Chapter 3 – Requirements and Analysis

3.1 Overall Description

Product perspective

Developed tool throughout this project is a follow-on member of Database integration product family. It can be said that this tool is an extension to database client software, which has most features of database client and more rich functions to dealing with not only one database but several databases.

Heterogeneous Database Integration System is an autonomous product. And it is database management system independent.

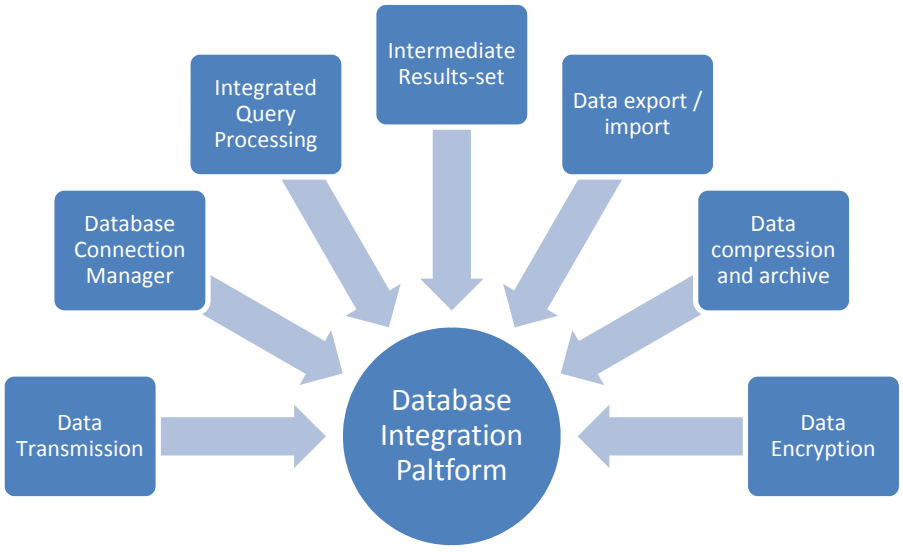


Figure 3.1 Major components of the overall system

3.2 User classes and characteristics

Database Administrators and Application Developers would be the as main users of the product.

In Table 3.1 we have differentiated user classes based on the frequency of use, subset of product functions used, technical expertise, and experience.

	Database Administrator	Application Developer
Frequency of use	Expected to use the tool frequently. For management reports, Data warehousing , Archiving user defined data sets	Not frequent as Database Administrator, but expects to integrate with applications that shows views on overall organizational perspective. Application developers are expected to develop plug-ins for connect to required databases
Subset of product functions	1. Add remove database connections 2. Create federated views 3. Load data from federated views 4. Save and query derived data 5. Export data from connected databases or from saved data 6. Restore archived data 7. Transmit data between connected databases	1. Create federated views 2. Retrieves Result set from defined federated views
Technical expertise	Network infrastructure, Database connection and configuration, Authentication mechanisms	JAVA, OO concepts, Java Database connectivity, Database concepts and architecture.
Experience	Through experience in business domain and applications, Database schemas and stored data	Through experience in business domain and applications, Database schemas and stored data

Table 3.1 - Users and characteristics

Operation Environment

Environment parameter	Value
Hardware Platform	Any
Operating System	Any with compatible JVM
Memory	512 Mb
Disk space	100 Mb (application only)
Connected databases	Java Database Connectivity 4.0 support
Processing power	Greater than 1 GHz processing power

Table 3.2 - Operational Environment

System is expected to use platform independently and will be able to configure (Memory consumption, Disk space, etc.) according to the available resources.

Figure 3.2 - Use-case diagram – Database Administrator's Perspective

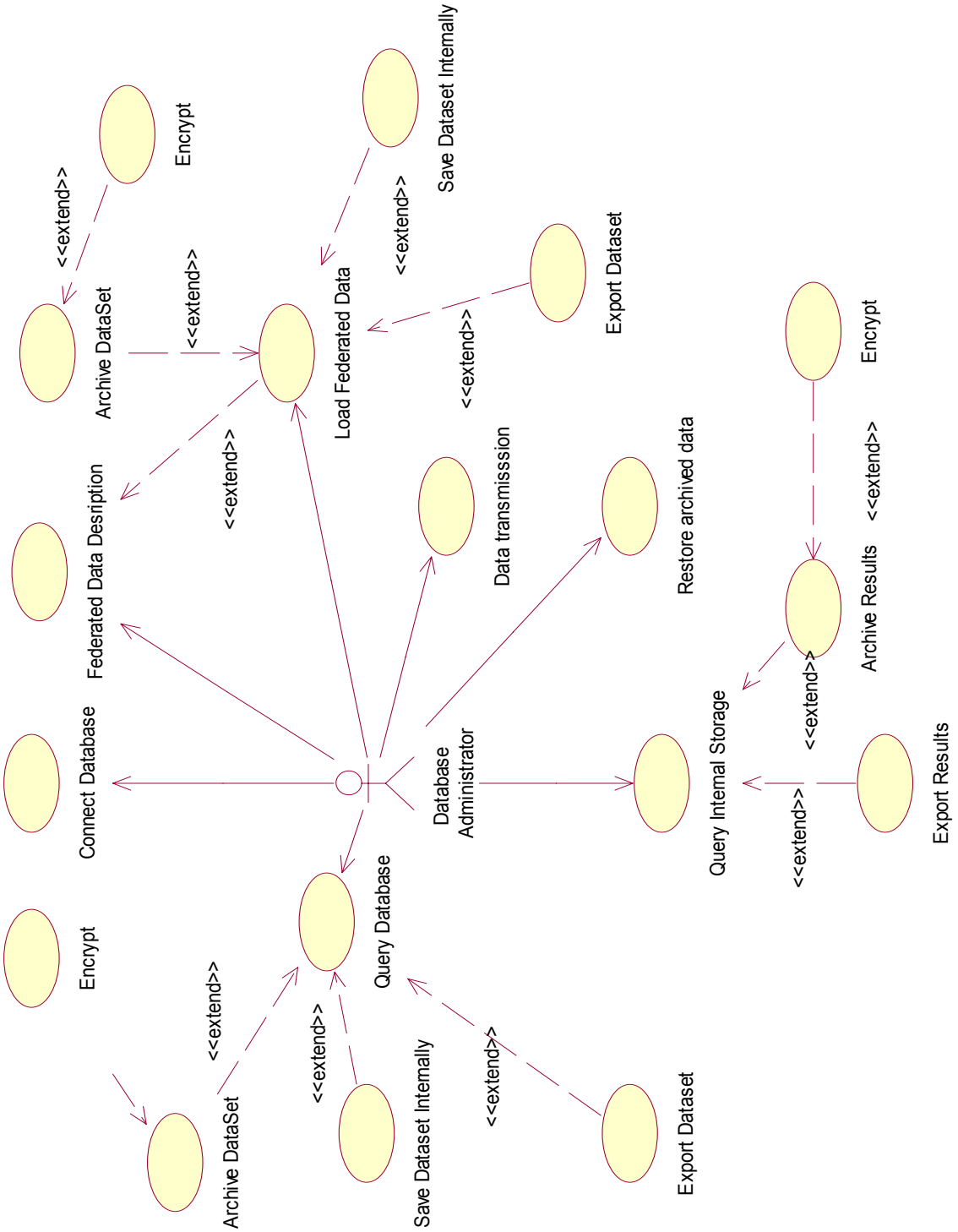
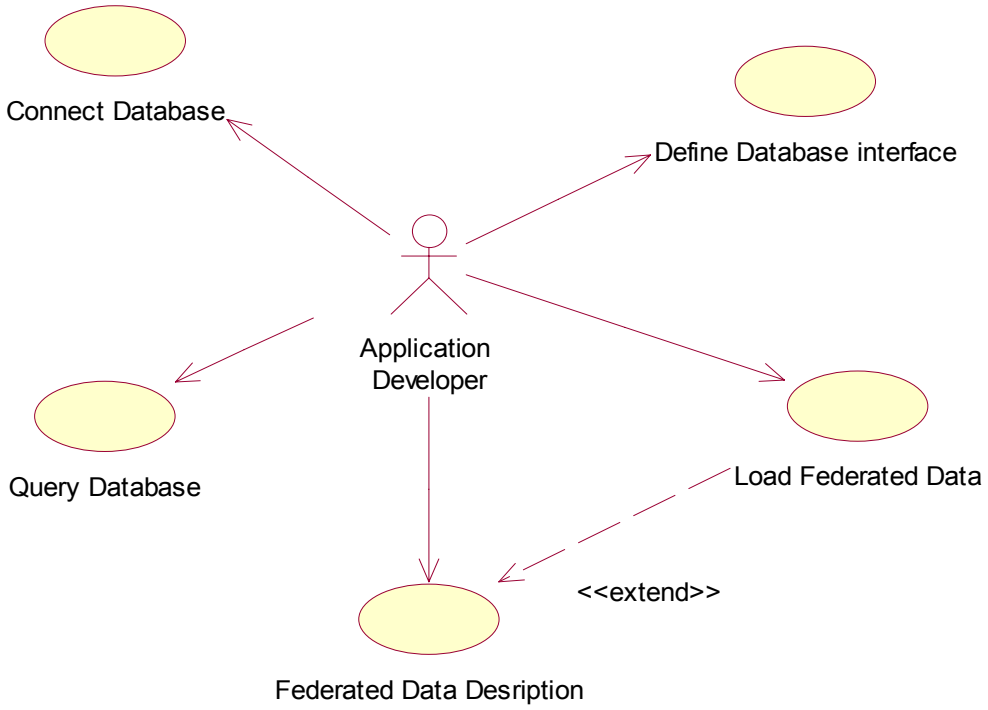


Figure 3.3 - **Use-case diagram – Application Developer's Perspective**



3.3 System Features

3.3.1 Connect Databases

a. Description and priority

Priority - high
System should be able to handle several databases. Therefore a mechanism to communicate with multiple databases is vital.

Stimulus / Response Sequence

User selects create new connection option and select the Database type. A unique connection reference name, database host, instance name, user name and password are provided. User tests the connection before adding to connection list. Only if the test is successful, new connection will be added to the database list.

Tree view of the databases list is then updated. Reference name of the new node will added as the new root node, tables and table columns of that instance are added as leaf nodes.

b. Functional Requirements

This is the first task user must do before using a database instance with the tool. Once a connection established, database objects/tables has to be listed down for the user reference. Unique reference name is important, because all objects of the connected-databases are referenced through this name. In federated views, selected objects are identified by (reference name): (database table): (table column). By referring to a connection name system should be able to get all related information to that database instance, for an example tables, table columns, column sizes etc.

Database connections have to be established real time. Before adding to the connection list system should check the connection availability and it should be add to the list only if connection is possible to set up.

3.3.2 Create Federated data description

a. Description and priority

Priority – High

User creates a description of a federated data set, using table columns of the connected databases.

b. Stimulus / Response Sequence

Users select the columns which are required in federated view from connected database tables. Clicking on the column will add itself to the selection list. And specify the join condition of tables simulating as a normal SQL join. Then save the description as a file.

c. Functional Requirements

Federated data description should be created from the table columns of connected databases. Terminology of creating a federated data set is expected to maintain simplicity like creating an usual SQL query. Selected columns have to be distinguish by combining (reference name): (database table): (table column). In order to perform join operation we need to specify name of the table and table columns join on and a condition to meet for the join to happen.

This function should allow a user to save the federated data description as a file. That file must be self descriptive and should be able to load when a user wants to retrieve data. Particular file should contain all the data required for database integration. It should contain database references, SQL for retrieve data from each database; join attributes of the result sets. Users load federated data description and retrieve real time data from connected database.

d. Meta data for Federated Dataset

- Database reference name
- Database tables

- Selected columns from each table
- Join column of results from each site

3.3.3 Load Federated Data

a. Description and priority

Priority – High

In this function databases are queried according to a federated data description which is already saved through the above function.

b. Stimulus / Response Sequence

User will open stored description of a federated dataset. Then system will conduct verification and validation operation on description file and information will display on the interface. If the description is correct user can continue with data loading operation.

Using valid federated query description file, system will perform database integration and results will be displayed as a normal SQL query output.

c. Functional Requirements

Objective of this function is to reuse already defined federated query descriptions. And save the result internally and extract the required view for further operations.

Federated data description needs to be maintained separately and when required, it should be able to use for data loading. When data is loading to the system, it should load real time data from federated databases.

Once data is loaded user will be able to save data internally export and compress data or transmit to another connected database. When saving internally or transmitting to a database, user should be able to map destination table columns with the exporting data columns. If required table is not available, system should

prompt the user to create a new table. Exported data has to be a combination of data and its structure.

d. Validation criteria

A result generated from database integration should give same data set such that all the tables were resides in a one database management system.

3.3.4 Archive data

a. Description and priority

Priority – medium

Retrieved data should be able to be saved in database independent file format and later restore in a connected database. Saved file must be compressed and secured.

b. Stimulus / Response Sequence

User loads federated data set to the system either using federated query or a directly from connected database using SQL, and then select the archiving option. After that system will export self descriptive and compressed data set in to a file. Compressed file can be encrypted if user requires additional security.

c. **Functional Requirements**

One of the main objectives of this project is proposing an alternative data archiving solution. Hence data should be saved in a format that it is independent from where and how it's retrieved. For example, results of federated query and results of direct query from a connected database should be saved in a same structural format.

System exported data must be less storage consume than the data exported through the normal database export system. And satisfactory level of security should be provided. XML is the preferred method for saving data.

d. Sample Metadata for data saving

A	B
a	b1
a	b2
...	

```
<data>
  <A>a
    <B>b1</B>
    <B>b2</B>
  </A>
   ...
</data>
```

e. Sample metadata for saving data structure

```
<table>
 <column>
        <default></default>
        <size></size>
        <type></type>
< /column >
<table>
```

3.3.5 Save data internally

a. Description and priority

Priority – high

Retrieved data from databases should be saved internally in an observable manner for further manipulation.

b. Stimulus / Response Sequence

User retrieves data and select internally saving option. This will lead to save the data internal to the application. Saved data would be able to identify by the federated query name or a given name by the user. If required data structure is not available or already exist with data user will be warned.

c. Functional Requirements

Users may not be able to query the exact information from a federated query. Therefore Tool should provide the facility to manipulate federated data set further to get the exact view the user required. Therefore intermediate manipulation of the data is important. Results of the manipulated data should also be able to transmit to a connected database or archive.

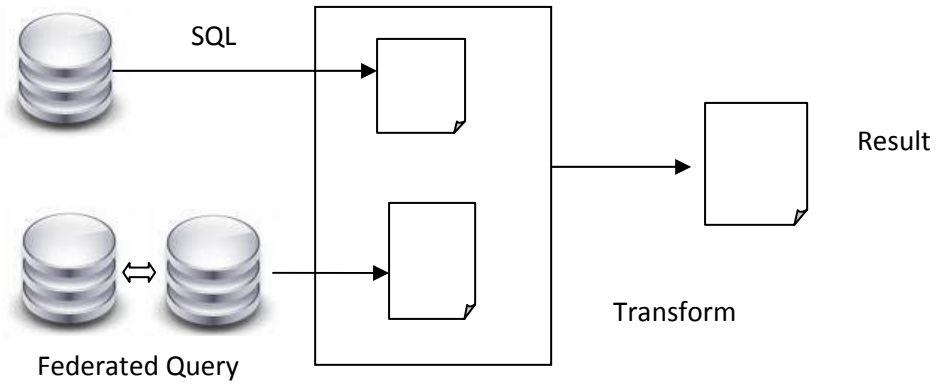


Figure 3.4 – Extract Transform Load

3.3.6 Export Federated data

a. Description and priority

Priority – medium

Data loaded from federated query should be able to transmit to a different connected database.

b. Stimulus / Response Sequence

User executes the federated query and loads the data. Then select the export option. In the export option user has to select the destination database and destination table. If Destination table is not available user has an option to create the table according to the exported dataset. User maps the retrieved data columns and destination columns next, and starts the export process.

c. **Functional Requirements**

Federated data views might be more meaningful in connected database, for an example, in a data mining process. Organizations can define federated views from different data repositories and save them periodically in to a data mining database or application. This will be a more effective process than taking all collected data saved in a data mining purpose.

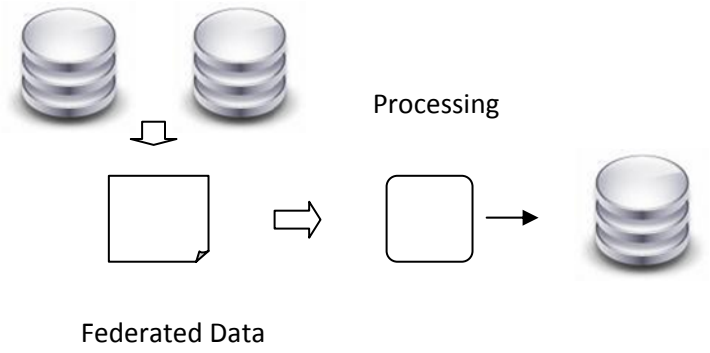


Figure 3.5 – Integrated Query Processing

3.3.7 Query internal Storage

a. Description and priority

Priority – high

Retrieve and manipulate internally stored data.

b. Stimulus / Response Sequence

User open the internal storage browsing tool and execute operations on stored data. Querying, Creating new objects, Drop existing objects, etc available as user functions.

c. **Functional Requirements**

Sometimes it might be easier to join several federated datasets other than creating one view satisfying all requirements. As a result system should be able

to manipulate fundamental operations on internally stored data, like grouping, summation etc.

It is required that transformed results be able to archive or transmit to a connected database.

3.3.8 Query Connected Database

a. Description and priority

Priority – high

The developing tool must have a common interface for query connected databases in order to compare and contrast datasets.

b. Stimulus / Response Sequence

User opens the query window and enters required SQL statement. Then specify the target database and execute the statement. If the SQL statement is a DML[1] statement results will be displayed on the results pane. If the SQL statement is a DDL[2] statement execution status will be displayed on the results pane.

c. Functional Requirements

Using the developing tool user can connect to several databases. And it should have functionality to retrieve data from connected databases following same procedure which is independent from the type of the database.

User should be able to query connected database and manipulate them using a single access point. And results must be displayed according to the SQL statement. Similar to the federated query processing derived results should be able to save internally, archive or transmit to another database.

[1] Data Manipulation Language
[2] Data Definition Language

3.3.9 Import / Export Utility

a. Description and priority

Priority – high

This function facilitates to import data from connected database and export saved data to a destination database.

b. Stimulus / Response Sequence

User selects a record set from a connected database using a SQL statement. Archiving option will save the results and result set structure into files. Saved file will be compressed. If required, the compressed file can be encrypted using a given key or system generated encryption key.

For data importation function, the source will be the compressed files (data structure and data) exported in the above function. When importing data user has to select destination from connected database list. After selecting destination user has to map the source and destination columns. If required table is not available system will provide the functionality to create it.

c. **Functional Requirements**

Ordinary database clients have data export facility. However proposed tool expect to have more advanced features than just data exporting. Such as,

- Exported data has to be prearranged and should be able to easily import to a different database easily even without having the destination table. In other words data export should be self descriptive.
- Data export should consume less disk space than usual.

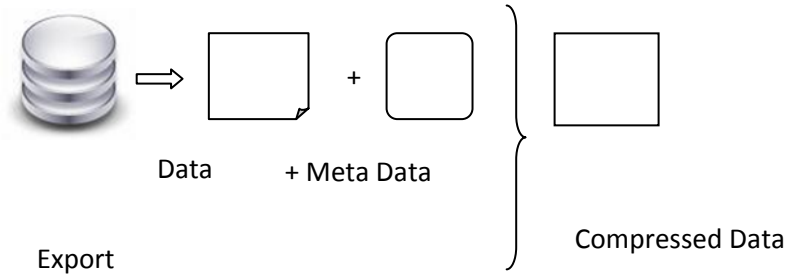


Figure 3.6 – Data Compression

3.3.10 System Enhancements

a. Description and priority

Product is intended to be developing as an open-source project. Therefore it is expected to enhance from time to time.

b. **Functional Requirements**

Interested software developers will extend and enhance the product. Particularly different plug-ins is expected to develop for communication with different databases. Also well defined database interface for database communication is must.

3.4 External User Interface Requirements

User Interfaces

Database integration system interact its users through a Graphical User Interface.

Content Presentation

Since developing product is database related, GUI should be optimized to use with different data views. Especially it should provide facility to view several result sets at the same time. So Multiple Document Interface Model should follow to display query results

and database integration results. Database connections are commonly used for most operations. Therefore it should be estranged from other interfaces. A tree view can be used to represent connected databases, tables and table columns. System should consist of a menu bar and the most frequently used items should be integrated to a tool bar.

When writing a SQL query or preparing a Federated Query user should be able to drag and drop tables or columns to required locations, as it append reference name to the existing text.

When processing an extensive task user should be clued-up about the stats using a progress bar. If the process is a combination of several tasks user should be able to see the current progress, for example completed task and its status.

Executing a command must be non-blocking. For an example it should be able to execute two database queries on separate query windows at the same time.

System should support usual copy, cut, paste, open, save, etc. In general when users look at the interface, they understand which pane is used for which purpose. Each task of an interface should be specified clearly and users should use them correctly. For example, when users press any button on interface, they should know which operations are associated with this button. The user interface should be easy to learn. When users use the user interface, they should know which element is used to which operations.

User instruction

Users should be able to get Online and Offline help. So system should include comprehensive instructions itself. And as an additional facility it is proposed to use a supervision window which always appear aside of the interface.

Software Interfaces

As typical database client software, Database Integration System will need to interface with a number of databases. For that developers required to implement JDBC [3]based interface for a particular database.

3.5 Other Nonfunctional Requirements

Performance Requirements

The system should use the minimum part of memory. The processes of the system should use the processor most efficiently. User should finish operation in the least time interval.

Safety Requirements

Database Integration System shouldn't make any connected database inconsistent. In any case of connection loss or operation abort, databases have to be in a consistent stage. System shouldn't do any partially committed operations.

In the integration all related data must be considered, In case of partial data unavailability system should abort the integration process. In the data transmission process required data should be fully transmitted to the destination, and shouldn't override or delete any existing data. And consistency and integrity of the internally stored data is significant.

Security Requirements

All stored passwords must be encrypted. If it is required exported data should be able to encrypt.

Quality Attributes

Other quality attributes system should consider.

Quality Attribute	Target User	Description
Availability	DBA	System expected to have 100% availability. In case of abnormal termination or misbehavior

[3] JAVA DATABASE CONNECTIVITY

		system should release all database connections and allocate memory.
Correctness	DBA / Developers	In database integration output should be exactly same as all tables were in a same database instance.
Maintainability	Developers	Source code must be well commented and modularized for further improvement.
Reusability	Developers	Developed component must have well defined interfaces.
Testability	Developers	All functions must be well defined and documented for testing.
Integrate-able	Developers	Developed component must have well defined and modularized.

Table 3.3 - Quality attributes