CHAPTER 3
THREE-TIER SERVICE ORIENTED BUSINESS INTELLIGENCE ARCHITECTURE

3.1 INTRODUCTION

The multi layered service oriented business intelligence (SOBI) architectures have a Reporting Data Store (RDS) which is accessed by business intelligence services for analysis. The data in RDS are populated using Extraction, Transformation and Load (ETL) service which provide an abstract view of the data source (Liya et al 2007, Lee-Kwun et al 2011, Alberto & Oscar 2012). The RDS is heavy and it is difficult to modify for dynamically changing requirements. This research proposes three-tier service oriented business intelligence architecture as shown in Figure 3.1. This architecture merges the logical layer and ETL layer into service layer to improve the response time of BI and make integrated data store portable.

The data tier contains the data source (semi structured and structured) and operational data store or integrated data store. The middle tier is the service tier where all business intelligence processes are provided as a service. The user or application tier acts as an interface for user or application and services.
Figure 3.1 Three Tier Service Oriented Business Intelligence Architecture

The business intelligence services include selection and discovery, composition, query decomposition, source identification, data extraction, data consolidation, data cleaning, data transformation, aggregation and summarization, log, metadata, ontology, schema mapping, OLAP, security, reporting, polling, data federation, data propagation, data loading, data analysis and ad-hoc query processing services. These are implemented using web services which inherit following characteristics: device independent, reusable, scalable, flexible, simple and consistent.

The user tier accepts the requests of users or application and process the request. The required attributes are extracted from the request for creating an integrated data store using appropriate service from service tier. The constructed integrated data store is analyzed for insight or knowledge and the results are communicated.
The service tier stores all services interfaces and services for all BI processes. This tier contains data structures to store binding information of the services along with service interface. Service and interfaces in this tier are created or advertised, modified, updated, discovered and combined as per requirement.

The data tier contains all data sources and the integrated data store, constructed for analysis. This tier adds new data sources, creates a data source, modifies the data source, converts the data sources to different format and permits BI services to extract, integrate and analyze data. This architecture can be used for strategic BI, tactical BI and operation BI.

3.2 METHODOLOGY

The proposed three tier SOBI dynamically constructs an integrated data store for the user request. This integrated data store is used for analysis. The data sources that participate in the analysis are converted into lightweight XML databases and these are updated along with OLTP to access live data.

Integrated data store is populated extracting the required contents from these XML data sources using sub queries for user request. This IDS is preserved to achieve portability inspite of extra space requirement. The contents of integrated data source are analyzed for extracting knowledge and insight using BI services. The flow diagram that illustrates the basic functionality and sequence of execution of the proposed system for data federation and data consolidation are shown in Figure 3.2 and Figure 3.3 respectively.
Figure 3.2 Flow diagram of proposed SOBI system for Data Federation
The services available in service tier will not have any user interface. The user tier permits the user or application to submit the request. These requests are communicated to service using programming interfaces. The request is initially processed and required attributes are extracted removing stop words from the request.

The attributes are used to create a global schema. A XML based integrated data store is created with the global schema. In this methodology,
the system creates a permanent or temporary integrated data store using consolidation or federation respectively (Calegari et al 2005, Stefan & Michael 2008).

3.2.1 Data Consolidation

Data consolidation is the process of constructing a permanent integrated data store extracting all data from data sources using global schema (Figure 3.4). The data consolidation is performed using two methodologies (i) creating new web service for populating the integrated data store (ii) modifying the existing online transaction processing (OLTP) module to push the contents to integrated data store. On either case, the global schema is to be designed and mapped to local schema, resolving heterogeneities like naming heterogeneity, schematic heterogeneity, structural heterogeneity and semantic heterogeneity.

![Figure 3.4 Integrated XML data store using Consolidation or Federation](image-url)

<?xml version="1.0"?>
<xml>
  <Cars>
    <Car>
      ...
    </Car>
    ...
  </Cars>
</xml>

Original Data Source

Location – 1
Data source

Location – 2
Data source

Location – n
Data source

Converted XML Data Source

Data Consolidation / Federation

Extraction and Loading

<?xml version="1.0"?>
<xml>
  <Cars>
    <Car>
      ...
    </Car>
    ...
  </xml>
</xml>

(Location – 1, 2, n)

Data Consolidation / Federation

Extraction and Loading

<?xml version="1.0"?>
<xml>
  <Cars>
    <Car>
      ...
    </Car>
    ...
  </xml>
</xml>

(Location – 1, 2, n)

Data Consolidation / Federation

Extraction and Loading

<?xml version="1.0"?>
<xml>
  <Cars>
    <Car>
      ...
    </Car>
    ...
  </xml>
</xml>

(Location – 1, 2, n)
Steps involved in Data Consolidation are listed below:

**Step 1:** Creation of a global schema that satisfies all decision support and analytics requirements.

**Step 2:** Creation of a XML Database using the global schema.

**Step 3:** Mapping local database attributes with XML database attributes.

**Step 4:** Populating the XML database with data source contents using required transformation and loading.

Data consolidation process stores the integrated data store in the data tier and permits user to analyze and extract the knowledge for the queries. However to handle dynamic business analytics the integrated data store is to be rebuilt. Similar to data consolidation, the data federation integrates the data for analysis.

### 3.2.2 Data Federation

The data federation constructs a virtual view based on user request (Sheth & Larson 1990). Steps involved in data federation are listed below:

**Step 1:** Receiving the query from the user.

**Step 2:** Identification of the required attributes from the query.

**Step 3:** Creation of a global schema for federation.

**Step 4:** Identification of all data sources and extraction of local schemas.

**Step 5:** Mapping the global schema attributes with respective local schema attributes.
**Step6:** Extracting the required records from data source and populating the integrated data store for analysis.

Unlike consolidation, the federation constructs a transient integrated data store based on user request. This integrated data view is cached for future reference. To improve the performance of OLTP application and also support decision support system, the data sources that participate in the federation are converted into XML data sources from their native state as shown in Figure 3.4. These XML data sources are updated frequently or on demand basis.

An alternative methodology that uses a web service to push the updated content from the database to an XML data source shall be deployed along with OLTP application. The SOBI architecture uses these XML data sources for data federation rather using the native database. These are easy to replicate and consume less space than the original database. Various modules involved in the creation of the integrated data store are discussed in the following subsections.

### 3.2.2.1 Global schema construction

The global schema construction process constructs global schema using the extracted input attributes. The obtained user input for data federation is processed and the attributes are identified. The service priority is assigned based on the attribute of user choice.

An XML based integrated data store using the extracted attribute is created for storing federated data which is stored in data tier of SOBI.
3.2.2.2 Schema mapping and sub query construction

All participating data sources follow their own schema (local schema) which typically differs from the global schema. Schema mapping service checks the availability of global schema attribute with the local schema attributes and constructs the sub query to the respective data source.

The schema mapping service uses a schema mapping table for identification of local schema attributes. Each global schema attribute is mapped with the local schema attribute of respective data sources. The mapping table construction process is accomplished with domain expert and is semi-automatic.

The domain expert assign the local schema attribute for corresponding global schema attribute using Meta data or data dictionary of data sources. The data type and data size of heterogeneous data sources are verified using an automated data verification service. The identified attributes of data sources through mapping process is used to construct sub query.

The sub query for each data source is constructed using XPath by appending the required attributes that exist in global schema. Functional dependencies and derived attributes of local schemas are used during attribute selection to include more databases in data integration.

3.2.2.3 Data extraction service and data federation

Data Extraction service accepts the sub-query and uses an XPath query to navigate the respective local XML data sources and extracts the contents. These contents are stored as a separate data set and integrated. Steps involved in Data Extraction Service are listed below,
Step 1: Execution of each XPath sub query over respective XML data sources for extraction of required records.

Step 2: Storing the extracted contents in respective global schema attribute using the mapping table created in schema mapping.

Step 3: Repeat Step 2 until extracted contents of all data sources are populated.

3.3 EXPERIMENTAL RESULTS

The proposed architecture provides a framework that allows heterogeneity, integration and reusability of the participant components in a flexible environment. Data federation is implemented in the proposed three-tier Service-Oriented Business Intelligence Architecture.

For experimentation, Car Re-sales databases of seven organizations are considered. The experiment was conducted in a Local Area Network (LAN) with one server containing required BI services.

Car re-sales databases are stored in different workstations using MySQL, Oracle, MS Access, PostgreSQL, MS SQL Server, SQLite and IBM DB2 respectively. The database schemas of these databases are shown below,

i. **Car.salesdb**\{*Make, Model, Year, Fuel_type, Acceleration, No_of_Doors, Body_Style, City_Mileage, Highway_Mileage, Original_Cost, Resale_Cost, Owner_Name*\}

ii. **Resales**\{*R_ID, Manufacturer_Name, Model, Fuel_Type, Year, Age_of_car, Body_Style, Original_cost, Depreciation, Owner_Details, Mileage, Seating_Capacity*\}
iii. **Car.Resales**\{Make, Model, Year_of_Manufacture, Fuel_Type, Mileage, No_of_Doors, Original_Cost, Resale_Cost, Owner_Name\}

iv. **Car.Resalesdb**\{ID, Make, Model, Year_of_Manufacture, Fuel_Type, Mileage, Body_Style, Original_Cost, Resale_Cost, Owner_Name\}

v. **Resales.Cardb**\{Make, Model, Year_of_Manufacture, Fuel_Type, Mileage, No_of_Doors, Original_Cost, Resale_Cost, Owner_Name\}

vi. **Cars.db**\{ Make, Model, Year, Fuel_type, Aspiration, No_of_Doors, Age_of_car, Body_Style, Mileage, Original_Cost, Resale_Cost, Owner_Name\}

vii. **Resales.Carsdb**\{ID, Make, Model, Year_of_Manufacture, Fuel_Type, Mileage, Body_Style, Original_Cost, Resale_Cost, Owner_Name\}

The database schemas (local schema) of relational databases are stored in a data dictionary. The local schema will be updated when there is change in the relational database design. Using this schema, XML database is constructed. The XML schema for ‘Car.salesdb’ is shown in Figure3.5. These XML data sources are populated with existing records of local database using loading service.
The XML data sources are updated along with relational data sources using the update service which will delete, modify and append records to ensure consistency. The missing values in XML data sources are filled with average values and frequently used values. For instance, the missing values in Mileage attribute are filled with average values of similar records and Body Style attribute value is filled with frequently available data of similar records.
Since services do not have any interface, a user interface is provided to the user to submit query. This user interface accepts two types of queries where the user can give attributes for data integration or provide specific values along with attributes. For instance, consider the query with attributes Model, Make, Mileage and Resale Cost. The global schema to construct an integrated data store for the instance is shown in Figure 3.6.

```xml
<?xml version="1.0"?>
<Cars>
  <Car>
    <Make/>
    <Model/>
    <Mileage/>
    <ResaleCost/>
  </Car>
</Cars>
```

**Figure 3.6 Global Schema for integrated Data Store (For attributes Make, Model, Mileage and Resale Cost)**

The mapping table is constructed using attribute matching, derived attributes and functional dependencies. The mapping table can be created either manually or automatically; here manual mapping table is constructed for experimentation. For instance, consider Age of Car attribute value can be derived from Year of Manufacturing. A subset of attributes considered for experimentation is shown Table 3.1.
Table 3.1 Mapping table for global schema and attributes of Car.SalesDB and Resales local Schemas

<table>
<thead>
<tr>
<th>Global Schema Attributes</th>
<th>Local Schema Attribute for Data Source -1</th>
<th>Local Schema Attribute for Data Source – 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make</td>
<td>Make</td>
<td>Manufacturer_Name</td>
</tr>
<tr>
<td>Model</td>
<td>Model</td>
<td>Model</td>
</tr>
<tr>
<td>Year</td>
<td>Year</td>
<td>Year</td>
</tr>
<tr>
<td>Fuel Type</td>
<td>Fuel_Type</td>
<td>Fuel_Type</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Acceleration</td>
<td>-</td>
</tr>
<tr>
<td>No of Doors</td>
<td>No of Doors</td>
<td>Derived from Body Style*</td>
</tr>
<tr>
<td>Age</td>
<td>Derived from Year*</td>
<td>Age_of_Car</td>
</tr>
<tr>
<td>Original Cost</td>
<td>Original_Cost</td>
<td>Original_Cost</td>
</tr>
<tr>
<td>Resale Cost</td>
<td>Resale_Cost</td>
<td>Derived From Original_Cost and Depreciation*</td>
</tr>
<tr>
<td>Seating Capacity</td>
<td>-</td>
<td>Seating_Capacity</td>
</tr>
</tbody>
</table>

* Attributes value is to be derived from other attributes of respective data sources.

Priority is assigned to all global schema attributes. These attributes are verified with local schema attributes using the Data Source Verification Service. If higher priority attribute is missing in the local schema, then that data source is rejected from data federation. For instance, as the “Seating Capacity” is not available in Car.Salesdb, so it is removed from data federation. A separate instance of Data Source Verification Service is created for each data source and verified simultaneously.
The Federation Service extracts the data by creating and executing sub queries using XPath. A separate instance of each sub query created and executed over the respective data source is available in different workstations. The extracted records are populated in the integrated data store for analysis and decision support.

The experimentation for data federation was started by populating 50 records in each data source and Data federation is performed for the attributes Make, Model, Year and Resale cost. The process is repeated by appending 25 records till number of records reach 250 in each data source. The response time of data federation of the proposed methodology with XML data source over SOBI architecture is compared with existing data federation methodology (Liya et al 2007) that uses the original data sources and database controllers over five layered architecture which is shown in Figure 3.7.

![Graph showing response time of data federation](image)

**Figure 3.7** Response Time of Data Federation (For attribute Make, Model, Year and Resale Cost)

Using the XML data source and proposed federation methodology over three tier SOBI architecture, the response time is improved by 12.1%
than the existing methodology that uses five layered architecture with relational databases and ODBC controllers.

The proposed system uses derived data of each data source for data federation. This process increases the number of data sources in federation which, in turn, increases the number of records retrieved from the sources. Experimentation is conducted with various combinations of attributes without specific values for attributes and the number of records retrieved from data sources are tabulated in Table 3.3.

Table 3.2 Number of Records retrieved with and without, Derived Attributes of data source

<table>
<thead>
<tr>
<th>Total No. of Records in all 7 Data Sources</th>
<th>Attributes in Global Schema</th>
<th>Number of records retrieved from data sources</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Make, Model, Year and Resale Cost</td>
<td>Without Derived Attributes: 1500, With Derived Attributes: 1750</td>
<td>Resale cost for Resale database is derived from Original_Cost and Depreciation</td>
</tr>
<tr>
<td></td>
<td>Make, Model, Year, and Body Style</td>
<td>Without Derived Attributes: 1250, With Derived Attributes: 1250</td>
<td>Body_Style attribute is not available and cannot be derived in Resales.Cardb and Car.Resales data sources</td>
</tr>
</tbody>
</table>
For the first query ‘Resale’ data source is not included in data federation since Resale_Cost is not available in the data source, but it can be derived using Original_Cost and Depreciation attributes. In second query, the Body_Style cannot be derived from any attribute, so two sources will not participate in data federation.

Since all data sources are converted into lightweight XML data sources the space required for respective data source is less than that of the original data sources.

3.4 BENEFITS OF THREE TIER ARCHITECTURE

- The three tier SOBI architecture is extendable for efficient service discovery and composition.

- The proposed system creates a dynamic Light weight integrated data store which consumes less space and it follows the open standard which is reusable, portable and easily replicated in various geographical locations.

- The BI users need to pay the provider only for the duration to which service is used.

- All services in service tier are reusable, discoverable and composable.

- The XML data sources and the integrated data stores are transparent. The users or application can directly interact with data sources using data service.
3.5 SUMMARY

Three tier SOBI architecture is designed and implemented to cater to the needs of Business Intelligence Operations. The architecture converts the native data sources into XML databases and stores in the data tier and is updated using push methodology. The BI services from service tier use these data sources for Analysis, extraction of Knowledge and insight. The Services in service tier are implemented using Web services which inherit characteristics like loose coupling, autonomy, abstraction, reusability, statelessness, discoverability and composability. The system creates a XML based light weight dynamic integrated data store based on user request for decision support. Data federation is experimented over the proposed three tier SOBI architecture and experimental results determine the response time for data federation which is 12.1% better than the existing five layered SOBI architecture that uses relational data sources and data controllers.