IBM® Industry Models

Best practices: Deploying the IBM Banking Data Warehouse to IBM InfoSphere BigInsights

Austin Clifford
Lead Data Warehouse Specialist
IBM Dublin Lab

Issued: March 2014

Table of Contents

Introduction

An IBM industry model solution is a comprehensive set of industry-specific predesigned models that form the basis of a business and software solution, optimized for business challenges in a particular sector. Domain areas include data warehousing, business intelligence, business process management, service-oriented architecture, business terminology, and business glossary templates.

The IBM Industry Models solutions covers a range of industries that include banking, health care, retail, and telecommunications. The IBM Industry Models solutions provides you with an extensive and extensible data model for your industry sector. Use the logical data model as provided by IBM to build a physical data model that is customized for your reporting requirements, and then deploy and populate an IBM InfoSphere® BigInsights™environment.

This paper shows how you can deploy the IBM Banking Data Warehouse (BDW) solution to BigInsights. This paper introduces the logical data model concept and then focuses on what you must do to transform a non-vendor-specific logical data model into a production-ready BigInsights BigSQL schema. The key steps of the deployment process are as follows:

1. Create a physical data model that is a subset of the supplied logical data model.
2. Refine the physical data model to reflect your reporting and analytics needs.
3. Create the physical schema from the physical data model.
4. Populate the test environment, to further optimize the physical design to reflect the anticipated query, ingest, and maintenance workload.

The example used in this paper is the Involved Party coverage within the BDW.

Although BigInsights includes a SQL interface (BigSQL), there are unique considerations when deploying to BigInsights. This paper explains how to translate reporting needs into BigInsights design decisions as part of the transformation process.

InfoSphere BigInsights

InfoSphere® BigInsights™ is a software platform for discovering, and visualizing data from disparate sources. You can use this software to help process and analyze the volume and variety of data that continually enters your organization. BigInsights is built on the scalable Apache Hadoop open source framework that runs on commonly available, low-cost hardware. By using BigInsights, users can extract new insights from data to enhance knowledge of your business.

BigInsights includes text analytics for analyzing large volumes of text, which often contains extraneous data ("noise") that is not important to your business, to gain insights into unconventional data. You can incorporate these capabilities into your IT infrastructure to complement and extend existing analytic capabilities by augmenting your data warehouse with these insights. You can use BigInsights as an engine to store, filter, analyze, and transform incoming data. You can then extract business insights and important summary data and then load into your data warehouse. You can also use BigInsights as a query-ready archival system for your data warehouse to quickly access data that is typically archived. You can offload data to BigInsights, where you can query it at any time. This implementation saves time and resources by integrating into your existing architecture.

BigSheets is a browser-based analytic tool that is included in the BigInsights console. You can use BigSheets to collect data from multiple sources, format it, and explore it by using a spreadsheet-like interface. You can also apply visualizations such as tag clouds, bar charts, maps, and pie charts to provide consumable output that highlights relationships and distill insights from previously disconnected data.

Of particular relevance to this paper is BigSQL. BigSQL is a software layer that you can use to create tables and query data in BigInsights by using familiar SQL statements. BigSQL uses standard SQL syntax and, in some cases, SQL extensions that IBM created to make it easy to exploit certain Hadoop-based technologies. The BigSQL query engine supports joins, unions, grouping, common table expressions, and other familiar SQL expressions. Depending on the nature of a query, the data volumes, and other factors, BigSQL can use Hadoop's MapReduce framework to process various query tasks in parallel or run a query locally within the BigSQL server on a single node whichever is the most appropriate for the query.

The Banking Data Warehouse

The BDW is an industry-specific blueprint that provides data warehouse design models, business terminology and analysis templates to help accelerate the development of business applications. The software can improve reporting, credit and risk management, and consolidate trusted information across multiple viewpoints.

The BDW contains support for social media through the requirements, analysis, and design models. The BDW allows a financial institution to harness social media data into a form that it can use to derive real business insight. This paper uses social media as an example of one of the many types of information in the BDW that can be deployed to BigInsights.

The BDW social media content has various business applications. For example, a financial institution might want to calculate various social media metrics such as share of voice and reach or distinguish between positive, negative, and neutral postings on the company's page. A financial institution might also want to understand what trends and patterns are emerging so that it can make better business decisions.

The model content also allows a financial institution to make the link between a customer and a social media persona, bringing together the traditional sources of data with the new social media data that's available. The financial institution can use this extended view of the customer to identify any concerns early that might affect a customer's creditworthiness. Social persona analysis also allows new relationships to be uncovered and identifies possible cross-selling opportunities that are linked to life events that are uncovered though social media.

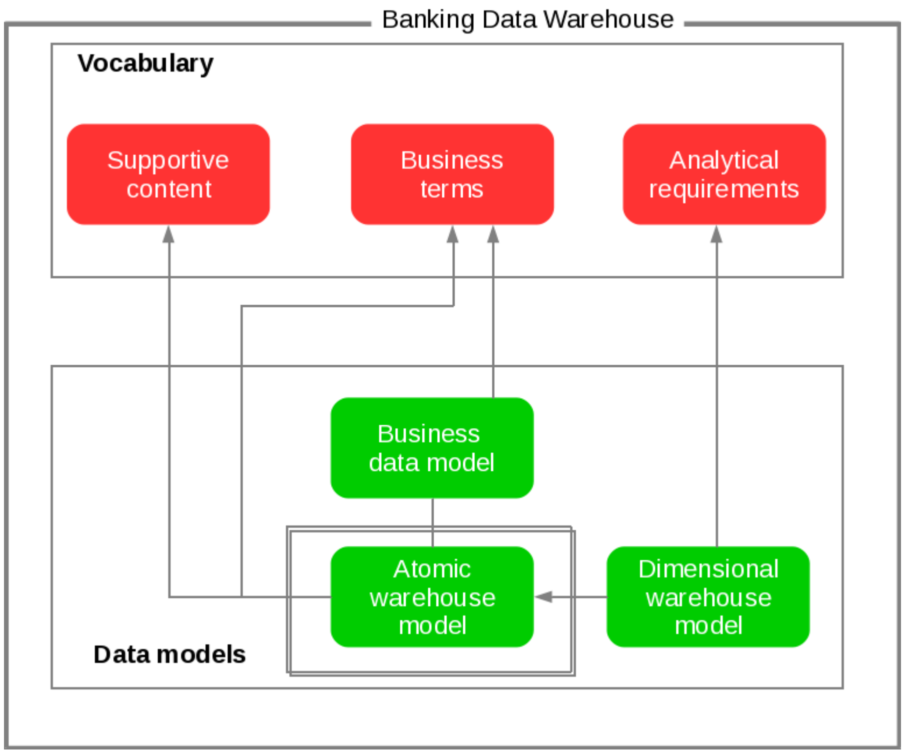


Figure 1. The components of the BDW

Figure 1 illustrates the various components of the BDW. The vocabulary describes the business content that is supported by the data models, and provides a consistent terminology and helps to understand the information used by related IT assets.

The business terms are industry concepts in plain business language and with no modeling. The mapping of business terms to the data models allows the transformation of requirements into IT data structures. Analytical requirements are high level groups of business information to express business measures along axes of analysis, which are named dimensions. The analytical requirements are the basis for building the dimensional warehouse model.

The data models are used to build reporting solutions, such as business intelligence and standardized reporting. They are mapped back to the vocabulary and act as a bridge between the business terminology and the deployed IT assets. The data models consist of business, atomic warehouse and dimensional warehouse models. Of particular relevance to this paper is the atomic warehouse model, which is described in the next section.

The atomic warehouse model

The following figure shows the atomic warehouse model (AWM) within the context of the IBM Big Data reference architecture.

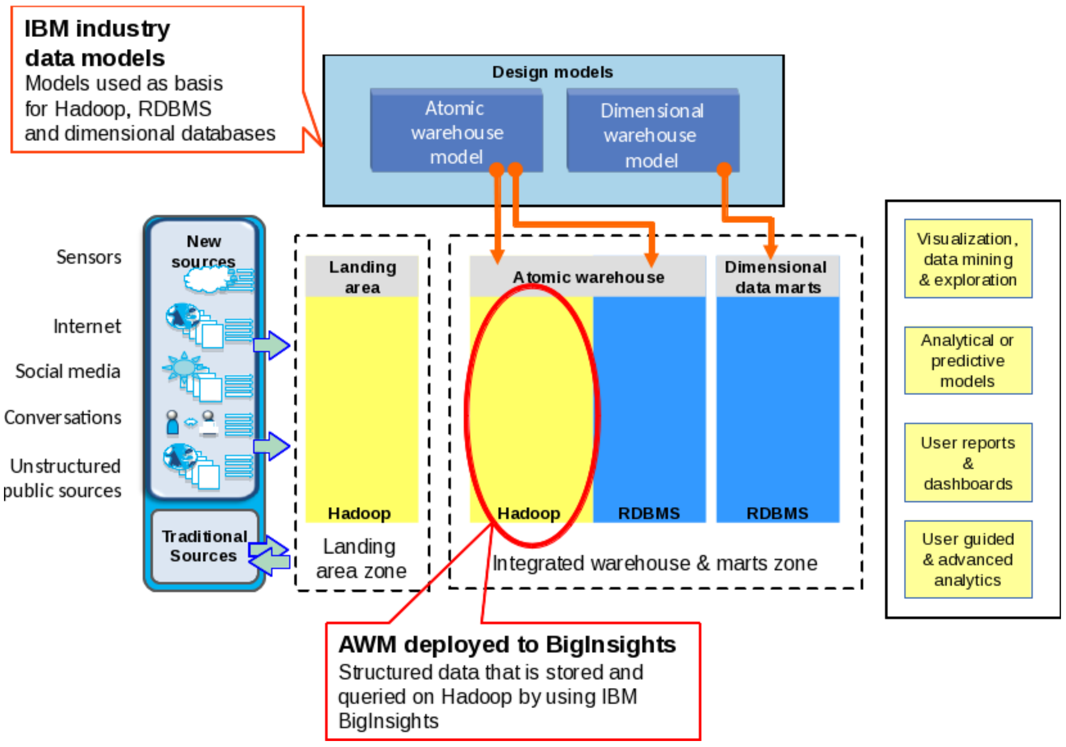

IBM industry data models
Models used as basis for Hadoop, RDBMS and dimensional databases
Design models
Atomic warehouse model
Dimensional warehouse model
Sensors
Internet
Social media
Conversations
Unstructured public sources
New sources
Traditional Sources
Landing area
Hadoop
Landing area zone
Atomic warehouse
Hadoop
RDBMS
Dimensional data marts
RDBMS
Integrated warehouse & marts zone
Visualization, data mining & exploration
Analytical or predictive models
User reports & dashboards
User guided & advanced analytics
AWM deployed to BigInsights
Structured data that is stored and queried on Hadoop by using IBM BigInsights

Figure 2. The AWM within the context of the IBM Big Data reference architecture

Data moves from left to right across the landscape depicted in the figure. Data received from both traditional and new sources is initially persisted in the landing area zone, before moving through the integrated warehouse and marts zone. The data models are applied to both the atomic warehouse and dimensional data marts. Data scientists and business users interact with the landing area, integrated warehouse, and data marts zones for performing reporting, visualization, and advanced analytics on the data.

The AWM is a design-level data model that represents the enterprise-wide repository of atomic data that is used for informational processing. This informational processing includes the changes to business information that the business wants to track for analytical purposes. The AWM is highly normalized, although it might contain some duplication and derivation, for ease of navigation and performance. In contrast to the dimensional warehouse model (DWM), which is optimized for analytics at the speed of business, the AWM is typically applied at the detailed enterprise system-of-record level.

BigInsights is a natural deployment choice for parts of the AWM that involve large volumes of data that is not response-time critical and that might have an indefinite retention requirement.

Implementing the BDW

Implementing a logical data model as a physical schema presents technical challenges. There are several steps through which you create and optimize your physical schema for production use.

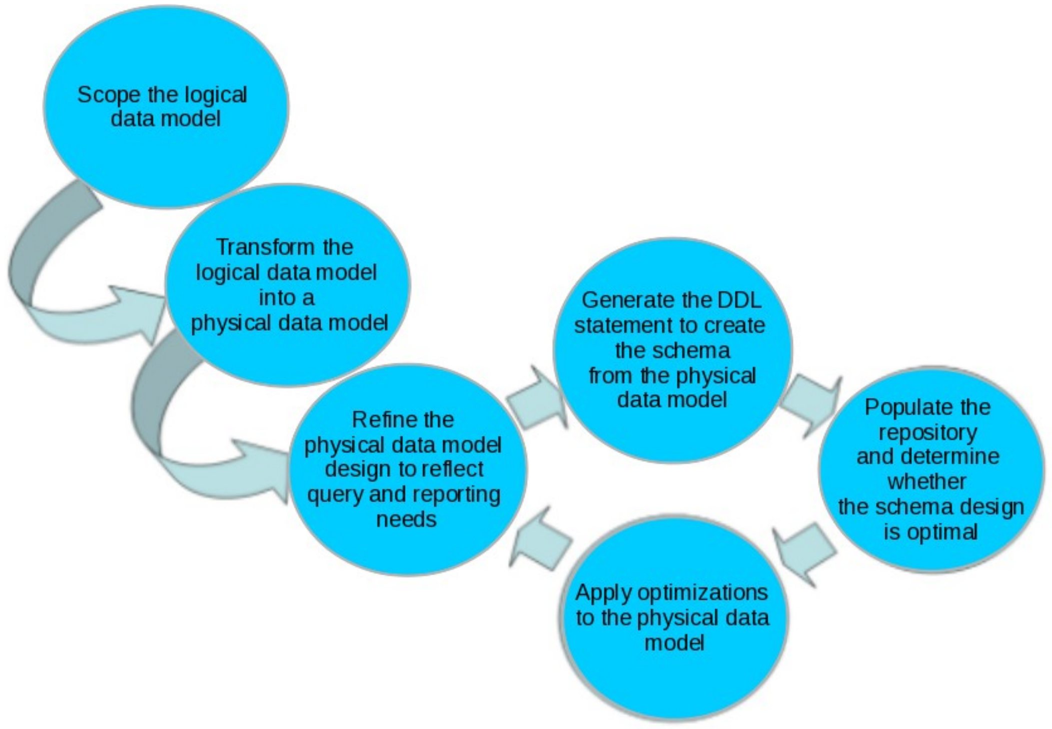

Scope the logical data model
Transform the logical data model into a physical data model
Refine the physical data model design to reflect query and reporting needs
Generate the DDL statement to create the schema from the physical data model
Populate the repository and determine whether the schema design is optimal
Apply optimizations to the physical data model

Figure 3. Typical deployment patterns for creating a physical repository

This process includes the following phases:

1. Scoping the logical data model and transforming it into a physical data model.

   Mapping reporting requirements to the logical data model to determine the scope of the model. Include only those entities and attributes for which you have a reporting requirement.

2. Transforming the logical data model into a physical data model.

   When you complete the process of scoping, you can transform your logical data model into a physical data model by selecting a menu option in IBM InfoSphere Data Architect.

3. Refining the physical data model to reflect your query and reporting needs.

   A number of modifications are possible to improve the physical data model which include data types, denormalization, table type, external tables and location, storage format, compression, and partitioning.

4. Generating the DDL statements to create the schema from the physical data model.

5. Populating the repository and determining whether the schema design is optimal.

6. Applying optimizations to the physical data model.

IBM InfoSphere Data Architect is a data design tool that you can use to scope, transform, and customize the data models that are supplied in the BDW solution. The examples in this paper reference InfoSphere Data Architect. For more details about the product, see the “**Further reading**” section.

Scoping the logical data model

The logical data model is designed to meet all aspects of reporting for an industry sector. Your enterprise might not need all of the objects that are provided. Scoping is the process, by using InfoSphere Data Architect or other modeling tools, of selecting those entities from the logical data model that align with your analytical requirements.

Refine your scope as much as possible to address your current data and reporting needs.

When you use InfoSphere Data Architect to scope the BDW to create your logical data model, follow these steps:

1. Create a diagram into which you can drag those entities that you need to address your warehousing and reporting needs.

   Creating a diagram for your logical data model avoids directly changing the base model. By using this method, you can more easily accept future BDW upgrades.

2. Navigate through the AWM, using InfoSphere Data Architect to identify and include all related entities in your diagram. Avoid manually moving individual related entities, because this can affect the integrity of the resulting schema.

In the BDW example, the business requirement is to capture social media postings, initially from Twitter, to track social media sentiment regarding large corporate customers' creditworthiness. Specifically, the requirement is facilitate reporting to include a subset of attributes from the Twitter JSON schema, as shown in the following example:

```
{
   "created_at":"Fri Nov 05 20:21:12 +0000 2012",
   "id":28174561684,
   "in_reply_to_status_id":null,
   "retweet_count":0,
   "retweeted":false,
   "source":"<a
href=\"http://twitter.com/#!/download/iphone\"
rel=\"nofollow\">Twitter for iPhone</a>",
   "text":"The actions of Carbon Winter Melon GmBH in
Antarctica are obscene",
   "user":{
```

```
            "created_at":"Fri Dec 11 16:46:29 +0000 2009",
            "id":17077356,
            "followers_count":657,
            "following":null,
            "friends_count":265,
            "screen_name":"EDDIE8704"
        }
    }
```

Additionally, a daily summary is required to include the number of connections, as well as an influence score for the Twitter user. The influence score can be derived as a numeric score from several different measures that are used to assess the influence of a social media persona.

Three main entities in the AWM are accordingly identified as candidates to be scoped to meet the reporting need:

- **Social Media Post**

  A Social Media Post identifies a communication by a social media persona whose purpose is to post a message to an online community, for example, a “tweet” on Twitter.

- **Social Media Persona**

  A Social Media Persona is an online profile for which an organization can store or process information. A Social Media Persona can exist independently of any real-world individual or organization. Alternatively, an individual or organization can have many different Social Media Personas. You can link Social Media Personas to other Involved Party entities through the Involved Party or Involved Party Rltnp associative entity.

- **Social Media Persona Summary**

  A Social Media Persona Summary is a summary of activity that is related to Social Media Personas for a particular time period. This summary contains key measures such as the number of connections and number of posts for specific measurement periods and scenarios.

For each of these three entities, the following attributes are scoped:

Entity	**Attribute**	**Twitter JSON**
Social Media Post	Social Media Post ID	id
Social Media Post	Related Post ID	in_reply_to_status_id
Social Media Post	Post Text	text
Social Media Post	Post Date	created_at
Social Media Post	Social Media Persona ID	User.id
Social Media Post	Social Media Website ID	User.source
Social Media Post	Social Media Post Type ID	Default value

Social Media Persona	Social Media Persona ID	User.id
Social Media Persona	Name	screen_name
Social Media Persona	Effective Date	created_at
Social Media Persona	Reliability Rating ID	Calculated
Social Media Persona Summary	Social Media Persona ID	User.id
Social Media Persona Summary	Measurement Period ID	Calculated
Social Media Persona Summary	Closing Number of Connections	User.friends_count
Social Media Persona Summary	Population Date	Calculated
Social Media Persona Summary	Population Time	Calculated
Social Media Persona Summary	Source System ID	Default value
Social Media Persona Summary	Influence Score	Calculated
Social Media Persona Summary	Source Veracity ID	Calculated

The resulting logical data model diagram is as follows:

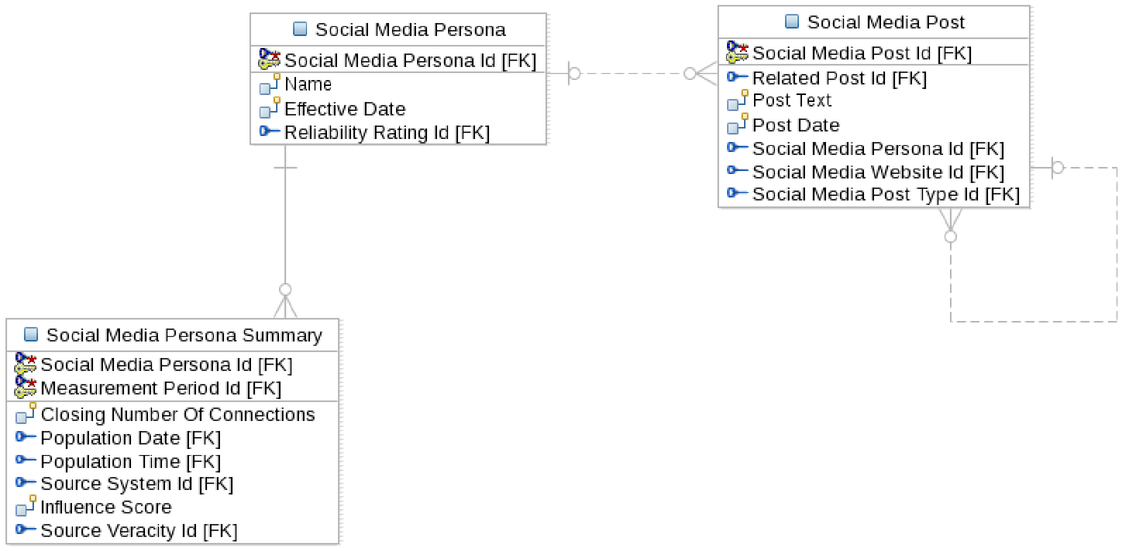

Social Media Persona
Social Media Persona Id [FK]
Name
Effective Date
Reliability Rating Id [FK]
Social Media Post
Social Media Post Id [FK]
Related Post Id [FK]
Post Text
Post Date
Social Media Persona Id [FK]
Social Media Website Id [FK]
Social Media Post Type Id [FK]
Social Media Persona Summary
Social Media Persona Id [FK]
Measurement Period Id [FK]
Closing Number Of Connections
Population Date [FK]
Population Time [FK]
Source System Id [FK]
Influence Score
Source Veracity Id [FK]

Figure 4. The scoped logical data model

Transforming the logical data model into a physical data model

Because the logical data model applies to all databases, minimize the database architecture and design changes that you make to the logical data model. Instead, implement as many changes as possible in the physical data model. This strategy provides the following benefits:

- An easier upgrade strategy for future releases of the BDW because only semantic differences will exist between your model and the BDW.
- The ability to focus technical modeling effort on the physical data model and retain a logical data model that is suited for all databases.
- More easily controlled changes, in both the logical and physical data models, by assigning clear roles to each model. The logical data model functions as a semantic master, and the physical data model is the technical master.

To help accommodate future upgrades, apply architecture and design changes that are specific to BigInsights to the physical data model rather than the logical data model.

You can add entities to the diagram later and merge them into a physical data model by using the compare and merge functionality in InfoSphere Data Architect.

Transform your logical data model into a physical data model as follows:

1. Select a blank area in the diagram.
2. From the InfoSphere Data Architect main menu, click **Data > Transform > Physical Data Model.**
3. When prompted, provide details to complete the transformation process:
    a. Select **BigInsights** as the database and select the version that you require, for example 2.1.
    b. Accept the default values of the remaining settings, other than perhaps taking advantage of the opportunity to set the target schema name, for example, BDW_ATOMIC.

The transformation process converts the entities and their attributes from the logical data model into an equivalent set of tables and associated columns in the physical data model.

During this process, the surrogate key and identity column properties are disabled in the transformation wizard. This is because identity columns are not supported in BigSQL. Moreover, Hadoop is not optimized for surrogate key assignment. In general, surrogate key assignment is better left to either RDBMS or conventional ETL tooling such as IBM DataStage™. Therefore, although the BDW supports the use of both surrogate and natural keys, in the example in this paper natural keys are used.

The AWM uses a combination of approaches for managing historical change, including record updates with effective dates, snapshot tables, and periodic history tables. The implementation of some of these approaches relies on database features such as surrogate keys and record updates. In addition to the considerations around surrogate keys, record updates are not always supported in Hadoop. Therefore, it might not be appropriate to use Hadoop for data where a history of change is required.

Refining the physical data model to reflect query and reporting needs

There are a number of refinements possible to the physical data model before generating the DDL statements to create the schema. These modifications reflect the expected query and reporting needs and include considerations for data types, denormalization, table type, external tables and location, storage format, compression, and partitioning.

Data types

BigInsights supports most standard SQL data types. There is support for INTEGER, DECIMAL, floating-point numeric, CHAR, VARCHAR, TIMESTAMP, and BINARY types.

However, BigSQL does not have a DATE or TIME data type, though there is support for a TIMESTAMP data type. This data type requires the following format: *yyyy-mm-dd hh*:*mm*:*ss*.*fffffffff*, where *fffffffff* is an optional fraction of a second. Therefore, if the logical data model represents a time stamp as separate DATE and TIME attributes, merge these attributes into a single TIMESTAMP column for BigSQL. You should do this in the physical data model. For example, in the SOC_MEDIA_PSNA_SMY table in the BDW, the Population Date (PPN_DT) and Population Time (PPN_TM) columns are merged into a single column called PPN_TMST, which has the format *yyyy-mm-dd hh*:*mm*:*ss*.

Another restriction is that BigSQL does not support large objects. The maxium length of VARCHAR and VARBINARY columns is 32,768 bytes.

Merge separate DATE and TIME attributes into a single TIMESTAMP column in the physical data model.

Denormalization

Joins are a relatively costly and inefficient operation in Hadoop, in particular for large tables. If your query response times are unacceptable, you might need to denormalize the schema. This typically involves introducing controlled redundancy to reduce the number of table joins that are required in particular queries, which can significantly improve the query response times.

Although denormalization can improve query response times, it can incur additional processing overhead at load time, so there is a trade-off in performance. Also, there are many important reasons for maintaining normalization, including a conceptually cleaner data model, less data redundancy, reduced risk of update anomalies, and easier maintenance.

In general, do not denormalize unless it is the only way to obtain the required performance. Carefully document any denormalization that you undertake. In the case of the BDW example scenario, the query response times are deemed adequate for the envisaged analysis query pattern, such that denormalization is not required.

Consider denormalizing for improved query performance. Do not denormalize unless you need to.

Table type

BigInsights supports both Hive and HBASE-managed tables.

Hive tables have the following properties:

- Data is stored as files in a Hadoop Distributed File System (HDFS).
- They support appends only; they do not support updates.
- They are optimized for high throughput scans, where all or a large portion of the data is being analyzed.
- External tables allow multiple versions of schemas to be asserted for the same data set. Also, dropping the tables does not delete the files.
- They support a variety of file and row formats.
- They are extremely fast to load; simply upload data files to the HDFS.
- They support partitioning.
- They can be consumed directly by BigSheets.

HBASE tables have the following properties:

- They support indexes: both primary key and secondary indexes.
- They support updates and inserts.
- They are optimized for low latency point lookups based on key or targeted range scans.
- They use a column family architecture, which is very suited to wide, sparse data sets.
- They use a flexible schema. You can add columns to families at any time, enabling the table to adapt to changing application requirements.
- Internally, they can be verbose. All cell values are stored with the full coordinates, including the key, column family name, column qualifier, and time stamp. This metadata increases disk storage overhead, particularly for data sets that are not sparse, although compression and BigSQL many-to-one column mapping can minimize this overhead.
- They are slower to load than Hive tables, in particular if secondary indexes are defined.
- They do not support partitioning.
- They cannot be consumed directly by BigSheets.

For the BDW example, the superior throughput for large analytical data warehouse queries, together with the ability for tables to be consumed directly by BigSheets, favors the choice of Hive tables for deployment. InfoSphere Data Architect generates Hive tables by default for BigInsights.

Use Hive tables rather than HBASE tables for data warehouse style queries and convenient sharing with applications such as BigSheets.

Table location and external tables

When you create a Hive table, you can specify a table location in the CREATE TABLE statement. If you do not provide a table location, the BigSQL server chooses a location in the appropriate BigInsights warehouse directory.

Creating an external table means creating a table that points to an existing directory that contains data files. Creating an external table does not change the data files, and dropping the table does not remove those files. Multiple external tables can point at the same data files. Sometimes this is convenient to provide multiple views over the same data.

For the BDW example, the tables are created as `EXTERNAL` with the storage `LOCATION` set to '`/banking/BDW/AWM/BDW_ATOMIC.`*`tablename`*'. These are specified in the physical data model using InfoSphere Data Architect.

Use external tables for convenient assertion of schemas and the flexibility to define multiple views over the same data.

Storage format

BigSQL supports several stored file formats for Hive tables, including the following formats:

- TEXTFILE. A regular delimited text file. This format is the default.
- SEQUENCEFILE. A binary key-value row-oriented format.
- RCFILE. A binary column-oriented format that permits columns that are not accessed in the query to be skipped. RCFILE is an efficient format for wide tables for which only a few columns are typically queried.

Although TEXTFILE isn't always the most efficient storage format, it is flexible and offers convenient consumption by applications such as BigSheets. Therefore, for deploying the BDW, TEXTFILE tables are the favored option. InfoSphere Data Architect generates the TEXTFILE format by default.

Use the default TEXTFILE file format for convenient sharing with applications such as BigSheets.

A tab-delimited row format was chosen for the BDW example. You specify this format by using the `ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'` clauses for the CREATE TABLE statement in InfoSphere Data Architect. BigSQL also supports collection items and map keys, and you can specify separate delimiters for these in the `ROW FORMAT` clause.

Using InfoSphere Data Architect, you can also specify a custom Serializer-Deserializer (SerDe) Java class in the BigSQL CREATE TABLE statement to access JSON or pretty much any row format directly. However, the approach adopted for the BDW is to shred and flatten the JSON format to a tab-delimited TEXTFILE format and assert the Social Media tables schema by using the high-level JAQL scripting language.

Compression

Compression can significantly reduce the storage footprint of large data sets. BigInsights supports a number of compression coder-decoders (codecs). The favored option is the BigInsights LZO-based codec, which uses the CMX file suffix. This option supports splitable compression for the TEXTFILE format. Without splitable compression, a single mapper must decompress the source data set, which prevents parallelism and has a consequent impact on performance. Both BigSheets and BigSQL can transparently decode the compressed CMX data sets.

Although you can apply compression at the individual file or table level, you can more conveniently apply it at the cluster level, by setting the compression properties in the BigInsights configuration files, mapred-site.xml and core-site.xml.

Use BigInsights splitable compression to help save storage.

Partitioning

You can partition a Hive table by one or more columns. You specify the partitioning columns when you create the table. The data is stored in one directory per partition.

There is a performance advantage of using a partitioned table. Some queries that specify predicates on the partitioning columns can skip reading data from partitions that cannot hold qualifying rows. This approach is called *input pruning*.

Partition large tables to help improve performance through input pruning.

The BDW SOC_MEDIA_PST table is partitioned by month because queries on this table are expected to include predicates to filter data for a particular month.

Partitioning is achieved by specifying the `PARTITIONED BY` clause in the table definition, using InfoSphere Data Architect. For example, in the case of the SOC_MEDIA_PST table, the clause is specified as follows:

```
PARTITIONED BY (YEAR smallint, MONTH smallint)
```

Although the partitioning columns are defined in the physical data model, neither the table column block in the resulting DDL nor the associated data files contain values for the partitioning columns, since they are derived from the directory names.

You can add partitions later to the table in several ways:

- By using the `ALTER TABLE ADD PARTITION` statement, as shown in the following example:

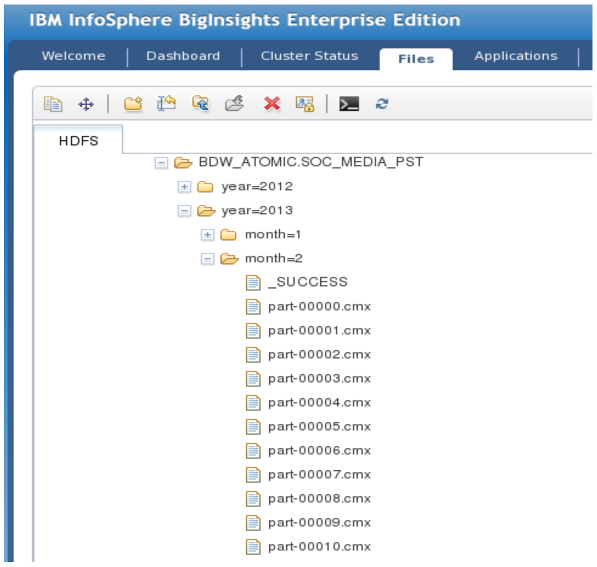
```
ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PST
ADD PARTITION (YEAR=2013, MONTH=2);
```

- By using the PARTITION clause of the `LOAD HIVE DATA` command.
- By using the LOAD USING JDBC command, which can create partitions dynamically. Also, the partitioning columns can be derived in the SELECT statement that is used in this LOAD command.

When designing partitioning, it is important to consider the impact on applications such as BigSheets. Choose partitioning columns in a nested fashion so that applications can navigate through the data hierarchy at different levels of granularity. For example, when partitioning daily by date, use partitioning columns comprising year, month, and day so that directories are nested day within month and month within year. BigSheets can then point at a particular month directory to pick up an entire month's worth of data or point at a particular year directory to pick up an entire year's worth of data.

Although you cannot directly update Hive tables at the file level, you can overwrite entire partitions. Therefore, if updates to a Hive table are necessary by merging in the changes and overwriting, consider the granularity of the updates in the selection of the partitioning columns. Also, ensure that the choice of partition columns results in a partition file size that is at least the same as the HDFS block size, which is 64 MB by default. This approach avoids a proliferation of small files that can impact performance and put pressure on the name node, which keeps all metadata for the file system in memory.

In the BDW example, the SOC_MEDIA_PST table is partitioned at the granularity of month, using the month and year columns that are derived from the Post Date (PST_DT) column as is shown in Figure 5.



```
BDW_ATOMIC.SOC_MEDIA_PST
  year=2012
  year=2013
    month=1
    month=2
      _SUCCESS
      part-00000.cmx
      part-00001.cmx
      part-00002.cmx
      part-00003.cmx
      part-00004.cmx
      part-00005.cmx
      part-00006.cmx
      part-00007.cmx
      part-00008.cmx
      part-00009.cmx
      part-00010.cmx
```

Figure 5. HDFS nested directories for a partitioned table

The refined physical data model is shown in figure 6 below.

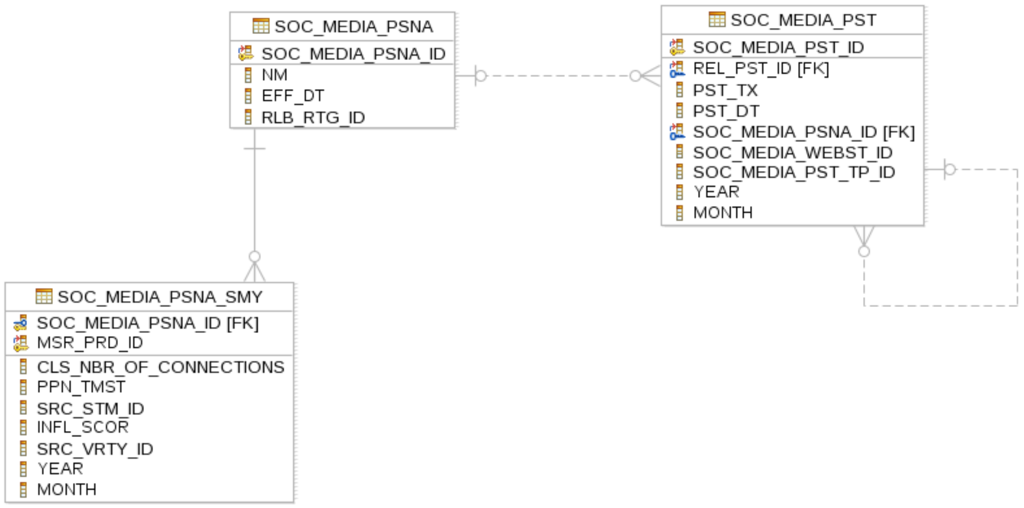


Figure 6. The refined physical data model

Generating the DDL statements to create the schema from the physical data model

The next step is to generate the DDL statements using InfoSphere Data Architect to create the schema for deployment to BigInsights. The procedure is as follows:

1. From the physical data model, select a blank area in the diagram.
2. From the InfoSphere Data Architect main menu, select **Data > Generate DDL.**
3. When prompted, provide details to complete the transformation.
   a. Accept the default values presented and click next until you get to the **Preview DDL** step.
   b. Select the **Run DDL on server** check-box.
   c. Provide the JDBC connection details.
   d. Click **Finish.**
4. Review any error messages, and confirm that all objects were created.

Assuming that no errors occurred, the BDW AWM is now deployed successfully to BigInsights and is ready for use. Finally, the BigInsights repository can now be populated with data to determine if any further tuning is required to the schema.

Conclusion

You can successfully deploy the BDW to InfoSphere BigInsights by following the recommendations in this paper.

Best practices

- Refine your scope as much as possible to address your current data and reporting needs.
- To help accommodate future upgrades, apply architecture and design changes that are specific to BigInsights to the physical data model rather than the logical data model.
- Merge separate DATE and TIME attributes into a single TIMESTAMP column in the physical data model.
- Consider denormalizing for improved query performance. Do not denormalize unless you need to.
- Use Hive tables rather than HBASE tables for data warehouse style queries and convenient sharing with applications such as BigSheets.
- Use external tables for convenient assertion of schemas and flexibility to define multiple views over the same data.
- Use the default TEXTFILE file format for convenient sharing with applications such as BigSheets.
- Use BigInsights splitable compression to help save storage.
- Partition large tables to help improve performance through input pruning.

Appendix A. Test environment

The test environment that was used in the research and development of this paper was a cluster comprising five IBM Power7 machines, each with approximately 400 GB of disk storage. InfoSphere BigInsights 2.1 was installed. The BDW AWM was deployed according to the procedure in this paper, and the cluster was populated with approximately 500 GB (one billion rows) of fictitious data.

Data Architect 9.1.1 was used.

The BDW that was used was BFMDW86.

Appendix B. DDL statements

The DDL statements for the BDW Social Media example are as follows:

```
CREATE EXTERNAL TABLE BDW_ATOMIC.SOC_MEDIA_PSNA (
          SOC_MEDIA_PSNA_ID INT NOT NULL,
          NM VARCHAR(32),
          EFF_DT TIMESTAMP,
          RLB_RTG_ID SMALLINT
    )
    ROW FORMAT DELIMITED
          FIELDS TERMINATED BY '\t'
    LOCATION '/banking/BDW/AWM/BDW_ATOMIC.SOC_MEDIA_PSNA';

CREATE EXTERNAL TABLE BDW_ATOMIC.SOC_MEDIA_PSNA_SMY (
          CLS_NBR_OF_CONNECTIONS SMALLINT,
          SOC_MEDIA_PSNA_ID INT NOT NULL,
          MSR_PRD_ID SMALLINT NOT NULL,
          PPN_TMST TIMESTAMP,
          SRC_STM_ID SMALLINT,
          INFL_SCOR SMALLINT,
          SRC_VRTY_ID SMALLINT
    )
    PARTITIONED BY (
          YEAR SMALLINT,
          MONTH SMALLINT
    )
    ROW FORMAT DELIMITED
          FIELDS TERMINATED BY '\t'
    LOCATION '/banking/BDW/AWM/BDW_ATOMIC.SOC_MEDIA_PSNA_SMY';

CREATE EXTERNAL TABLE BDW_ATOMIC.SOC_MEDIA_PST (
          REL_PST_ID INT,
          SOC_MEDIA_PST_ID INT NOT NULL,
          PST_TX VARCHAR(2000),
          PST_DT TIMESTAMP,
          SOC_MEDIA_PSNA_ID INT,
          SOC_MEDIA_WEBST_ID INT,
          SOC_MEDIA_PST_TP_ID SMALLINT
    )
    PARTITIONED BY (
          YEAR SMALLINT,
          MONTH SMALLINT
    )
    ROW FORMAT DELIMITED
          FIELDS TERMINATED BY '\t'
    LOCATION '/banking/BDW/AWM/BDW_ATOMIC.SOC_MEDIA_PST';

ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PSNA_SMY ADD PARTITION
(YEAR=2012, MONTH=11);
ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PSNA_SMY ADD PARTITION
(YEAR=2012, MONTH=12);
ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PSNA_SMY ADD PARTITION
(YEAR=2013, MONTH=1);
```

```
ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PSNA_SMY ADD PARTITION
(YEAR=2013, MONTH=2);
ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PSNA_SMY ADD PARTITION
(YEAR=2013, MONTH=3);

ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PST ADD PARTITION (YEAR=2012,
MONTH=11);
ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PST ADD PARTITION (YEAR=2012,
MONTH=12);
ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PST ADD PARTITION (YEAR=2013,
MONTH=1);
ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PST ADD PARTITION (YEAR=2013,
MONTH=2);
ALTER TABLE BDW_ATOMIC.SOC_MEDIA_PST ADD PARTITION (YEAR=2013,
MONTH=3);
```

Further reading

- "Governing and Managing Enterprise Models: Series" (http://www.ibm.com/developerworks/rational/library/10/governingandmanagingenterprisemodels-series/index.html)
- "Scoping the IBM Industry Model for banking using Enterprise Model Extender and InfoSphere Data Architect" (http://www.ibm.com/developerworks/data/tutorials/dm-1003bankindustrymodel)
- "Understanding InfoSphere BigInsights" (http://www.ibm.com/developerworks/data/library/techarticle/dm-1110biginsightsintro/index.html)

Contributors

Gary Thompson
Information Architect, Industry Models Architecture

Pat G. O'Sullivan
Senior Technical Staff Member, Industry Models Architecture

Enda McCallig
DB2 Data Warehouse Specialist

Bryan Tierney
Business Analyst, Industry Models

Notices

Trademarks