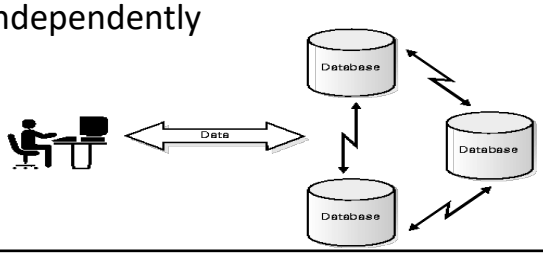
Distributed Databases
/
Database Sharding

Brendan Tierney

1

Distributed Databases

- A distributed database (DDB) is a collection of
  - multiple logically interrelated databases
  - distributed over a computer network,
  - and a distributed database management system (DDBMS) as a software system that manages a distributed database while making the distribution transparent to the user.
- Data is stored at several sites, each managed by a DBMS that can run independently



2

2

Properties

- **Distributed Data Independence :** Users should not have to know where data is located (extends physical and logical independence principles)
- **Distributed Transaction Atomicity :** Users should be able to write transactions accessing multiple sites just like local transactions.

3

3

Advantages

- **Management of distributed data with different levels of transparency**: A DBMS should be distribution transparent in the sense of hiding the details of where each Table or Relation is physically stored.
  - *Distribution or network transparency :* Where we have location transparency and Naming transparency.
    - Location transparency relates to the issuing of commands without the need to specify the locations necessary to complete the command.
    - Naming transparency relates to the naming of objects such that an object can be accessed unambiguously.
  - *Replication transparency :* copies of data may be stored at multiple sites for better availability, performance, and reliability. Replication transparency makes the user unaware of the existence of copies.
  - *Fragmentation transparency :*
    - Horizontal fragmentation distributes a relation into sets of tuples (rows).
    - Vertical fragmentation distributes a relation into subrelations where each subrelation is defined by a subset of the columns of the original relation. A global query by the user must be transformed into several fragment queries.

4

4

Advantages

- **Increased reliability and availability:**
  - In a centralized system, failure at a single site makes the whole system unavailable to all users. In a distributed database, some of the data may be unreachable, but users may still be able to access other parts of the database.
  - Reliability is the probability that a system is running at a certain time point,
  - whereas availability is the probability that the system is continuously available during a time interval.
  - In a distributed environment one site may fail while other sites continue to operate. The data and software that exist at the failed site cannot be accessed.
  - This improves both reliability and availability. Replication can improve availability and reliability making data available at more than one site.

5

5

Advantages

- **Improved performance:** A distributed DBMS fragments the database by keeping the data closer to where it is needed most.
  - Reduces contention for CPU and I/O services and simultaneously reduces access delays involved in wide area networks.
  - Local DBs are smaller and therefore local queries and transactions accessing data at a single site have better performance because of the smaller local databases.
  - Each site has a smaller number of transactions executing than if all transactions are submitted to a single centralized database.
- **Easier expansion:** Expansion of the system in terms of adding more data, increasing database sizes, more locations or adding more processors is much easier.

6

6

Additional Features

- DDBMs must have a number of additional features to those of a traditional DBMS to manage the additional complexity :
  - Keeping track of data : data distribution, fragmentation, etc.
  - Distributed query processing
  - Distributed transaction management
  - Replicated data management
  - Distributed database recovery
  - Security
  - Distributed metadata management

7

7

Distributed Database Architectures

- Distributed database systems describe different systems that differ from one another and the only thing that they have in common is the fact that data and software are distributed over multiple sites connected by some form of communication network.

- An important factor to consider is the degree of homogeneity.
  - Homogeneous - Is where all servers (or individual local DBMSs) use identical software and all users (clients) use identical software to access and manipulates the data.
  - Heterogeneous - Is where all the servers use different software and users can use different software to access and manipulate the data. One server may be a relational DBMS, another a network DBMS, or an object or hierarchical DBMS;
    - in such a case it is necessary to have a canonical system language and to include language translators to translate subqueries from the canonical language to the language of each server

8

8

Distributed Database Architectures

- Federated Database Systems
  - Also known as a multi-database system, each server is an independent and autonomous centralized DBMS that has its own local users, local transactions, and DBA and hence has a very high degree of *local autonomy.*
  - The term federated database system **(FDBS)** is used when there is some global view or schema of the federation of databases that is shared by the applications.
  - On the other hand, a multidatabase system does not have a global schema and interactively constructs one as needed by the application. Both systems are hybrids between distributed and centralized systems.

  – Design Issues
  - Differences in data models
  - Differences in constraints
  - Differences in query languages
  - Differences in semantics

9

9

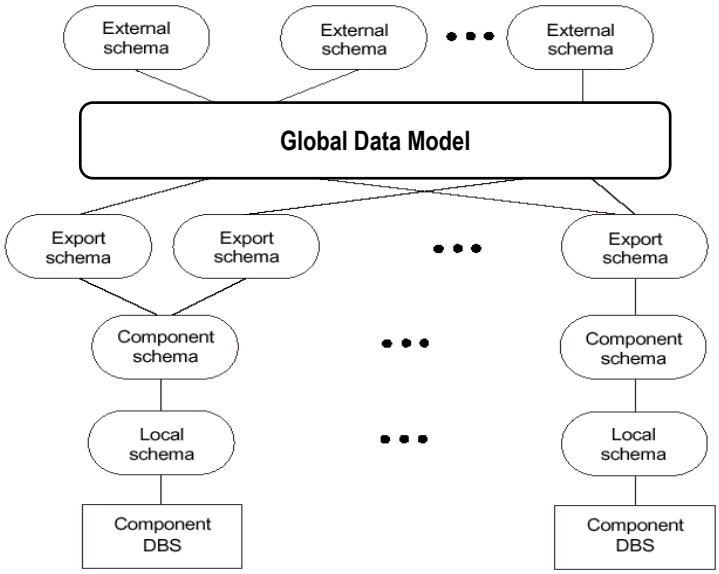
Distributed Database Architectures

- Semantic Heterogeneity
  - Semantic heterogeneity occurs when there are differences in the meaning, interpretation, and intended use of the same or related data. Semantic heterogeneity among component database systems creates the biggest hurdle in designing global schemas of heterogeneous databases.
  - Design considerations
    - The universe of discourse from which the data is drawn : 2 different customer databases with different attributes
    - Representation and meaning
    - Understanding, meaning and interpretation of data
    - Transaction and policy constraints

10

10

Distributed Database Architectures

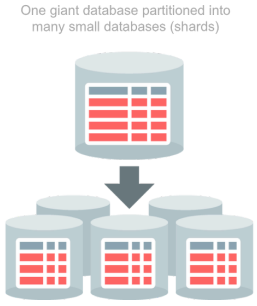


11

11

Data Management

- The main components of distributed database design are :
  - Data fragmentation
  - Fragment allocation
  - Data Replication
  - Transparency



- All information (meta-data) relating to this will be stored in the global directory (Distributed Metadata Management).

Data Fragmentation = Data/Database Sharding

12

12

Data Management – Data Fragmentation

- One of the main design challenges in distributed database design is where to store the different portions of the data.
  - A portion can be a relation, a certain group of relating tuples or a set of attributes of a relation.
- There are 3 types of data fragmentation :
  - Horizontal fragmentation
  - Vertical fragmentation
  - Hybrid fragmentation

Sharding

13

13

Data Management – Data Fragmentation

- Horizontal Fragmentation
  - A horizontal fragment of a relation is a subset of the tuples on that relation (portions/partitions).
  - The tuples that belong to the horizontal fragment are specified by a condition on one or more attributes of the relation (SQL Select statement with a WHERE clause).
  - Horizontal fragmentation divides a relation "horizontally" by grouping rows to create subsets of tuples, where each subset has a certain logical meaning. These fragments can then be assigned to different sites in the distributed system.
  - Example
    - An example would be an Employee relation. Each site in the distributed environment will have an Employee table but will only have the tuples relating to the Employees for their specific site. Where each site may represent a different department.

14

14

Data Management – Data Fragmentation

- Horizontal fragmentation will not be for one or a few relations but for the entire database. Only local data will be stored in the database. Each site will have the same or common schemas.
- This is called Derived horizontal fragmentation and applies the partitioning of a primary relation (DEPARTMENT in our example) to other secondary relations, which are related to the primary via a foreign key. This way, related data between the primary and the secondary relations gets fragmented in the same way.

15

15

Horizontal Fragmentation

branch-name	*account-number*	*balance*
Hillside	A-305	500
Hillside	A-226	336
Hillside	A-155	62

branch-name	*account-number*	*balance*
Valleyview	A-177	205
Valleyview	A-402	10000
Valleyview	A-408	1123
Valleyview	A-639	750

16

16

Data Management – Data Fragmentation

- Vertical Fragmentation
  - Vertical fragmentation divides a relation "vertically" by columns. A vertical fragment of a relation keeps only certain attributes of the relation.
  - Each site may not need all the attributes of a relation. Vertical fragmentation allows the distribution of a relations attributes across multiple sites.
  - Example
    - an employee relation would be to group the attributes into personal information and work/salary related information.
  - When using vertical fragmentation it is vital that the identifying attributes are shared with all fragments. This gives the **duplication of attributes** and hence a storage overhead.
    - But are vital in order to allow the merging of the different portions of information. Joins are performed using the common (primary keys) attributes.

17

17

Vertical Fragmentation

branch-name	*customer-name*	*tuple-id*
Hillside	Lowman	1
Hillside	Camp	2
Valleyview	Camp	3
Valleyview	Kahn	4
Hillside	Kahn	5
Valleyview	Kahn	6
Valleyview	Green	7

account number	*balance*	*tuple-id*
A-305	500	1
A-226	336	2
A-177	205	3
A-402	10000	4
A-155	62	5
A-408	1123	6
A-639	750	7

18

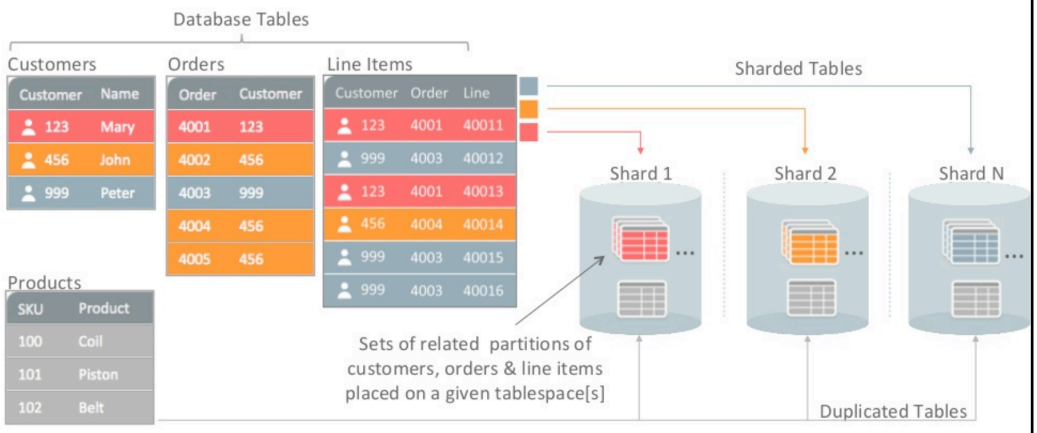
18

Data Management – Data Fragmentation

- Hybrid Fragmentation
  - Hybrid fragmentation is when a mixture of horizontal and vertical fragmentation is used to distribute the database among the sites.
  - Example
    - employee example we may have the data fragmented by department (horizontal) and by groupings of its attributes (vertical).
  - Hybrid fragmentation can be very complex to design and manage. It should only be used in situations that require it.
  - If used then an allocation schema tool is required to store information about the allocation of fragments to sites of the DDBS
    - it is a mapping that specifies for each fragment the site(s) at which it is stored. If a fragment is stored at more than one site, it is said to be replicated.

19

19



20

20

Data Management – Data Replication

- Data replication evolves the copying of data from one site to anther or the entire data in the distributed database to one site.
  - The copying of data from one site to another is call partial replication. The copying of the entire data in the database is known as full replication.
  - Full Replication - the availability of the database is guaranteed if at least one site remains available.
- The principle aim of data replications is to make data available to facilitate the optimal performance of the applications. This mainly revolves around query processing.

21

21

Data Management – Data Replication

- The disadvantages of full replication is
  - that it can slow down update operations drastically, since a single logical update must be performed on every copy of the database to keep the copies consistent.
  - Full replication also makes the concurrency control and recovery techniques more expensive than they would be if there were no replication.
- With partial replication of the data some fragments of the database may be replicated whereas others may not.
  - The number of copies of each fragment can range from one up to the total number of sites in the distributed system.
  - Examples include mobile applications.

22

22

Data Management – Data Replication

- Publish & Subscribe
- Push & Pull

23

23

Distributing the Data

- Headquartered in NY, a company's database consists of 6 large tables: A, B, C, D, E, F.
  - The company has major sites in Los Angeles, Memphis, New York, Paris, and Tokyo
- With a centralised database, all 6 tables would be located in NY.
- The first and **simplest** idea in distributing the data would be to disperse the six tables among the five sites, perhaps based on frequency of use of each table.

24

24

Distributing the Data

- Tables A and B are kept at New York
- Table C is moved to Memphis
- Tables D and E are moved to Tokyo
- Table F is moved to Paris.

- Paris employees can now access Table F without incurring telecommunications costs associated with accessing Table F in NY.

- Local autonomy - Paris employees, e.g., can take responsibility for Table F -- its security, backup and recovery, and concurrency control.

25

25

Distributing the Data: Problems

- When the database was centralised at New York, a query issued at any of the sites that required a join of two or more of the tables could be handled in the standard way by the computer at New York.
- The result would then be sent to the site that issued the query.
- In the dispersed approach, a join might require tables located at different sites!

26

26

Replicated Tables

- Duplicate tables at two or more sites on the network.
- Advantages
  - Availability - during a site failure, data can still be accessed at a replicated location.
  - Local access - Replicate table at a site requiring frequent access.
- Disadvantages
  - Security risk
  - Concurrency control - How do you keep data consistent when it is replicated in tables on three continents ?

27

27

Full Data Replication

- The **maximum** approach of replicating every table at every site.
- Great for availability
- Great for joins

28

28

Full Data Replication

- **Worst** for concurrency control - every change to every table has to be reflected at every site.
- **Worst** for security
- Takes up **a lot** of disk space
- **Increase** server specification at each site
- **Most** expensive

29

29

Partial Replication

- Have a copy of the entire database at headquarters in New York and have each table replicated exactly once at one of the other sites.

30

30

Partial Replication

- Improves availability -each table is now at two sites.
- Security and concurrency exposures are limited.
- Joins occur at NY.

31

31

Partial Replication

New York could tend to become a bottleneck.

If a table is heavily used in both Tokyo and Los Angeles, it can only be placed at one of the two sites (plus the copy of the entire database in New York), leaving the other with speed and telecom cost problems.

32

32

Replication Principles

- Place copies of tables at the sites that use them most heavily in order to minimise telecommunications costs.
- Ensure that there are at least two copies of important or frequently used tables to realise the gains in availability.
- Limit the number of copies of any one table to control the security and concurrency issues.
- Avoid any one site becoming a bottleneck.

33

33

Data Management – Transparency

- Distribution Transparency
  - Fragmentation Transparency
  - Location Transparency
  - Replication Transparency
  - Local Mapping Transparency
  - Naming Transparency
- Transaction Transparency
  - Concurrency Transparency
  - Failure Transparency
- Performance Transparency
  - DBMS Transparency
- DBMS Transparency

34

34

Distribution Transparency

- Distribution transparency allows user to perceive database as single, logical entity.
- If DDBMS exhibits distribution transparency, user does not need to know:
  - data is fragmented (fragmentation transparency),
  - location of data items (location transparency),
  - otherwise call this local mapping transparency.
- With replication transparency, user is unaware of replication of fragments .

35

35

Naming Transparency

- Each item in a DDB must have a unique name.
- DDBMS must ensure that no two sites create a database object with the same name.
- One solution is to create central name server. However, this results in:
  - loss of some local autonomy;
  - central site may become a bottleneck;
  - low availability; if the central site fails, remaining sites cannot create any new objects.
- Alternative solution - prefix object with identifier of site that created it.
- For example, Branch created at site S_1 might be named S1.BRANCH.
- Also need to identify each fragment and its copies.
- Thus, copy 2 of fragment 3 of Branch created at site S_1 might be referred to as S1.BRANCH.F3.C2.
- However, this results in loss of distribution transparency

36

36

Naming Transparency

- An approach that resolves these problems uses aliases for each database object.
- Thus, S1.BRANCH.F3.C2 might be known as LocalBranch by user at site S_1.
- DDBMS has task of mapping an alias to appropriate database object.

- Database views can be used to hid some of the naming transparency issues

37

37

Transaction Transparency

- Ensures that all distributed transactions maintain distributed database's integrity and consistency.
- Distributed transaction accesses data stored at more than one location.
- Each transaction is divided into number of subtransactions, one for each site that has to be accessed.
- DDBMS must ensure the indivisibility of both the global transaction and each of the subtransactions.

38

38

Concurrency Transparency

- All transactions must execute independently and be logically consistent with results obtained if transactions executed one at a time, in some arbitrary serial order.
- Same fundamental principles as for centralized DBMS.
- DDBMS must ensure both global and local transactions do not interfere with each other.
- Similarly, DDBMS must ensure consistency of all subtransactions of global transaction.

39

39

Concurrency Transparency

- Replication makes concurrency more complex.
- If a copy of a replicated data item is updated, update must be propagated to all copies.
- Could propagate changes as part of original transaction, making it an atomic operation.
- However, if one site holding copy is not reachable, then transaction is delayed until site is reachable.
- Could limit update propagation to only those sites currently available. Remaining sites updated when they become available again.
- Could allow updates to copies to happen asynchronously, sometime after the original update. Delay in regaining consistency may range from a few seconds to several hours.

40

40

Failure Transparency

- DDBMS must ensure atomicity and durability of global transaction.
- Means ensuring that subtransactions of global transaction either all commit or all abort.
- Thus, DDBMS must synchronize global transaction to ensure that all subtransactions have completed successfully before recording a final COMMIT for global transaction.
- Must do this in presence of site and network failures.

41

41

Performance Transparency

- DDBMS must perform as if it were a centralized DBMS.
  - DDBMS should not suffer any performance degradation due to distributed architecture.
  - DDBMS should determine most cost-effective strategy to execute a request.
- Distributed Query Processor (DQP) maps data request into ordered sequence of operations on local databases.
- Must consider fragmentation, replication, and allocation schemas.
- DQP has to decide:
  - which fragment to access;
  - which copy of a fragment to use;
  - which location to use.

42

42

Performance Transparency

- DQP produces execution strategy optimized with respect to some cost function.
- Typically, costs associated with a distributed request include:
  - I/O cost;
  - CPU cost;
  - communication cost.

43

43

Query Processing

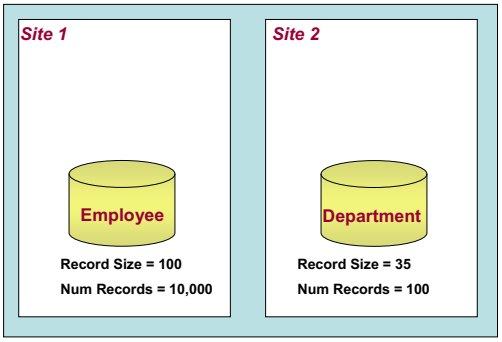
- Main consideration when issuing queries is the cost of communication for distributing the query and the results
- Cost of transferring data over the network
  - includes intermediate files being transferred to other sites for processing
  - the results of the query.
- The aim of DDBMS query optimization algorithms is to consider the goal of reducing the *amount of data transfer* as an optimization criterion.

44

44

Query Processing

- Example



Query

For each employee, retrieve the employee name and the name of the department for which the employee works

- Write the SQL statement for this query.
- How many records will the query produce ?

45

45

Query Processing

- Assuming that the query was issued another site (Site 3) and the query result is 40 bytes long, what are the different strategies for executing this query.

  1. Transfer both the EMPLOYEE and the DEPARTMENT tables to the result site (Site 3), and perform the join at site 3.
  2. Transfer the EMPLOYEE table to site 2, execute the join at site 2, and send the result to site 3.
  3. Transfer the DEPARTMENT table to site 1, execute the join at site 1, and send the result to site 3.

- The aim must be to select the solution that requires the minimum amount of work.
  - Which one would you select and why

46

46

47

47

Query Processing

- Using the previous example

For each department, retrieve the department name and the name of the department manager.

The query result will only have 100 records as there are only 100 records in the department table (the query result size is 40 bytes per record again).

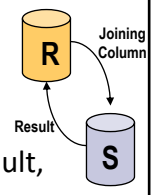
- What are the different strategies that can be considered in this case ?
- Suppose the query was issued at site 2. What are the different strategies.

48

48

Semi-Join Queries

- Query processing using the *semi-join operation* is to reduce the number of tuples in a relation before transferring it to another site.
- Send the *joining column* of one relation R to the site where the other relation S is located; this column is then joined with S.
- The join attributes, along with the attributes required in the result, are sent back to the original site and joined with R.
- Only the joining column of R is transferred in one direction, and a subset of S with no extraneous tuples or attributes is transferred in the other direction.
- If only a small fraction of the tuples in S participate in the join, this can be quite an efficient solution to minimizing data transfer.



49

49

Concurrency Control

- Concurrency control specific issues in distributed databases are :
  - Handling multiple copies of data items : must be responsible maintaining consistency among these copies.
  - Failures at individual sites : should continue as normal with available sites.
  - Failure of communication links : need to avoid network partitioning i.e. 2 separate parts.
  - Distributed commit : What if one site fails. Need to use 2 phase commit
  - Distributed deadlock : added complexity having multiple sites
- 2 main approaches to concurrency control in a DDBMS.
  - Distinguished Copy
  - Voting

50

50

Concurrency Control

- Distinguished Copy
  - Is where a particular copy of a data item (table) is used for the locking mechanism.
  - All requests that are handled by the locking mechanism will look for the site that maintains the distinguished copy of the data item in question.
  - There are a number of different methods.
    - In the primary site technique, all distinguished copies are kept at the same site. A modification of this approach is the primary site with a backup site.
    - The primary copy method is where the distinguished copies of the various data items can be stored in different sites.
  - A site that includes a distinguished copy of a data item basically acts as the coordinator site for concurrency control on that item.

51

51

Concurrency Control

- Voting
  - Is where a lock request is sent to all sites that have a copy of the data item.
  - Each copy maintains its own lock and can grant or deny the request for it.
    - If a transaction that requests a lock is granted that lock by a majority of the copies, it holds the lock and informs all copies that it has been granted the lock.
    - If a transaction does not receive a majority of votes granting it a lock within a certain time-out period, it cancels its request and informs all sites of the cancellation.
  - The disadvantages to this method is the volume of messaging traffic among the sites, compared to the distinguished method
    - added complexity to handle the possibility of site failures during the voting process.

52

52

Transaction Management

- A distributed transaction which involves data at more than one site should behave in the same way as a non-distributed transaction
  - It must be capable of enforcing the ACID conditions for transactions without user intervention.
- A DDBMS will have one or more transaction managers that ensure that transactions are executed correctly
  - As does the transaction manager in a centralised system.
- The transaction manager in a DDBMS might not have access to all the required management data, nor have the control permissions necessary to perform the appropriate management actions at all the sites.
  - In systems where sites have high autonomy, implementing global transaction management strategies is extremely complex and may not always be possible.

Transaction Management

- Success / Failure of Transaction
  - A distributed transaction must exhibit the same behaviour (success/failure) as a centralised transaction.
  - If a sub-transaction fails the transaction management system must ensure that, globally, the entire distributed transaction fails and that all previously executed sub-transactions are rolled back.
- Two-Phase Commit
  - Is a co-ordination protocol that enforces the ACID conditions for distributed transactions and co-ordinates the commit or rollback operations of a collect of sub-transactions of a distributed transaction.
  - Requires one site to co-ordinate the other sites involved in the transaction, this is usually the site where the transaction originated.
  - Each site involved in the transaction is called a participant and will process a sub-transaction. Each participant will communicate with the co-ordinating site.

Transaction Management

- Two-Phase Commit
  - 2 phases to the protocol
    - **Voting phase** : during which each participating site makes its intention to commit or rollback known to the co-ordinator.
    - **Decision phase** : during which the global decision to commit or rollback is communicated to the participating sites.
- The co-ordinator expects each participating site to report its voting decision after it has executed its sub-transaction.
  - Any vote not received is interpreted as a failure at the participating site. In this case a global rollback decision will be issued
  - If there a participating site votes to rollback then a global rollback decision is issued.
  - If the co-ordinator receives votes to commit by all the participants then a global commit decision is issued.
  - After voting the participants must wait until they receive the decision instruction (commit or rollback) from the co-ordinator.

55

55

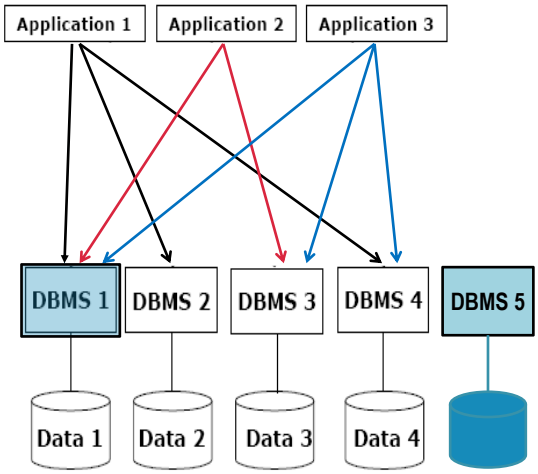
Distributed Database Recovery

- Very Complex
  - difficult to determine whether a site is down without exchanging numerous messages with other sites.
  - How up to-date is the site
  - What needs to be replicated.
  - For example, suppose that site X sends a message to site Y and expects a response from Y but does not receive it. There are several possible explanations:
    - The message was not delivered to Y because of communication failure.
    - Site Y is down and could not respond.
    - Site Y is running and sent a response, but the response was not delivered.
    - Without additional information or the sending of additional messages, it is difficult to determine what actually happened.
      - » Communication overhead

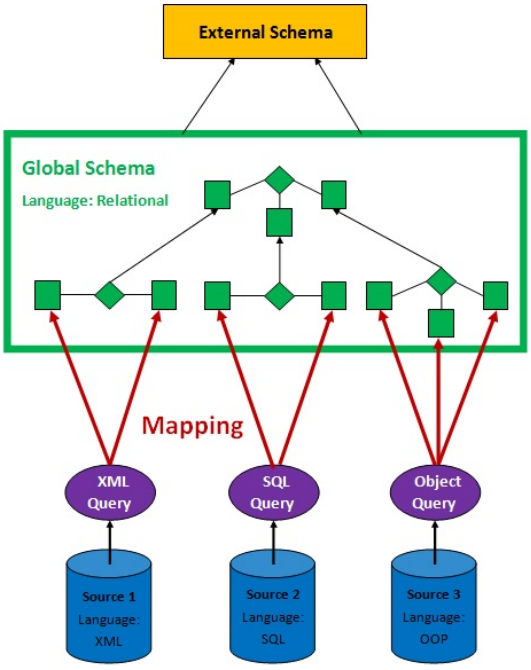
56

56

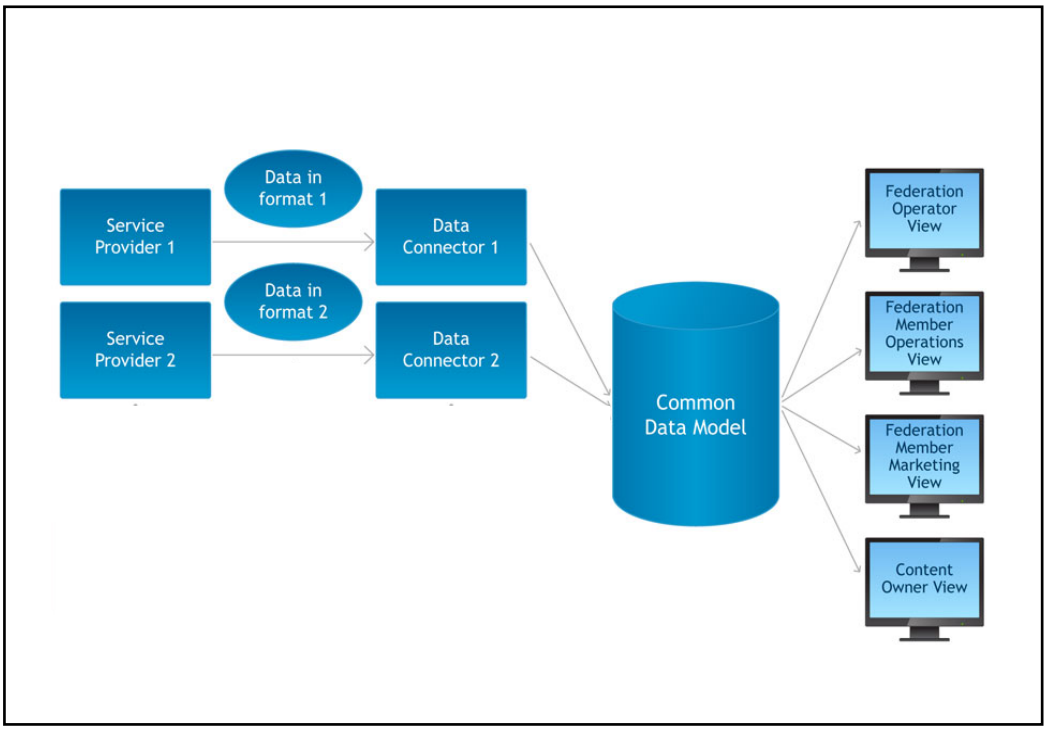
Integrating Databases



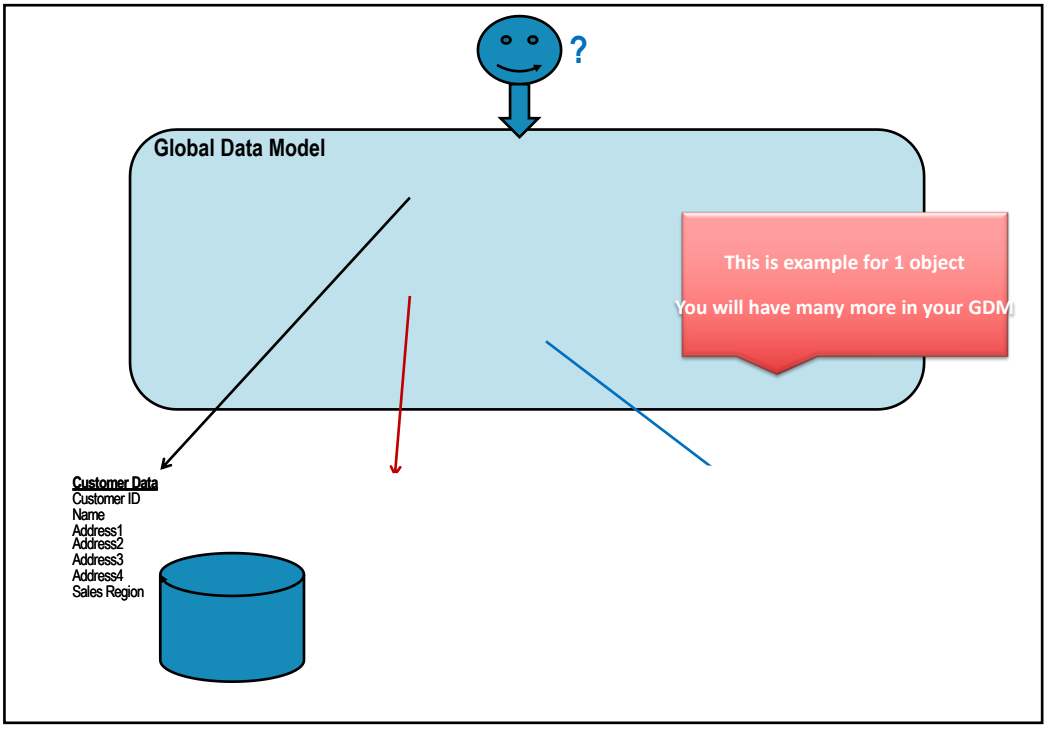
57



58

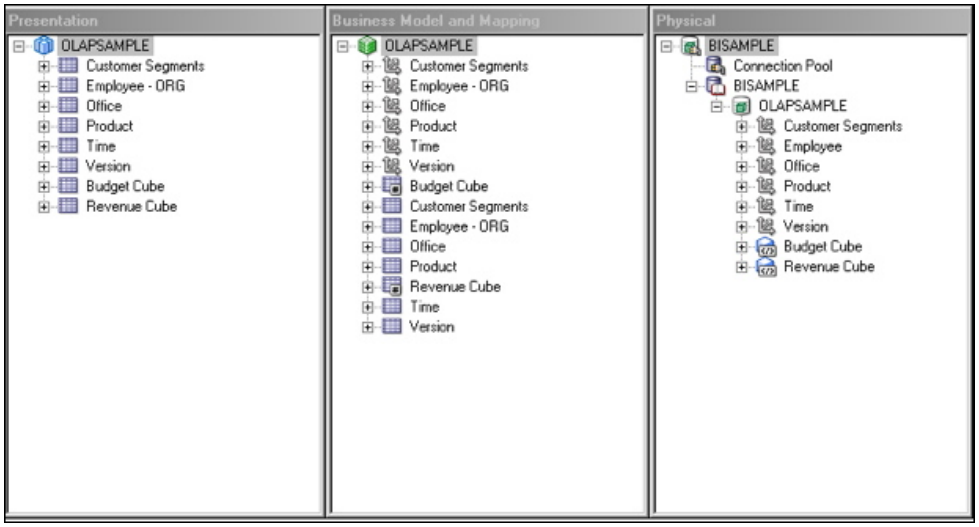

Service Provider 1
Service Provider 2
Data in format 1
Data in format 2
Data Connector 1
Data Connector 2
Common Data Model
Federation Operator View
Federation Member Operations View
Federation Member Marketing View
Content Owner View

59


?
Global Data Model
This is example for 1 object
You will have many more in your GDM
Customer Data
Customer ID
Name
Address1
Address2
Address3
Address4
Sales Region

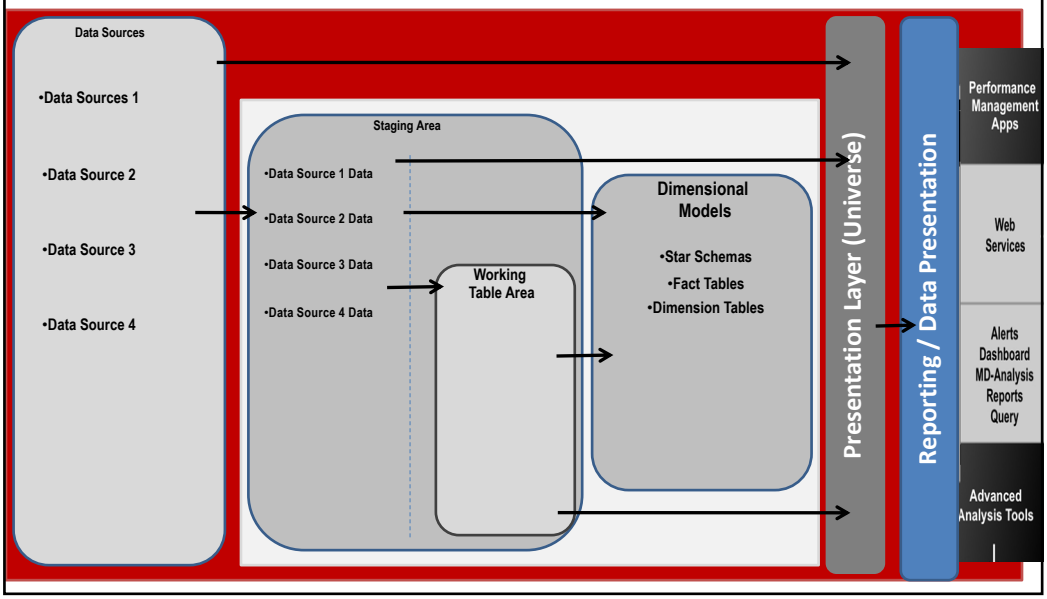
60

OBIEE + Others

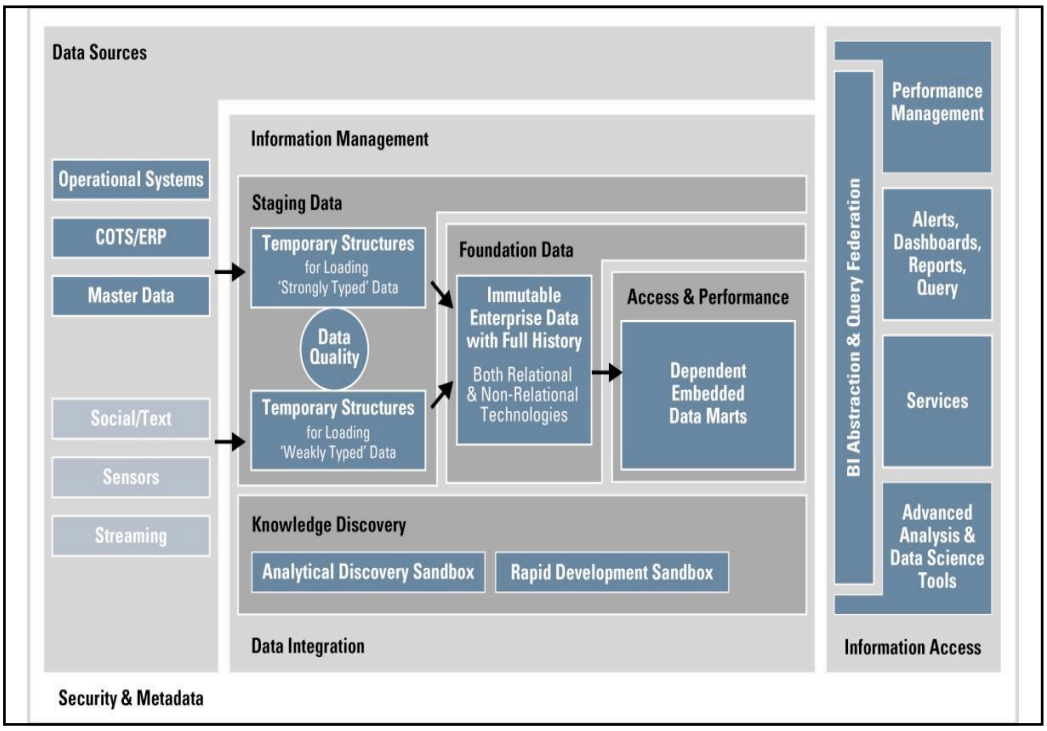


61

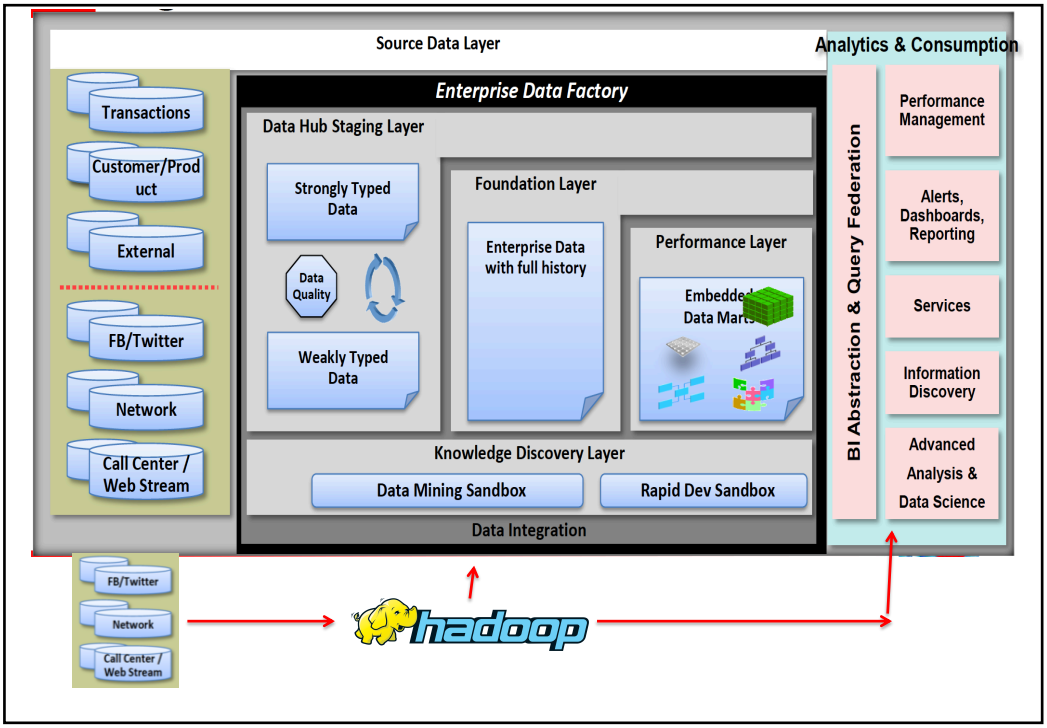
Oracle BI-DW Reference Architecture



62


Data Sources
Operational Systems
COTS/ERP
Master Data
Social/Text
Sensors
Streaming
Information Management
Staging Data
Temporary Structures for Loading 'Strongly Typed' Data
Data Quality
Temporary Structures for Loading 'Weakly Typed' Data
Foundation Data
Immutable Enterprise Data with Full History
Both Relational & Non-Relational Technologies
Access & Performance
Dependent Embedded Data Marts
Knowledge Discovery
Analytical Discovery Sandbox
Rapid Development Sandbox
Data Integration
Security & Metadata
BI Abstraction & Query Federation
Performance Management
Alerts, Dashboards, Reports, Query
Services
Advanced Analysis & Data Science Tools
Information Access

63


Source Data Layer
Analytics & Consumption
Transactions
Customer/Product
External
FB/Twitter
Network
Call Center / Web Stream
Enterprise Data Factory
Data Hub Staging Layer
Strongly Typed Data
Data Quality
Weakly Typed Data
Foundation Layer
Enterprise Data with full history
Performance Layer
Embedded Data Marts
Knowledge Discovery Layer
Data Mining Sandbox
Rapid Dev Sandbox
Data Integration
BI Abstraction & Query Federation
Performance Management
Alerts, Dashboards, Reporting
Services
Information Discovery
Advanced Analysis & Data Science
FB/Twitter
Network
Call Center / Web Stream
hadoop

64