- **data-oriented**: data in a system is visible/accessible to other systems
- **process-oriented**: systems offer services, exchange messages, (or data, documents), trigger activities

Process-oriented cooperation

- exchange of messages and documents;
  simple cooperation; satisfied with interoperability services
- structured exchange of messages and documents;
  more complex cooperation; satisfied with EDI services and sophisticated mailers
- cooperating processes: composed of activities of independent subjects that cooperate;
  complex cooperation; solution (?): workflow management systems

Cooperation: requirements

actual cooperation involves systems that are

- distributed
- heterogeneous
- autonomous

Distribution

- cooperation means that we have multiple systems;
  data cooperation means that we have multiple databases that handle data
- as opposed to usual distributed databases, here distribution is not a design decision, but a fact, due to the preexistence of the cooperating databases
- distribution may range from different databases on the same machine to databases spread over a geographic network

Heterogeneity

many aspects

- differences in the computing environment (hardware, operating system, network software)
- differences in the database management system:
  - different data model (relational, hierararchical, OO, files, ...)
  - details in the same data model (versions of the relational model: types, constraints, ...)
  - different languages (SQL and QBE, versions of SQL, ...)
- semantic heterogeneity: differences in the meaning of data

Autonomy

Absence of a common (or coordinated) control over the various systems: “distribution of control”

Technical aspects:

- design autonomy: the various systems are built independently, with different choices for many aspects (thus inducing heterogeneity)
- service autonomy: decision on if and how cooperation is established (what services are offered)
- execution autonomy: cooperation does not interfere with “private” operations; cooperating operations are executed under local control

Distribution, Heterogeneity, Autonomy

Classification criteria?

No, since in the cooperation we should be ready to tackle highly distributed, heterogeneous, and autonomous systems

Constraints?

Probably yes, at least in the worst case

Deficiencies to overcome?

Possibly, if there is some coordinating authority: cooperation can stimulate reengineering (reducing the degrees of heterogeneity, autonomy, and distribution)

The "ideal goal"

- complete database features in an integrated framework (in order to develop efficient/effective cooperative applications):
  - "integrated database"
  - query and transaction capabilities
  - absence of redundancy (and inconsistency)

however current technology does not really support this (at best requires effort and money)

New classification criteria for data-oriented cooperation

- degree of transparency of component data
- complexity of operations
- level of liveliness (or up-to-dateness, as opposed to obsolescence or latency) of data

Transparency

measures the need for hiding distribution and heterogeneity of component systems in a data-oriented cooperation

+ integration of component databases:
  the cooperative application needs to see one (virtual) database, offering an integrated schema (or set of functions)

– it is acceptable that each component database offers a set of services:
  each cooperative application is responsible for accessing, integrating, transforming the various pieces of data

Complexity of operations

measures the need for coordination in the execution of operations (queries and transactions)

+ complex operations (queries and updates): join of large relations (from different databases) or transactions with multiple updates in different databases;
  require nontrivial management

- simple operations (for example read-only, or local);
  do not require specific support

Liveliness of data

measures the need for actual availability of current data

+ on-line access to the primary copy of data: "access of actual data where it is"—the original goal of integrated databases?

- access to copies, with a controlled degree of obsolescence

Note: this criterion can be applied independently to the various components

- introduction
- classification criteria
- classification
- architectural techniques
- architectures and middleware tools
- conclusion and discussion

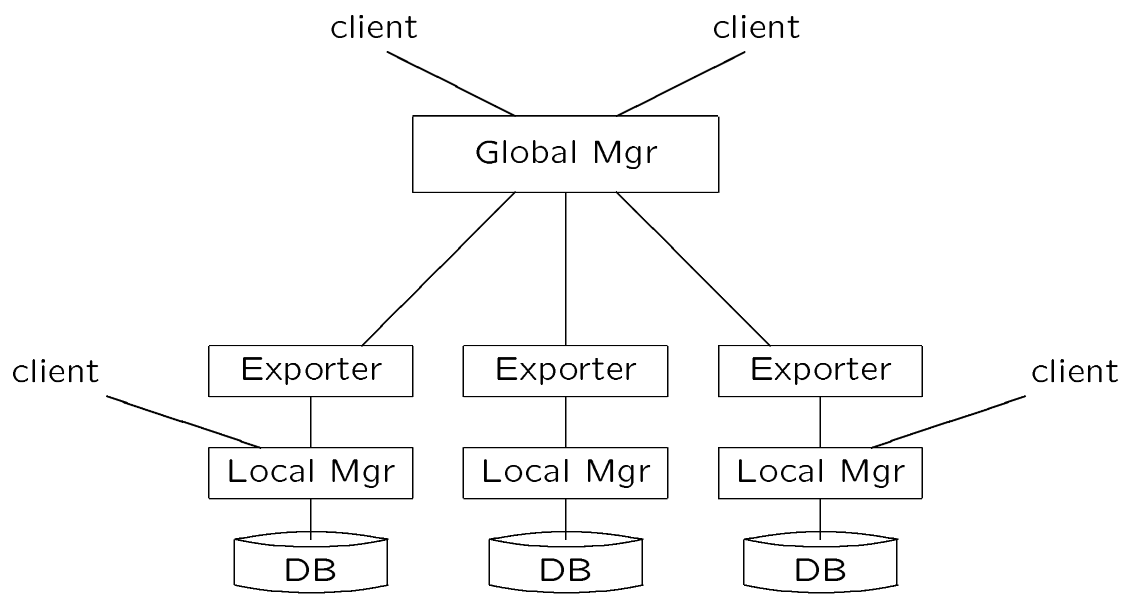
A classification for data-oriented cooperation

- there is no need to consider all combinations:
  - the criteria are not independent:
    a high degree of complexity requires an integration infrastructure, that is, it requires a high degree of transparency;
  - some cases are marginal or subsumed
- major classes
  - **multidatabases**: transparency, complexity, up-to-dateness
  - **data warehouses**: transparency, complexity
  - **local information systems with external data**: varying degree of up-to-dateness

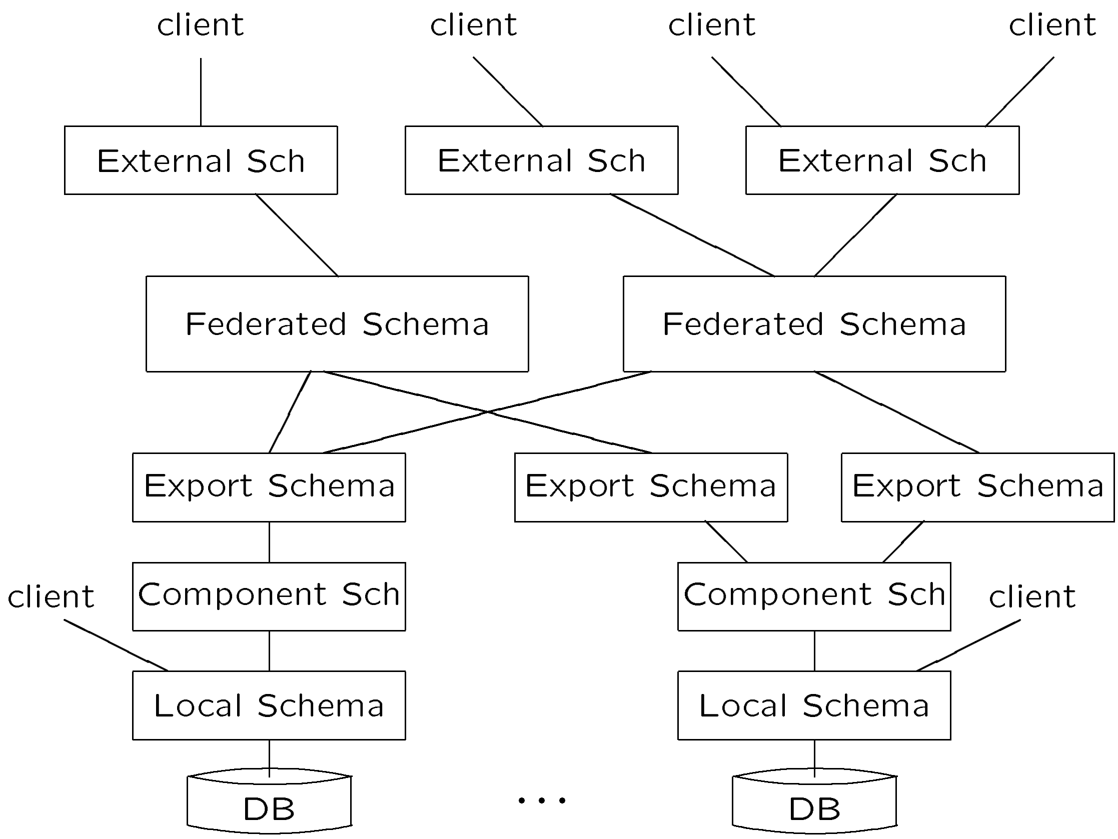
Multidatabases

- in the ideal case, there is a high degree of transparency, complexity, up-to-dateness
- there is a **global** system that integrates services:
  live data is directly accessed in a transparent and efficient way
- methodologies and tools for integration (of schemes, data, languages) are needed

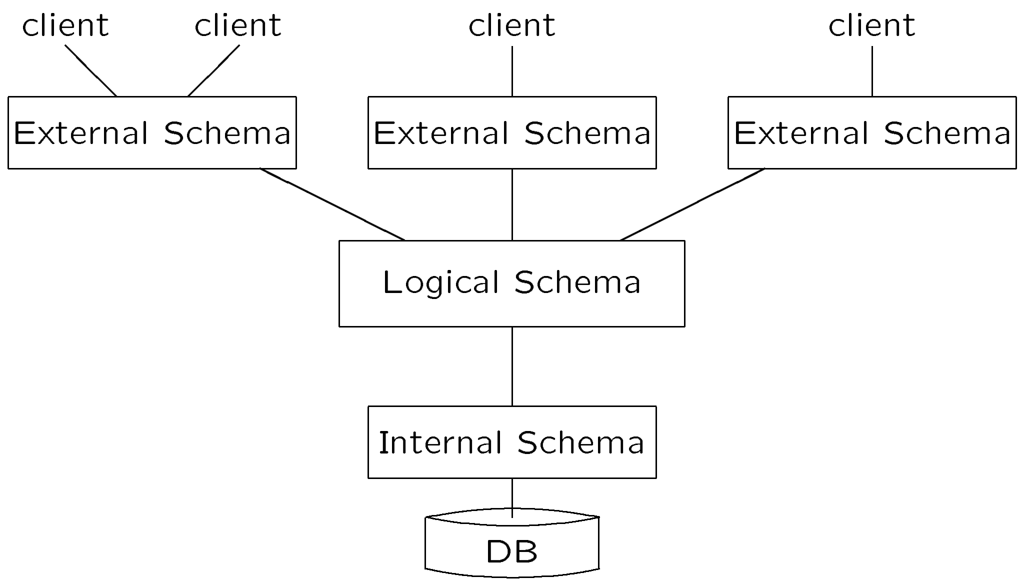
Multidatabase



Five-level architecture for federated database systems



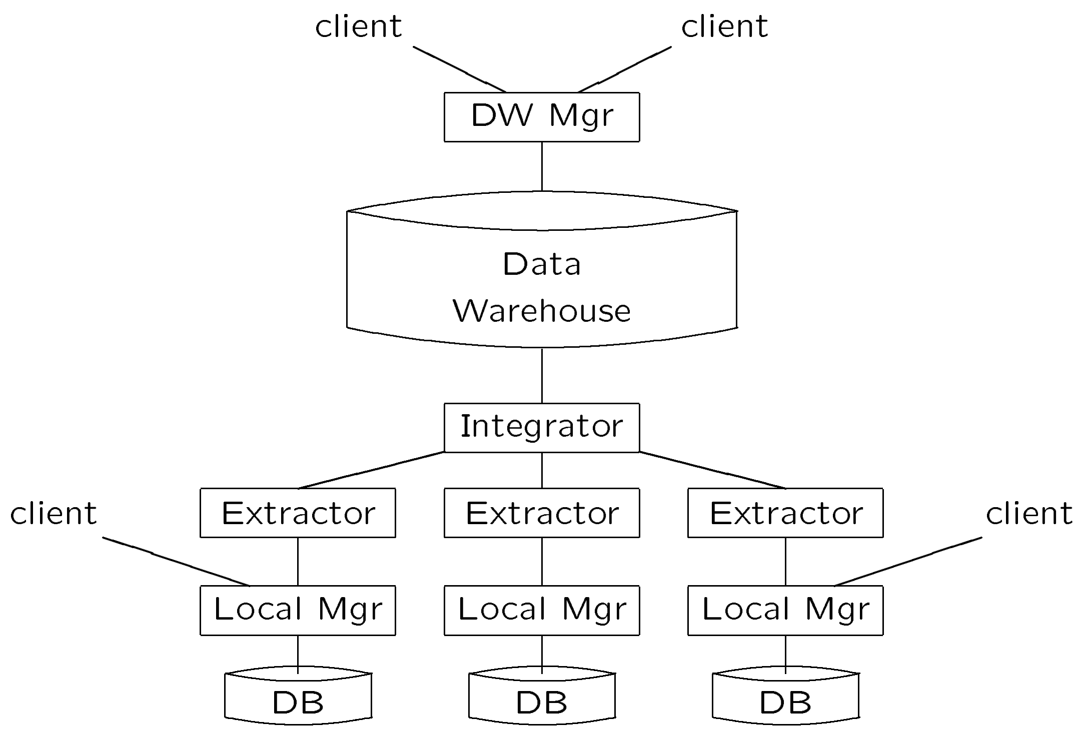
Three-level architecture for traditional database systems



Data Warehouse

- data are integrated off-line and stored in a new database (the **data warehouse**)
- typical applications: decision support (for marketing, sales, financial analysis), investigation, summarization
- great interest in the marketplace (OLAP, data cube, multidimensional databases, extractors, replicators)

Data Warehousing approach



OPERATIONAL DB	DATA WAREHOUSE
application oriented	subject oriented
serves the operation level	serves the managers level
detailed data	also summarized, with time series
accessed a unit at a time	accessed a set at a time
nonredundant data	redundant data
current	values over time and/or time series
varies with time	is not updated
used standardly	used casually
requirements known a priori	not known a priori
performance sensitive	performance relaxed
transaction driven	analysis driven
concurrency control needed	no need
high availability	relaxed availability
static structure	flexible structure
small amount of data per op	large amount of data

Advantages of Warehouses wrt Multidatabases

- on-line access in query processing can be slow:
  it competes with operational activities
- primary sources may be unavailable
- complex restructuring and aggregation may be needed
  (and primary sources may be heterogeneous)

Advantages of Multidatabases wrt Warehouses

- sometimes access to current (primary) data is essential
- primary data may change rapidly
- can support unpredicted queries

A proposed architecture for DW

- operational data
- enterprise warehouse data (granular, subject oriented, historical)
- “Data Marts:” departmental data (aggregated, theme oriented)
- individual (temporary, ad hoc, problem-based)

Little redundancy among the levels.

(an aside)

Data Mining

Two goals over large (derived) sets of data:

- find relationships
- make forecasts

Data mining is useful for providing information of various types:

- classes: group objects on the basis of properties
- clusters: “unpredicted” categories
- associations: correlations between different events
- sequences: specific patterns, for example in time-series
- forecasts
- similar sequences

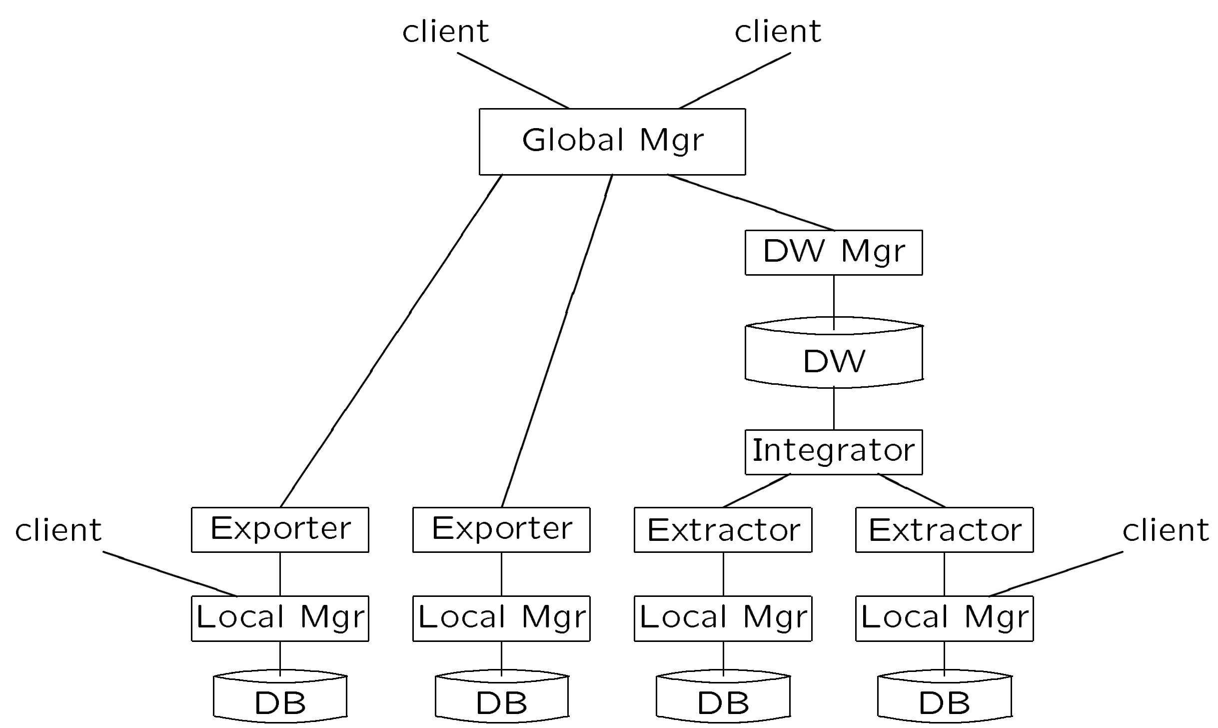
No clear cut

In a complex system there may be

- data whose up-to-dateness is essential
- data whose primary copy is expensive to access (wrt to the actual need for up-to-dateness)
- data that are always aggregated in the same way

Therefore, we could need an integration of replicated and primary data

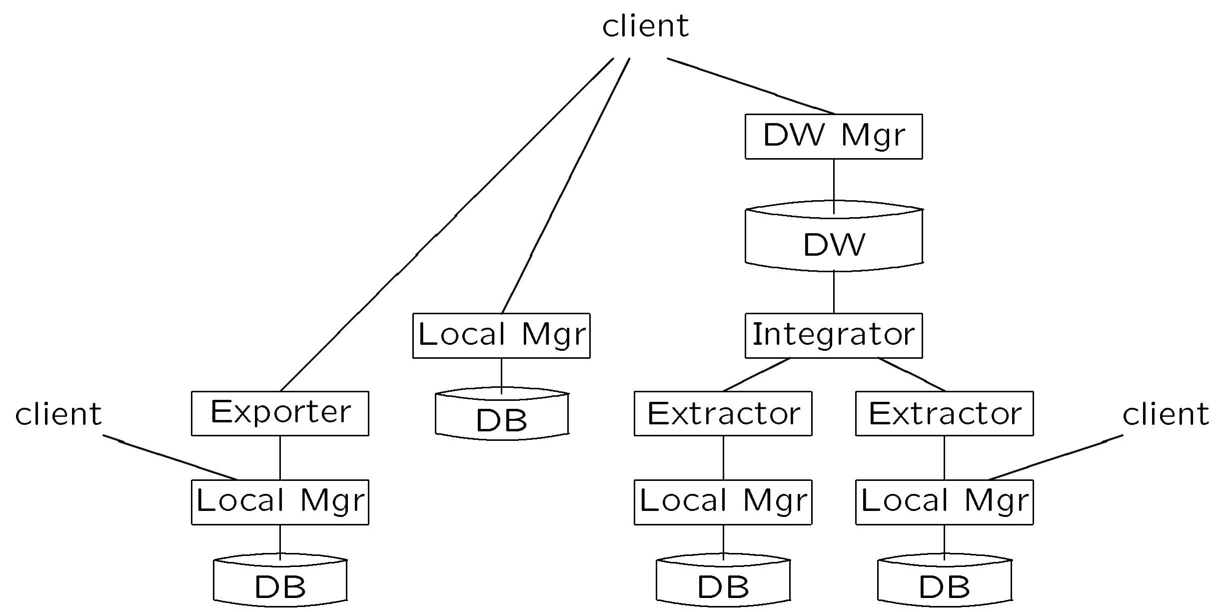
An intermediate solution



Local Information Systems with external data

- useful if a system has to access exported data from another system
- the application has to include the management of integration, translation, access control
- meaningful only with simple operations
- data can be primary or replicated

The application does the integration



- introduction
- classification criteria
- classification
- architectural techniques
- architectures and middleware tools
- conclusion and discussion

Architectures for data-oriented cooperation

- multi-level client-server architectures
- with middleware tools for
  - complex integration or
  - basic cooperation interfaces

Two-level Client/Server architecture

- **client**: presentation (interface, graphics, some local processing)
- **server**: application logic and data access

Three-level Client/Server architecture
(basic idea)

- **client**: presentation (interface, graphics, some local processing)
- **intermediate server**: application logic
- **back-end server**: data management

Benefits of three-level Client/Server architecture

- the application program, the specific and delicate part of each application, is separated from both UI and DB
- the DB is not exposed
- scalability and flexibility of components: no thin/fat client doubt
- modularity: reasonable encapsulation of legacy applications

Multi-level Client/Server architecture

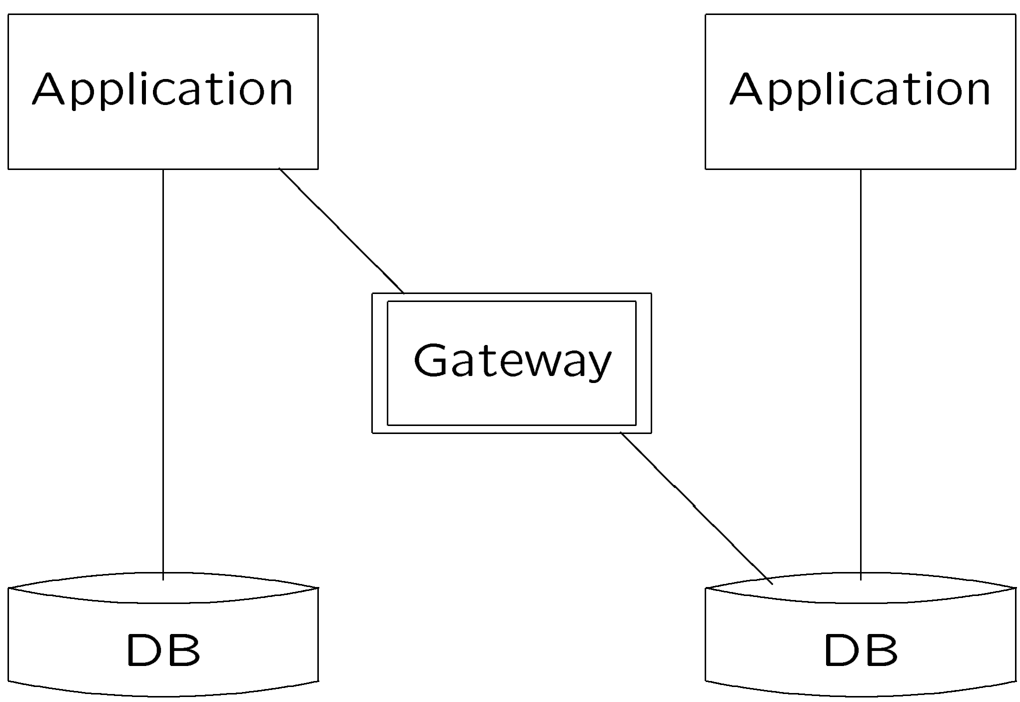
- **client**: presentation (interface, graphics, some local processing)
- **intermediate server**: encapsulation (on an open system) of heterogeneous (possibly legacy) applications
- **back-end server**: the encapsulated application (with the data server)

There can be multiple, intermediate servers (and even the back-end server may be a C/S system)

Basic techniques for Client/Server data cooperation

- on-line data transfer
- off-line data transfer
- on-line message exchange
- off-line message exchange
- on-line data access

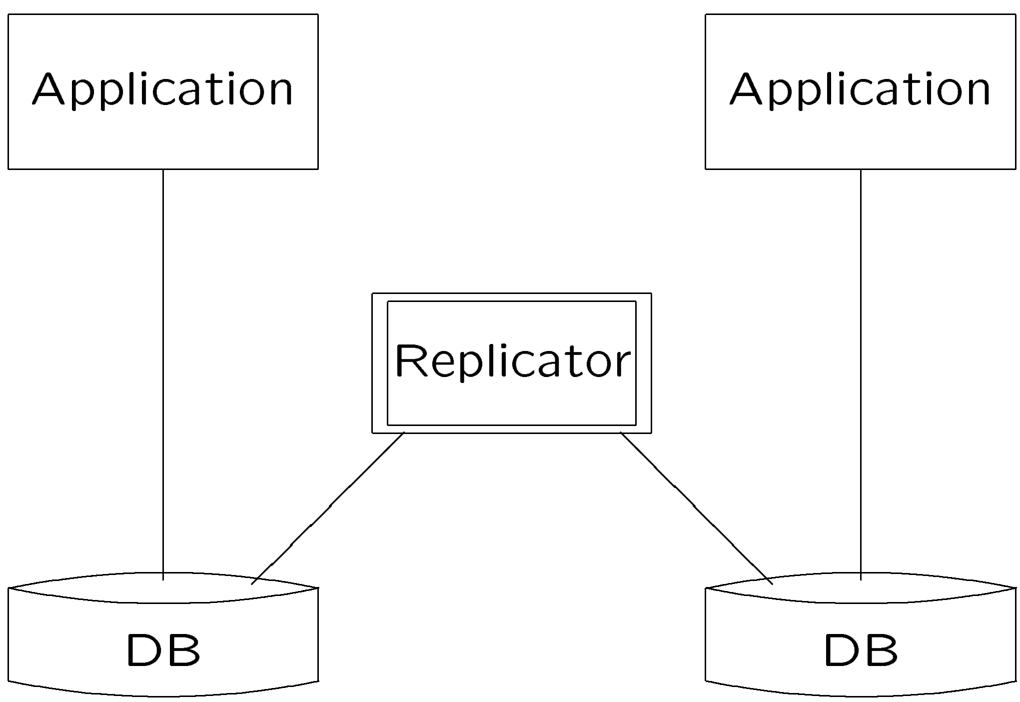
On-line data transfer



On-line data transfer: Gateways

- allow applications for one database to access data over another database
- there are different levels of transparency
- typically, the client makes use of SQL
- available in the relational world and to access legacy DBs from relational applications
- flexible (if authorized, allow access to the whole DB), although some tools allow read-only access
- rather inefficient (the server has to execute casual queries)

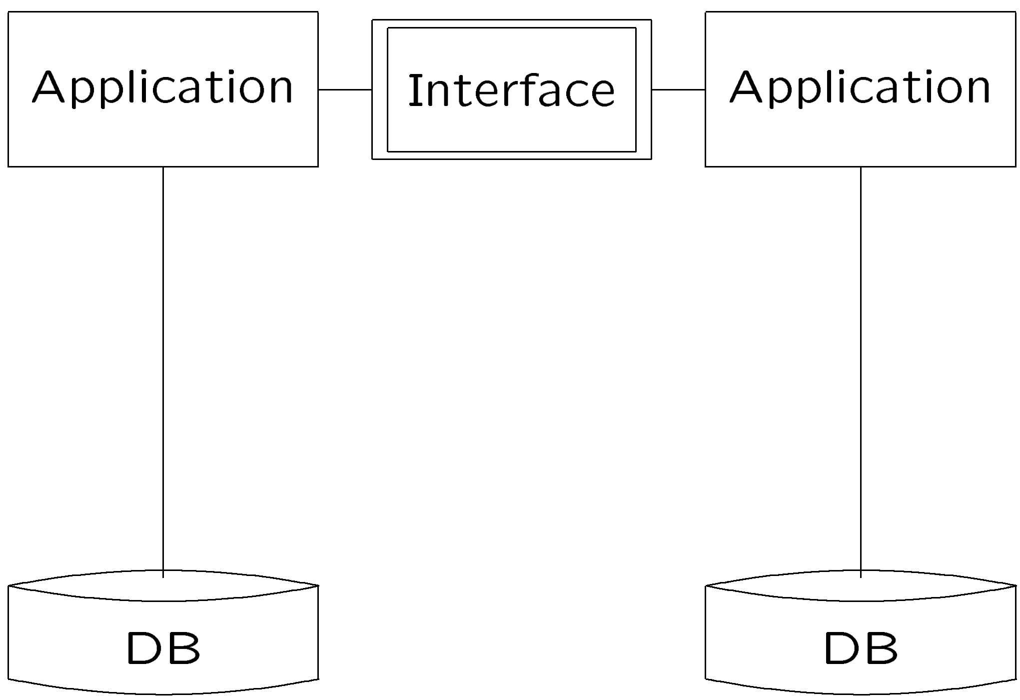
Off-line data transfer



Off-line data transfer

- data are extracted from one DB, transformed, and stored in another
- ad-hoc solutions have been used for decades
- recent interest in replication and warehousing tools
- tools for
  - extraction (incremental, with change detection)
  - translation, integration, cleaning, aggregation
  - OLAP processing

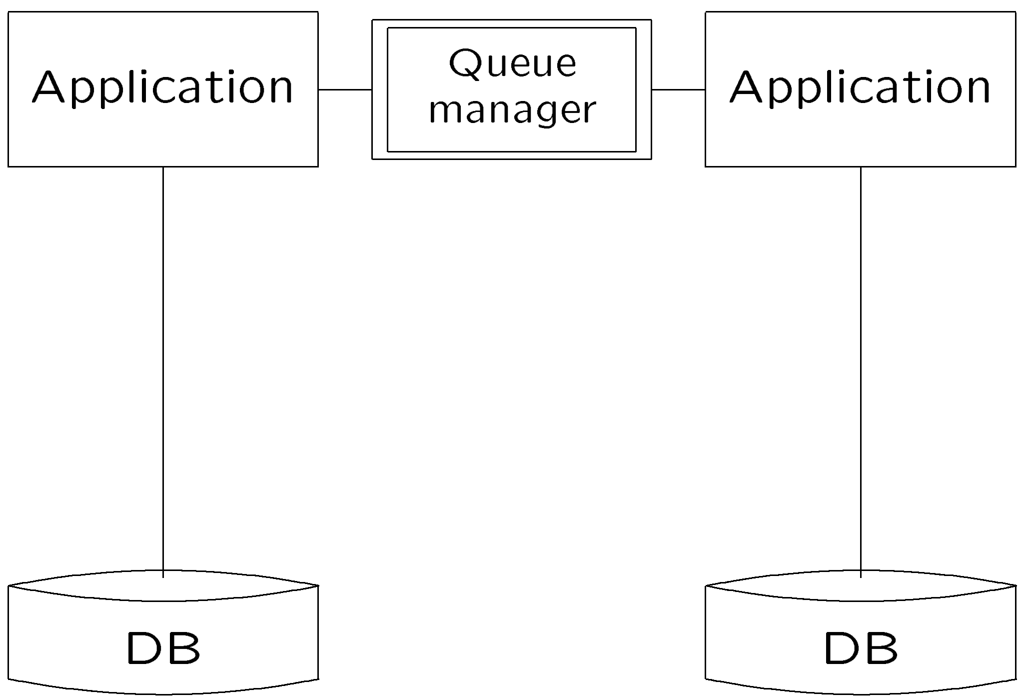
On-line message exchange



On-line message exchange

- function-oriented interface: remote procedure call (RPC)
- the client invokes the execution of a program on the server and gets the results
- very rough extreme: screen-scrapers
- stored procedures in DBMS or open systems APIs in distributed environments
- modern evolution: object-oriented services
- widely used in traditional TP monitors and in more modern distributed object technologies

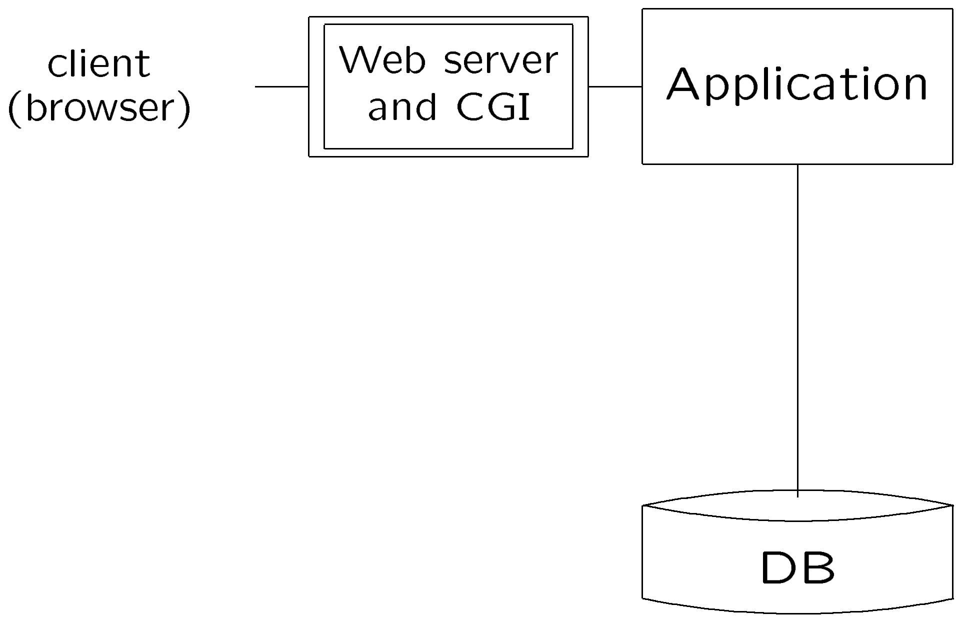
Off-line message exchange



Off-line message exchange

- function-oriented again (typically), but asynchronous
- a tool handles queues of messages (message-oriented middleware)
- tolerates unavailability of server connection

On-line data access



On-line data access
database access through WWW

- very common (and useful) in structured WWW servers
- Internet vs Intranet
- allows the wide dissemination of information

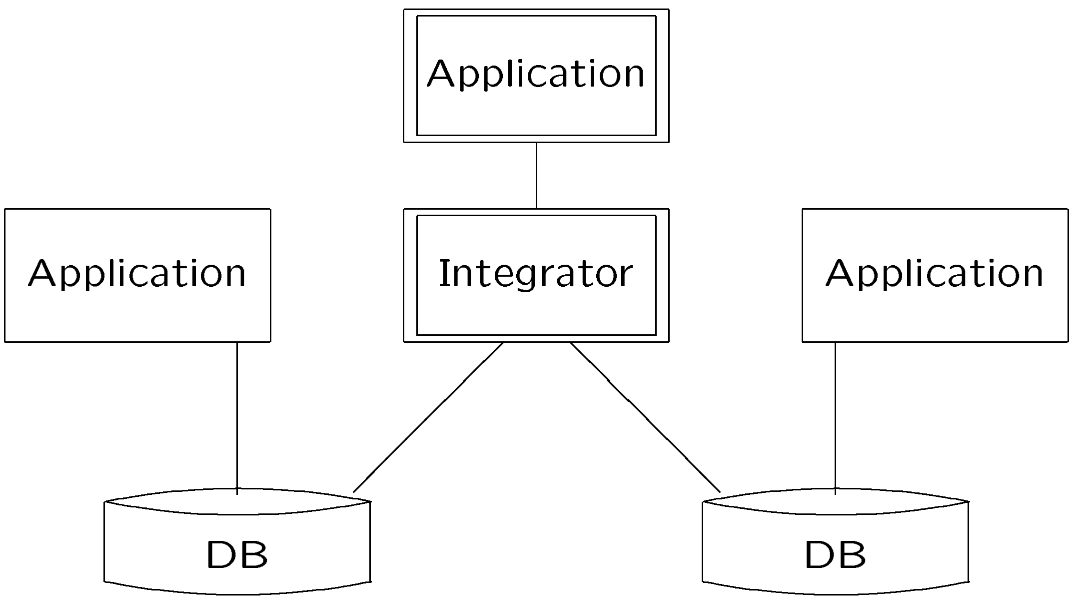
- introduction
- classification criteria
- classification
- architectural techniques
- $\boxed{\text{architectures and middleware tools}}$
- conclusion and discussion

Architectures for data cooperation

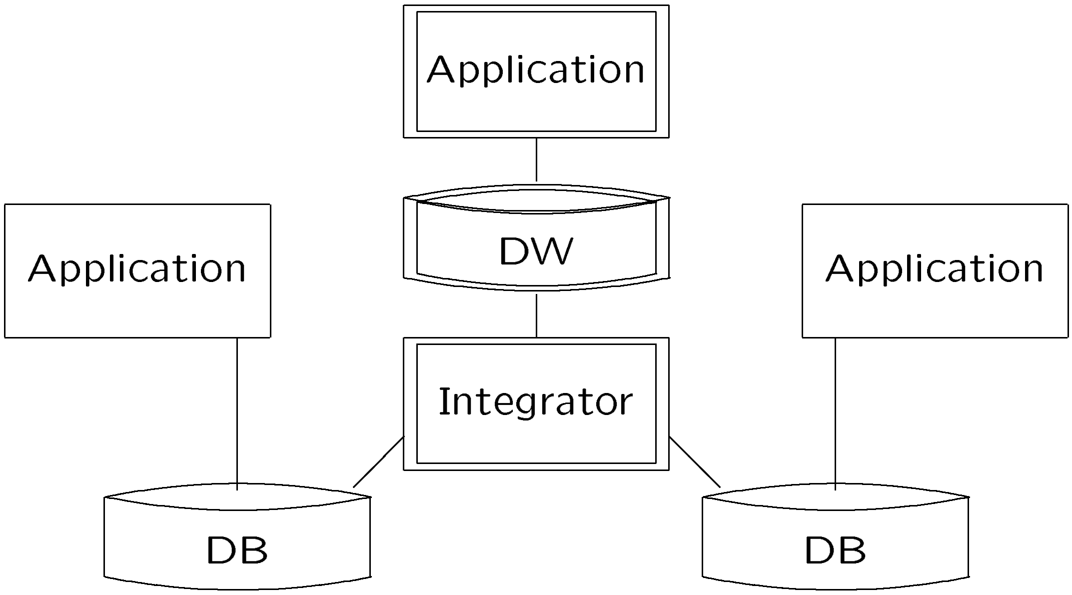
Based on

- elementary tools
- Database Gateway Servers or Distributed DBMSs with gateways
- Data Warehousing tools
- Distributed Transaction Monitors
- Object Request Brokers
- integrated tools

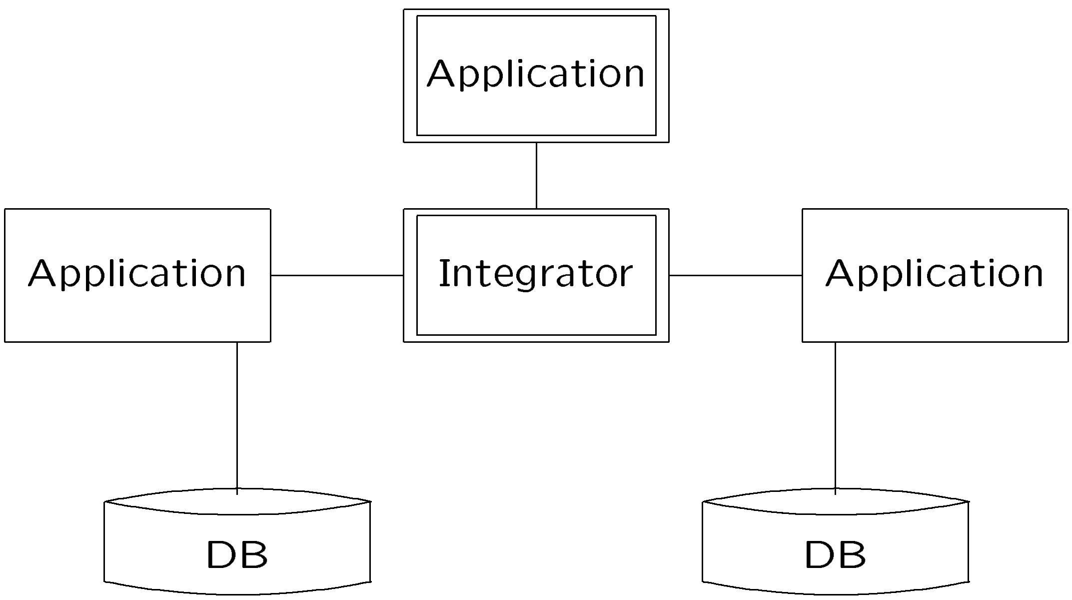
Architectures based on Database Gateway Server or Distributed DBMS with gateways



Architectures based on Data Warehousing tools



Architectures based on Distributed TP Monitors or Object Request Brokers



Transaction Processing Monitors

- traditional TP Monitors: efficient (queue mgmt) and reliable (correct transactions) access from remote terminals
- distributed TP Monitors: efficient and reliable access to remote services in a distributed environment
- currently, this is not a complete distributed computation environment

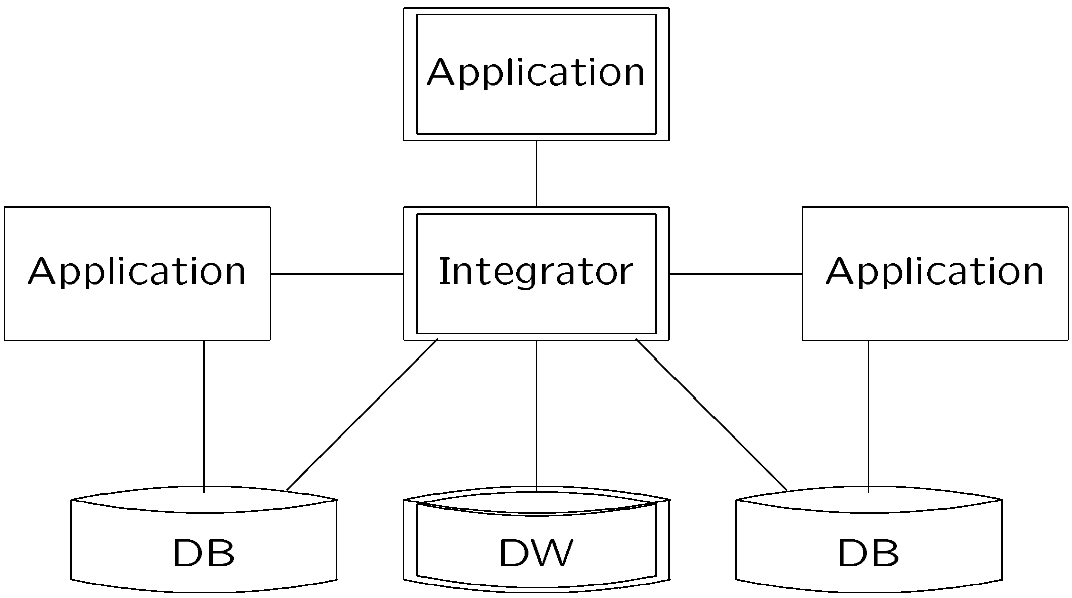
Object Request Brokers

- general-purpose distributed computing architecture
- object-oriented: object interfaces are used by clients that don't see implementations
- do not allow complete database transparency
- could be integrated with tools that support transaction management

Integrated Tools

- provide a suite of features
- often object-based
- often offer both development and execution support

Architectures based on Integrated Tools



- introduction
- classification criteria
- classification
- architectural techniques
- architectures and middleware tools
- conclusion and discussion

Conclusions

- cooperation can involve very different needs and so many alternatives exist
- careful evaluation of costs and benefits of the architecture: some soultions are complex and expensive
- cooperation does not require migration nor reengineering
- cooperation can stimulate and encourage migration and reengineering: it allows an incremental, low-risk approach to migration

General References

- P. Atzeni, L. Cabibbo, G. Mecca. Database cooperation: classification and middleware tools. Draft. Soon avalaible at `http://poincare.inf.uniroma3.it:8080`
- G. De Michelis et al. Cooperative information systems: A manifesto. To appear in *Cooperative Information Systems: Trends & Directions*, M. Papazoglu and G. Schlageter (eds), Academic Press, 1997.
- M.L. Brodie and M. Stonebraker. *Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach*. Morgan Kauffman, Los Altos, 1995.
- W. Kim, editor. *Modern Database Systems: the Object Model, Interoperability, and Beyond*. ACM Press and Addison Wesley, 1995.
- J.A. Larson. *Database Directions*. Prentice Hall, 1995.
- P. A. Bernstein. Middleware: A model for distributed system services. *CACM* 39, 2, Feb. 1996, 86–98.
- W.H. Inmon. *Building the Data Warehouse*, Second Edition. John Wiley, 1996
- OMG. OMA Executive Overview. `http://www.omg.org/omaov.htm`
- OMG. Suggested Readings. `http://www.omg.org/suggrdgs.htm`
- A.P. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.
- UCS Messaging Team *UCS TechKnowShare: A Primer on Middleware*, Indiana University, `http://msgwww.ucs.indiana.edu/messaging/infoshare/middleware.html`

References to tools and products

- M.L. Brodie and M. Stonebraker. *Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach*. Morgan Kauffman, Los Altos, 1995, Chapter 10.
- Web sites of all product vendors
- *DBMS 1996 Buyer's Guide and Client/Server Sourcebook. Middleware, Connectivity, and Internet Tools*. `http://www.dbmsmag.com/pcmidcon.html`
- UCS Messaging Team *UCS TechKnowShare: A Primer on Middleware*, Indiana University, `http://msgwww.ucs.indiana.edu/messaging/infoshare/middleware.html`