



# i2b2 Clinical Research Chart (CRC) Design Document

*Document Version:* 1.0

*I2b2 Software Release:* 1.3

# Table of Contents

<b>1.</b>	<b>INTRODUCTION</b> .....	<b>3</b>
<b>2.</b>	<b>I2B2 DATA MART</b> .....	<b>3</b>
<b>3.</b>	<b>I2B2 DATA MART TABLES</b> .....	<b>4</b>
3.1	GENERAL INFORMATION .....	4
3.2	OBSERVATION_FACT .....	4
3.3	PATIENT_DIMENSION .....	6
3.4	VISIT_DIMENSION .....	7
3.5	CONCEPT_DIMENSION .....	8
3.6	PROVIDER_DIMENSION.....	9
3.7	CODE_LOOKUP .....	9
3.8	PATIENT_MAPPING.....	10
3.9	ENCOUNTER_MAPPING.....	11
3.10	JOINING COLUMNS.....	11
<b>4.</b>	<b>PATIENT DATA OBJECT</b> .....	<b>12</b>
<b>5.</b>	<b>PATIENT AND EVENT MAPPING SCENARIOS</b> .....	<b>14</b>
5.1	SELF MAPPING .....	15
5.2	NEW MAPPINGS – ADDING NEW VALUES.....	16
5.2.1	Case 1: (<pid> not found, generate [max+1]).....	16
5.2.2	Case 2: (<patient> not found, generate [max + 1]) .....	17
5.2.3	Case 3: (HIVE id (patient_num) not found).....	17
5.3	.....	18
5.4	HANDLING EXISTING VALUES .....	18
5.4.1	Case 1: (HIVE id found, but <patient_map_id> not mapped).....	18
	Case 2: (<patient_id> <> patient_num, and <patient_map_id> not mapped) .....	18
5.4.2	Case 3 : (patient_num already in mapping table, but with a different date).....	19
5.4.3	Case 4: (<patient> without HIVE number).....	19
5.5	INVALID XML.....	20
5.5.1	Case 1: (<pid> without patient_id - INVALID ).....	20
<b>6.</b>	<b>OBSERVATION FACT SCENARIOS</b> .....	<b>20</b>
6.1	CASE 1 EXAMPLE.....	21
6.2	CASE 2 EXAMPLE.....	22
<b>7.</b>	<b>DEFINITIONS OF TERMS</b> .....	<b>23</b>

## 1. INTRODUCTION

The Data Repository Cell (also called the Clinical Research Chart, or CRC), is designed to hold data from clinical trials, medical record systems and laboratory systems, along with many other types of clinical data from heterogeneous sources. The CRC stores this data in three tables, the patient, visit and observation tables. In addition to these three tables, there are three lookup tables, the concept, provider and code tables, and two mapping tables, patient\_mapping and visit\_mapping.

The three data tables, along with two of the lookup tables (concept and provider) make up the star schema of the warehouse. The code table is strictly a lookup table and is not part of the star schema. All of the tables that are part of the CRC are described in this document.

## 2. I2B2 DATA MART

The i2b2 data mart is a data warehouse modeled on the star schema structure first proposed by Ralph Kimball. The database schema looks like a star, with one central fact table surrounded radially by one or more dimension tables. The most important concept regarding the construction of a star schema is identifying what constitutes a fact.

In healthcare, a logical fact is an observation on a patient. It is important to note that an observation may not represent the onset or date of the condition or event being described, but instead is simply a recording or a notation of something. For example, the observation of 'diabetes' recorded in the database as a 'fact' at a particular time does not mean that the condition of diabetes began exactly at that time, only that a diagnosis was recorded at that time (there may be many diagnoses of diabetes for this patient over time).

The fact table contains the basic attributes about the observation, such as the patient and provider numbers, a concept code for the concept observed, a start and end date, and other parameters described below in this document. In i2b2, the fact table is called observation\_fact..

Dimension tables contain further descriptive and analytical information about attributes in the fact table. A dimension table may contain information about how certain data is organized, such as a hierarchy that can be used to categorize or summarize the data. In the i2b2 Data Mart, there are four dimension tables that provide additional information about fields in the fact table: patient\_dimension, concept\_dimension, visit\_dimension, and provider\_dimension.

### 3. I2B2 DATA MART TABLES

#### 3.1 GENERAL INFORMATION

The observation table has only required columns. The patient\_dimension and visit\_dimension tables have both required and optional columns. All the tables have five technically-oriented, or administrative, columns:

Column Name	Data Type	Nullable	Definition
update_date	datetime	Yes	Date the row was updated by the source system (date is obtained from the source system)
download_date	datetime	Yes	Date the data was downloaded from the source system.
import_date	datetime	Yes	Date data was imported into the CRC
sourcesystem_cd	varchar(50)	Yes	Coded value for the data source system
upload_id	decimal(38,0)	Yes	A numeric id given to the upload

#### 3.2 OBSERVATION\_FACT

The observation\_fact table is the fact table of the i2b2 star schema and represents the intersection of the dimension tables. Each row describes one observation about a patient made during a visit. Most queries in the i2b2 database require joining the observation\_fact table with one or more dimension tables together.

observation_fact		
PK	<u>encounter_num</u>	int
PK	<u>concept_cd</u>	varchar(50)
PK	<u>provider_id</u>	varchar(50)
PK	<u>start_date</u>	datetime
PK	<u>modifier_cd</u>	varchar(50)
	patient_num	int
	valType_cd	varchar(50)
	tval_char	varchar(255)
	nval_num	decimal(18,5)
	valueFlag_cd	varchar(50)
	quantity_num	decimal(18,5)
	units_cd	varchar(50)

observation_fact		
	end_date	datetime
	location_cd	varchar(50)
	observation_blob	text
	confidence_num	decimal(18,5)
	update_date	datetime
	download_date	datetime
	import_date	datetime
	sourcesystem_cd	varchar(50)
	upload_id	int

Observation_Fact			
Key	Column Name	Column Definition	Nullable? (Default =YES)
<b>PK</b>	<b>encounter_num</b>	Encoded i2b2 patient visit number.	NO
	<b>patient_num</b>	Encoded i2b2 patient number.	NO
<b>PK</b>	<b>concept_cd</b>	ID number for observation of interest (i.e. diagnoses, procedures, medications, lab test)	
<b>PK</b>	<b>provider_id</b>	Practitioner id or provider id.	
<b>PK</b>	<b>start_date</b>	Starting date-time of observation <i>(mm/dd/yyyy)</i>	
<b>PK</b>	<b>modifier_cd</b>	Ranking of Modifiers  1, 2, 3, ... 1.1, 1.2. 1.3... 1.1.1, 1.1.2. 1.1.3...	
	valType_cd	Format of the concept.  <i>N = Numeric</i> <i>T = Text</i>	
	tval_char	The operator used when valtype_cd = "N"  <i>E = Equals</i> <i>L = Less Than</i> <i>G = Greater Than</i>	
	nval_num	Stores a numerical value	
	valueFlag_cd	Used to flag certain outlying or abnormal values  <i>H = High</i> <i>L=Low</i> <i>A = Abnormal</i>	

	quantity_num	Quantity of nval	
	units_cd	Units of measurement or nval	
	end_date	The ending date-time for the observation	
	location_cd	A location code, such as for a clinic	
	confidence_num	Assessment of accuracy of data	
	observation_blob	Holds any extra data that exists, often encrypted PHI	
	update_date	(as above)	
	download_date	(as above)	
	import_date	(as above)	
	sourcesystem_cd	(as above)	
	upload_id	(as above)	

### 3.3 PATIENT\_DIMENSION

Each record in the Patient\_Dimension table represents a patient in the database. The table includes demographics fields such as gender, age, race, etc. Most attributes of the patient dimension table are discrete (i.e. Male/Female, Zip code, etc.).

Every Patient\_Dimension table has four required columns, Patient\_Num, Birth\_Date, Death\_Date, and Vital\_Status\_Cd. The patient\_num column must be filled (it must not be null). The patient\_num column is the primary key for the table and therefore can not contain duplicates. This column holds a reference number for the patient within the data repository. It is an integer. The birth\_date and death\_date columns can be null, and are date-time fields. They contain the birth and death dates for the patient if they exist. They are not standardized to a specific time zone, a limitation that may need to be addressed in the future. The vital\_status\_cd column contains a code that represents the vital status of the patient, and the precision of the vital status data. The codes for this field were determined arbitrarily, as there was no standardized coding system for their representation. The values for the vital\_status\_cd column are N for living (corresponds to a null death\_date) and Y for deceased (death\_date accurate to day), M for deceased (death\_date accurate to month), and X for deceased (death\_date accurate to year).

The Patient table may have an unlimited number of optional columns and their data types and coding systems are local implementation-specific. An example of a Patient\_Dimension table is shown below. In the example table, there are eight optional columns. The rules for using the codes in the columns to perform queries are represented in the metadata. For example, the columns shown below include a race\_cd column and a statecityzip\_cd column. Codes from the race\_cd column are enumerated values that may be grouped together to achieve a desired result, such as if there are 4 codes for the “white” race such as W, WHITE, WHT, and WHITE-HISPANIC, they can be counted directly to determine the number of white-race

patients in the database. Codes from the statecityzip\_cd are strings that represent hierarchical information. In this way, the string is queried from left to right in a string comparison to determine which patients are returned by the query, for example, if a code is MA\BOSTON\02114 and all the patients in BOSTON are desired, the string "MA\BOSTON\\*" (where \* is a wildcard) would be queried.

patient_dimension		
PK	<u>patient_num</u>	int
	vital_status_cd	varchar(50)
	birth_date	datetime
	death_date	datetime
	sex_cd*	varchar(50)
	age_in_years_num*	int
	language_cd*	varchar(50)
	race_cd *	varchar(50)
	marital_Status_cd *	varchar(50)
	religion_cd *	varchar(50)
	zip_cd *	varchar(10)
	stateCityZip_Path	varchar(700)
	patient_blob	text
	update_date	datetime
	download_date	datetime
	import_date	datetime
	sourcesystem_cd	varchar(50)
	upload_id	int

### 3.4 VISIT\_DIMENSION

The Visit\_Dimension table represents sessions where observations were made. Each row represents one session (also called a visit, event or encounter.) This session can involve a patient directly, such as a visit to a doctor's office, or it can involve the patient indirectly, as in when several tests are run on a tube of the patient's blood. More than one observation can be made during a visit. All visits must have a start time associated with them, but they may or may not have an end date. The visit record also contains specifics about the location of the session, such as at which hospital or clinic the session occurred, and whether the patient was an inpatient or outpatient at the time of the visit.

The Visit\_Dimension table has four required columns Patient\_Num, Start\_Date, End\_Date, and Active\_status\_cd. The visit\_num column is the primary key for the table and therefore can not contain duplicates. This column holds a reference number for the visit within the data repository. It is an integer. The start\_date and end\_date columns can be null, and are date-time fields. Because a visit is considered to be an

event, there is a distinct beginning and ending date and time for the event. However, these dates may not be recorded and the active\_status\_cd is used to record whether the event is still ongoing, along with the precision of the available dates. Conceptually, this makes it very similar to the vital\_status\_cd column in the patient table. The codes for this field were determined arbitrarily, as there was no standardized coding system for their representation. The codes are 'F' for final, 'P' for preliminary, 'A' for active (indicating there is no end\_date) and null if there are no dates.

The Visit\_Dimension table may have an unlimited number of optional columns, but their data types and coding systems are local implementation specific. An example of a Visit table is shown below. In the example table, there are four optional columns. The rules for using the codes in the columns to perform queries are represented in the metadata, and the values within the columns follow a similar pattern as described above for the Patient\_Dimension table.

visit_dimension		
PK	<u>encounter_num</u>	int
	<b>patient_num</b>	<b>int</b>
	active_status_cd	varchar(50)
	start_date	datetime
	end_date	datetime
	inout_cd*	varchar(50)
	location_cd*	varchar(50)
	visit_blob	text
	update_date	datetime
	download_date	datetime
	import_date	datetime
	sourcesystem_cd	varchar(50)
	upload_id	int

### 3.5 CONCEPT\_DIMENSION

The concept\_dimension table contains one row for each concept. Possible concept types are diagnoses, procedures, medications and lab tests. The structure of the table gives enough flexibility to store virtually any concept type, such as demographics and genetics data.

The concept\_path is a path that delineates the concept's hierarchy. The concept\_code is the code that represents the diagnosis, procedure, or any other coded value. Name\_char is the actual name of the concept.

concept_dimension		
<b>PK</b>	<b><u>concept_path</u></b>	<b>varchar(700)</b>
	<b>concept_cd</b>	<b>varchar(50)</b>
	name_char	varchar(2000)
	concept_blob	text
	update_date	datetime
	download_date	datetime
	import_date	datetime
	sourcesystem_cd	varchar(50)
	upload_id	int

### 3.6 PROVIDER\_DIMENSION

Each record in the Provider\_Dimension table represents a doctor or provider at an institution. The provider\_path is the path that describes the how the provider fits into the institutional hierarchy. Institution, department, provider name and a code may be included in the path.

provider_dimension		
<b>PK</b>	<b><u>provider_id</u></b>	<b>varchar(50)</b>
<b>PK</b>	<b><u>provider_path</u></b>	<b>varchar(700)</b>
	name_char	varchar(850)
	provider_blob	text
	update_date	datetime
	download_date	datetime
	import_date	datetime
	sourcesystem_cd	varchar(50)
	upload_id	int

### 3.7 CODE\_LOOKUP

The code\_looup table contains coded values for different fields in the CRC. For example, in the visit\_dimension table, there is the location\_cd field that may have different values for different types hospital locations and these values would be stored in the code lookup table. The first four fields of the table might look like this:

	table_cd	column_cd	code_...	name_char
1	VISIT_DIMENSION	LOCATION_CD	@	zz not recorded
2	VISIT_DIMENSION	LOCATION_CD	BWH	Brigham and Womens Hospital
3	VISIT_DIMENSION	LOCATION_CD	FH	Faulkner Hospital
4	VISIT_DIMENSION	LOCATION_CD	MGH	Massachusetts General Hospital
5	VISIT_DIMENSION	LOCATION_CD	NWH	Newton Wellesley Hospital
6	VISIT_DIMENSION	LOCATION_CD	SRH	Spaulding Rehabilitation Hospital

code_lookup		
<b>PK</b>	<b><u>table_cd</u></b>	<b>varchar(100)</b>
<b>PK</b>	<b><u>column_cd</u></b>	<b>varchar(100)</b>
<b>PK</b>	<b><u>code_cd</u></b>	<b>varchar(50)</b>
	name_char	varchar(650)
	lookup_blob	text
	update_date	datetime
	download_date	datetime
	import_date	datetime
	sourcesystem_cd	varchar(50)
	upload_id	int

### 3.8 PATIENT\_MAPPING

The patient\_mapping table maps the i2b2 patient\_num to an encrypted number from the source\_system,patient\_ide (the 'e' in ide is for 'encrypted'.) Patient\_ide\_source contains the name of the source system. Patient\_ide\_status gives the status of the patient number in the source system, for example, if it is Active or Inactive or Deleted or Merged.

patient_mapping		
<b>PK</b>	<b><u>patient_ide</u></b>	<b>varchar(200)</b>
<b>PK</b>	<b><u>patient_ide_source</u></b>	<b>varchar(50)</b>
	<b>patient_num</b>	<b>int</b>
	patient_ide_status	varchar(50)
	update_date	datetime
	download_date	datetime
	import_date	datetime
	sourcesystem_cd	varchar(50)

patient_mapping		
	upload_id	int

### 3.9 ENCOUNTER\_MAPPING

The encounter\_mapping table maps the i2b2 encounter\_number to an encrypted number from the source system, encounter\_id\_source (the 'e' in ide is for 'encrypted'.) Encounter\_id\_source contains the name of the source system. Encounter\_id\_status gives the status of the encounter in the source system, for example, if it is Active, Inactive, Deleted or Merged.

patient_mapping		
<b>PK</b>	<b><u>encounter_id</u></b>	<b>varchar(200)</b>
<b>PK</b>	<b><u>encounter_id_source</u></b>	<b>varchar(50)</b>
	<b>encounter_num</b>	<b>int</b>
	patient_id	varchar(200)
	patient_id_source	varchar(50)
	encounter_id_status	varchar(50)
	upload_date	datetime
	download_date	datetime
	import_date	datetime
	sourcesystem_cd	varchar(50)
	upload_id	int

### 3.10 JOINING COLUMNS

All of the tables above can be linked together using SQL joins to obtain more data. For example, a concept will have a code in the observation\_fact concept\_cd field, but will have to be joined to the concept\_dimension concept\_cd field to find the name\_char and/or concept\_path that define the concept. Below are some examples of common columns used to join tables in the star schema.

#### Observation\_Fact

Encounter\_num in Observation\_Fact can be joined to encounter\_num in the Visit\_Dimension table.

Patient\_num in Observation\_Fact can be joined to patient\_num in the Patient\_Dimension and Visit\_Dimension tables.

Provider\_id in Observation\_Fact can be joined to provider\_id in the Provider\_Dimension table.

#### Patient\_Dimension

Patient\_num in Patient\_Dimension can be joined to patient\_num in the Observation\_Fact and Visit\_Dimension tables.

#### Visit\_Dimension

Encounter\_num in Visit\_Dimension can be joined to encounter\_num in the Observation\_Fact table.

Patient\_num in Visit\_Dimension can be joined to patient\_num in the Observation\_Fact and Visit\_Dimension tables.

#### Concept\_Dimension

Concept\_cd in Concept\_Dimension can be joined to concept\_cd in the Observation\_Fact table.

#### Provider\_Dimension

Provider\_id in Provider\_Dimension can be joined to provider\_id in the Observation\_Fact table.

## 4. PATIENT DATA OBJECT

The Patient Data Object (PDO) is the XML representation of patient data. This data corresponds to the values in the star schema tables in the database. Below is a sample PDO. Definitions of the fields can be found in the last section of this document.

```
<repository:patient_data xmlns:repository="">
  <event_set>
    <event *>
      <event_id source="hive">1256</event_id>
      <patient_id source="hive">4</patient_id>
      <start_date>1999-02-28T13:59:00</start_date>
      <end_date>1999-02-28T13:59:00</end_date>
      <active_status_cd>F</active_status_cd>
      <param name="admission status">Inpatient</param>
      <param name="site">MGH</param>
    </event *>
  </event_set>
</repository:patient_data>
```

```

        <param name="location">Oral Surgery</param>
        <event_blob/>
    </event>
</event_set>
<concept_set>
    <concept *>
        <concept_path>Diagnoses\athm\C0004096</concept_path>
        <concept_cd>UMLS:C0004096</concept_cd>
        <name_char>Asthma</name_char>
        <concept_blob/>
    </concept>
</concept_set>
<observer_set>
    <observer *>
        <observer_path>MGH\Medicine\C0004096</observer_path>
        <observer_cd>M00022303</observer_cd>
        <name_char>Shawn Murphy MD</name_char>
        <observer_blob/>
    </observer>
</observer_set>
<pid_set>
    <pid>
        <patient_id source="hive">4</patient_id>
        <patient_map_id source="MGH" status="A" *>0051382</patient_map_id>
        <patient_map_id source="EMPI" status="A" *>10034586</patient_map_id >
    </pid>
</pid_set>
<eid_set>
    <eid>
        <event_id source="hive">1256</event_id>
        <event_map_id source="MGHTSI" status="A"
            patient_id="0051382" patient_id_source="MGH" *>KST004</event_map_id>
    </eid>
</eid_set>
<patient_set>
    <patient *>
        <patient_id source='hive">4</patient_id>
        <birth_date>1930-02-28</birth_date>
        <death_date>2001-02-28</death_date>
        <vital_status_cd>Y<vital_status_cd>
        <param name="gender">Female</param>
        <param name="age in years">71</param>
        <param name="language">English</param>
        <param name="race">Black</param>
        <param name="marrital status">Married</param>
        <param name="zipcode">12345-1234</ param>
        <patient_blob/>
    </patient>
</patient_set>
<observation_set path="">
    <observation *>

```

```

    <event_id source="hive">1256</event_id>
    <patient_id source="hive">4</patient_id>
    <concept_cd name="Asthma">UMLS:C0004096</concept_cd>
    <observer_cd name="Doctor, John A., MD">B001234567</observer_cd>
    <start_date>1999-02-28T13:59:00</start_date>
    <modifier_cd>1.1</modifier_cd>
    <valtype_cd>N</valtype_cd>
    <tval_char>E</tval_char>
    <nval_num units="ml">1.0</nval_num>
    <valueflag_cd name="High">H</valueflag_cd>
    <quantity_num>1.0</quantity_num>
    <units_cd>ml</units_cd>
    <end_date>1999-02-28T13:59:00</end_date>
    <location_cd name="Oral Surgery">MT045</location_cd>
    <confidence_num></confidence_num>
    <observation_blob/>
  </observation>
</observation_set>
<code_set>
  <code *>
    <table_cd>observation_fact</table_cd>
    <column_cd>ValueType_CD</dimension_path>
    <code_cd>N</dimension_cd>
    <name_char>Numeric</name_char>
    <code_blob/>
  </code>
</code_set>
</repository:patient_data>

```

\* indicates the following technical metadata parameters may be included in the tag (shown here with sample data values):

```

update_date="1999-02-28T13:59:00"
download_date="1999-02-28T13:59:00"
import_date="1999-02-28T13:59:00"
sourcesystem_cd="RPDRASTHMA"

```

## 5. PATIENT AND EVENT MAPPING SCENARIOS

A patient may have more than one identifier in different source systems and will be given a unique i2b2 identifier. All of these identifiers are grouped together in the XML Patient Data Object (PDO) in the <pid\_set> and are also added to the patient\_mapping table in the database. A similar process occurs for encounters from different systems grouped together in the <eid\_set> in the PDO and in the encounter\_mapping table in the database.

The patient and event mapping tables link the values used in the i2b2 database to their counterparts in the source systems from which the numbers came. The patient\_mapping and event\_mapping tables are populated by existing hive numbers when the database is created and are also added to as new patients and encounters are added. Each patient number corresponds to a row in the patient table and each encounter or event has a row in the encounter\_mapping table. The following examples review different scenarios for adding data to the mapping tables. (The examples refer to the patient\_mapping table, but can be applied to the encounter\_mapping table in the same way, i.e. patient\_num is to patient\_ide as encounter\_num is to encounter\_ide).

Encrypted identifiers are indicated by appending ‘\_e’ to the name of the source system. So, for example, if the identifier is an encrypted number from Massachusetts General Hospital, the source will be ‘MGH\_e’. The scenarios below refer both to the XML objects in the PDO and to the dimension tables and mapping tables in the database. Patient\_num is the field name for the i2b2 identifier in the database and corresponds to the value of <patient\_id> when the source is ‘HIVE’.

Below is a generic <pid\_set> from the XML Patient Data Object (PDO).

```
<pid_set>
  <pid>
    <patient_id source= source >value</patient_id>
    <patient_map_id source="source status="A >value</patient_map_id>
    <patient_map_id source= source status="A >value</patient_map_id >
    &
  </pid>
</pid_set>
```

The following cases describe possible scenarios for different combinations of <patient\_id> source and value and <patient\_id\_map> source and value for both the <pid> and the <patient> objects. An id source and its value are both needed to determine the parameters inserted into the mapping tables. These two fields are called the source/value pair. The patient\_id in the <pid> must have the same source/value pair as in the <patient> object and the rest of the PDO. There may be multiple <patient\_map\_ids> in one <pid>, with each one representing a different source system and identifier value for the same patient.

The mapping process requires checking to see if the source/value pairs for <patient\_id> and <patient\_map\_id> already exist in the i2b2 hive and then following the appropriate scenario below. The dates associated with the object must also be checked in order to determine the most recent values.

## 5.1 SELF MAPPING

Self-mapping occurs when the <patient\_id> source is HIVE and the <patient\_id> value already exists in the hive. All hive patient and encounter numbers are mapped to themselves and inserted into their respective tables (either patient\_mapping or

encounter\_mapping). The default mapping status is 'A' for ACTIVE and the source value is 'HIVE'.

Example:

```
<pid_set>
  <pid>
    <patient_id source= HIVE >1</patient_id>
  </pid>
</pid_set>
```

Row in patient\_mapping table:

patient_id	patient_id_source	patient_num	patient_id_status
1	HIVE	1	A

## 5.2 NEW MAPPINGS – ADDING NEW VALUES

The following cases address three different situations where there is a number that does not already exist in the i2b2 hive. Note that in these cases, the new number must be added to the patient\_dimension table as well as to the patient\_mapping table in the database.

### 5.2.1 Case 1: (<pid> not found, generate [max+1])

If the <patient\_id> source/value pair has not been added to the mapping table, a new patient\_num with value max(patient\_num)+1 should be generated and all the patient\_nums for this patient will get this value. The new patient number must also be added to the patient\_dimension table.

Example:

```
New <patient_id> source/value pair = 'EMPI'/1000000
Select max(patient_num) from patient_mapping = 527
New patient_num = max(patient_num) +1 = 528
```

```
<pid>
  <patient_id source="EMPI">1000000</patient_id>
  <patient_map_id source="MGH">123</patient_map_id>
  <patient_map_id source="BWH">777</patient_map_id>
</pid>
```

Rows in patient\_mapping table:

patient_id	patient_id_source	patient_num	patient_id_status
1000000	EMPI	528	A
123	MGH	528	A
777	BWH	528	A
528	HIVE	528	A

### 5.2.2 Case 2: (<patient> not found, generate [max + 1])

If the <patient\_id> source in the <patient> object is not 'HIVE' and the patient\_id source ('MGH') and value ('xyz') combination do not exist, a new patient\_num with value max(patient\_num)+1 should be generated all the patient\_nums for this patient will get this value. The new patient number must also be added to the patient\_dimension table.

Example:

```
<patient>
  <patient_id source="MGH">xyz</patient_id>
  <param name="zipcode">02149</param>
</patient>
```

Parameters:

New <patient\_id> source/value pair = 'MGH'/xyz  
 Select max(patient\_num) from patient\_mapping = 527  
 New patient\_num = max(patient\_num) + 1 = 528

Rows in patient\_mapping table:

patient_id	patient_id_source	patient_num	patient_id_status
xyz	MGH	528	A
528	HIVE	528	A

### 5.2.3 Case 3: (HIVE id (patient\_num) not found)

Here the <patient\_id> source is 'HIVE', but the value (1000000) does not exist in the mapping table. In this case, generating max+1 is not necessary, the value 1000000 can be added directly to the table, since it is not already in the table. This new patient number must also be added to the patient\_dimension table.

```
<pid>
  <patient_id source="HIVE">1000000</patient_id>
  <patient_map_id source="MGH ">123</patient_map_id>
  <patient_map_id source="BWH ">777</patient_map_id>
</pid>
```

Rows in patient\_mapping table:

patient_id	patient_id_source	patient_num	patient_id_status
1000000	HIVE	1000000	A
123	MGH	1000000	A
777	BWH	1000000	A

5.3

## 5.4 Handling Existing Values

The following cases address situations where the patient\_num has already been added to the mapping table.

### 5.4.1 Case 1: (HIVE id found, but <patient\_map\_id> not mapped)

In this case the patient\_num (1000000) has been added to the mapping table, but the <patient\_map\_id>s from BWH and MGH for the patient have not so the hive id (patient\_num) is applied to all of the <patient\_map\_id>s that are not currently mapped.

```
<pid>
  <patient_id source="HIVE">1000000</patient_id>
  <patient_map_id source="MGH">123</patient_map_id>
  <patient_map_id source="bwh">123</patient_map_id>
</pid>
```

Rows in patient\_mapping table before :

patient_id	patient_id_source	patient_num	patient_id_status
1000000	HIVE	1000000	A

Rows in patient\_mapping table after:

patient_id	patient_id_source	patient_num	patient_id_status
1000000	HIVE	1000000	A
123	MGH	1000000	A
777	BWH	1000000	A

### Case 2: (<patient\_id> <> patient\_num, and <patient\_map\_id> not mapped)

In this case, the <patient\_id> source and value ('EMPI'/100000) are already mapped to a patient\_num, but the <patient\_map\_id>s are not, so use that patient\_num for any of the <patient\_map\_id>s that are not already mapped.

```
<pid>
  <patient_id source="EMPI">100000</patient_id>
  <patient_map_id source="MGH">123</patient_map_id>
  <patient_map_id source="bwh">777</patient_map_id>
```

</pid>

Rows in patient\_mapping table before :

patient_id	patient_id_source	patient_num	patient_id_status
1000000	EMPI	528	A
528	HIVE	528	A

Rows in patient\_mapping table after:

patient_id	patient_id_source	patient_num	patient_id_status
1000000	EMPI	528	A
123	MGH	528	A
777	BWH	528	A
528	HIVE	528	A

#### 5.4.2 Case 3 : (patient\_num already in mapping table, but with a different date)

If the <patient\_id> value already exists in the mapping table, compare the update\_date with the current patient record's update date. If the new record has a more recent date, then update the current patient record with this data.

```
<patient update_date="2008-05-04 18:13:51.00">  
  <patient_id source="HIVE">100</patient_id>  
  <param name="zipcode">02149</param>  
</patient>
```

Row in patient\_mapping table before :

patient_id	patient_id_source	patient_num	patient_id_status	update_date
100	HIVE	100	A	2006-12-03 00:00:00

Row in patient\_mapping table after:

patient_id	patient_id_source	patient_num	patient_id_status	update_date
100	HIVE	100	A	2008-05-04 18:13:51.

#### 5.4.3 Case 4: (<patient> without HIVE number)

If the <patient\_id> source and value are already mapped to a patient\_num, then the update date should be compared to the existing record's update date. If the new record has a more recent date, then update the current patient record with this data.

```
<patient update_date="2006-05-04T18:13:51.0Z">
  <patient_id source="MGH">xyz</patient_id>
  <param name="zipcode">02149</param>
</patient>
```

## 5.5 Invalid XML

### 5.5.1 Case 1: (<pid> without patient\_id - INVALID )

This example is invalid, because it contains patient\_map\_ids without a patient\_id. Every <pid> must have a <patient\_id>. In this case the <patient\_id> should be added to the PDO.

```
<pid>
  <patient_map_id source="MGH">123</patient_map_id>
  <patient_map_id source="bwh">123</patient_map_id>
</pid>
```

## 6. OBSERVATION FACT SCENARIOS

The updates to the observation fact can be classified into two cases.

Case 1) Replace the old set of facts with the new set of facts for the matching encounter.

Case 2) Add new facts, irrespective of whether the fact s encounter exists or not.

The case (2) also involves overwriting any matching facts. i.e. if the incoming fact matches a particular stored fact and its update date greater than the matched fact's update date, then the new fact will overwrite the old fact.

## 6.1 Case 1 example

Row in observation\_fact table before :

Encounter_num	Patient_num	Concept_Cd	Nval_num	update_date
100	100	FC30.00620	10.9	2008-05-04 18:13:51
100	100	FC30.00621	20.2	2008-05-04 18:13:51
100	100	FC30.00622	6.0	2008-05-04 18:13:51

```

<observation update_date="2008-05-04T18:13:51.498-04:00" sourcesystem_cd="PFT">
  <event_id source="HIVE">100</event_id>
  <patient_id source="HIVE">100</patient_id>
  <concept_cd>LCS-I2B2:pulheight</concept_cd>
  <nval_num>6.0</nval_num>
</observation>
<observation update_date="2008-05-04T18:13:51.498-04:00" sourcesystem_cd="PFT">
  <event_id source="HIVE">100</event_id>
  <patient_id source="HIVE">100</patient_id>
  <concept_cd>LCS-I2B2:pulweight</concept_cd>
  <nval_num>100.9</nval_num>
</observation>
<observation update_date="2008-05-04T18:13:51.498-04:00" sourcesystem_cd="PFT">
  <event_id source="HIVE">100</event_id>
  <patient_id source="HIVE">100</patient_id>
  <concept_cd>LCS-I2B2:pulfev1pred</concept_cd>
  <nval_num>76</nval_num>
</observation>

```

Row in observation\_fact table after:

Encounter_num	Patient_num	Concept_Cd	Nval_num	update_date
100	100	LCSI2B2:pulweight	100.9	2008-05-04 18:13:51
100	100	LCSI2B2:pulheight	6.0	2008-05-04 18:13:51
100	100	LCSI2B2:pulfev1pred	76	2008-05-04 18:13:51

## 6.2 Case 2 example

Row in observation\_fact table before:

Encounter_num	Patient_num	Concept_Cd	Nval_num	update_date
100	100	FC30.00620	10.9	2008-05-04 18:13:51
100	100	FC30.00621	20.2	2008-05-04 18:13:51
100	100	FC30.00622	6.0	2008-05-04 18:13:51

```

<observation update_date="2008-05-04T18:13:51.498-04:00" sourcesystem_cd="PFT">
  <event_id source="HIVE">100</event_id>
  <patient_id source="HIVE">100</patient_id>
  <concept_cd>LCS-I2B2:pulheight</concept_cd>
  <nval_num>6.0</nval_num>
</observation>
<observation update_date="2008-05-04T18:13:51.498-04:00" sourcesystem_cd="PFT">
  <event_id source="HIVE">100</event_id>
  <patient_id source="HIVE">100</patient_id>
  <concept_cd>LCS-I2B2:pulweight</concept_cd>
  <nval_num>100.9</nval_num>
</observation>
<observation update_date="2008-10-04T18:13:51.498-04:00" sourcesystem_cd="FC">
  <event_id source="HIVE">100</event_id>
  <patient_id source="HIVE">100</patient_id>
  <concept_cd>FC30.00622</concept_cd>
  <nval_num>76.0</nval_num>
</observation>
  
```

Row in observation\_fact table after:

Encounter_num	Patient_num	Concept_Cd	Nval_num	update_date
100	100	FC30.00620	10.9	2008-05-04 18:13:51
100	100	FC30.00621	20.2	2008-05-04 18:13:51
100	100	FC30.00622	76.0	2008-10-04 18:13:51
100	100	LCSI2B2:pulweight	100.9	2008-05-04 18:13:51
100	100	LCSI2B2:pulheight	6.0	2008-05-04 18:13:51

**Assumption:** the record(s) in the update file (new record) has the same primary key as a record(s) in the associated table (existing record).

Primary Key includes:  
 Encounter number  
 Patient number  
 Concept code  
 Start date  
 Modifier code  
 Observer code

Following conditions will result in the new record **replacing** the existing record:

new record update date	equal to (=)		update date on the existing record	
new record update date	greater than (>)		update date on the existing record	
new record update date	is not null	AND	update date on the existing record	null
new record update date	null	AND	update date on the existing record	null

Following conditions will result **ignoring** the new record and **not** updating the existing record:

new record update date	less than (<)		update date on the existing record	
new record update date	null	AND	update date on the existing record	is not null

## 7. DEFINITIONS OF TERMS

**patient\_data:** The root element that holds data from the patient data tables. May contain any number of observation\_set's, and none or one patient\_set, event\_set, concept\_set, observer\_set, code\_set, pid\_set, or eid\_set. They can occur in any order.

**event\_set:** data from the visit\_dimension table

**event:** One row of data from the visit\_dimension table.

**event\_id:** A choice between Encounter\_Num (if source is HIVE) or Encounter\_Id if another source. A source with “\_e” at the end is encrypted.  
**patient\_id:** A choice between Patient\_Num, (if source is HIVE) or Patient\_Id if another source. A source with “\_e” at the end is encrypted.  
**start\_date:** The date-time that the event started.  
**end\_date:** The date-time that the event ended.  
**active\_status\_cd:** A code to represent the meaning of the date fields above.  
**param:**  
**event\_blob:** XML data that includes partially structured and unstructured data about a visit.

**concept\_set:** data from the concept\_dimension table

**concept:** One row of data from the concept\_dimension table.

**concept\_path:**

**concept\_cd:** A unique code that represents a concept.

**name\_char:** A string name that represents this concept, idea or person.

**concept\_blob:** XML data that includes partially structured and unstructured data about a concept.

**observer\_set:** data from the provider\_dimension table

**observer:** One row of data from the provider\_dimension table.

**observer\_path:**

**observer\_cd:** A unique code that represents an observer.

**name\_char:** A string name that represents this observer, could be person or machine.

**observer\_blob:** XML data that includes partially structured and unstructured data about an observer.

**code\_set:** data from the code\_lookup table

**code:** One row of data from the code\_lookup table.

**table\_cd:** The name of one of the 8 tables represented in the PDO

**column\_cd:** The column name of the table where the code is found

**code\_cd:** The code itself

**name\_char:** The human-readable description of what the code represents

**pid\_set:** data from the patient\_mapping table

**pid:** One set of mappings on a single patient\_num

**patient\_id:** A choice between Patient\_Num, (if source is HIVE) or Patient\_Id if another source. A source with “\_e” at the end is encrypted.

**patient\_map\_id:** A patient\_id that should have the same patient\_num as the patient\_id in this pid.

**eid\_set:** data from the encounter\_mapping table

**eid:** One set of mappings on a single visit\_num

**event\_id:** A choice between visit\_num, (if source is HIVE) or Visit\_Id if another source. A source with “\_e” at the end is encrypted.

**event\_map\_id:** A visit\_id that should have the same patient\_num as the visit\_id in this eid.

**observation\_set:**

**observation:** One row of data from the observation\_fact table.

**event\_id:** A choice between Encounter\_Num (if source is HIVE) or Encounter\_Id if an other source. A source with “\_e” at the end is encrypted.

**patient\_id:** A choice between Patient\_Num, (if source is HIVE) or Patient\_Id if another source. A source with “\_e” at the end is encrypted.

**concept\_cd:** A unique code that represents a concept.

**observer\_cd:** An ID that represents the provider, which could be a physician or a machine such as an MRI machine.

**start\_date:** The date that the observation was made, or that the observation started. If the data is derived or calculated from another observation (like a report) then the start date is the same as the observation it was derived or calculated from.

**modifier\_cd:** hierarchical derivations of a common observation

**ValType\_Cd:** A code representing whether a value is stored in the TVal column or NVal column.

**TVal\_Char:** A text value.

**NVal\_Num:** A numerical value.

**ValueFlag\_Cd:** A code that represents the type of value present in the NVal\_Num or the TVal\_Char.

**Quantity\_Num:** The number of observations represented by this fact.

**Units\_Cd:** A textual description of the units associated with a value.

**End\_Date:** The date that the observation ended. If the data is derived or calculated from another observation (like a report) then the end\_date is the same as the observation it was derived or calculated from.

**Location\_Cd:** A code representing the hospital associated with this visit.

**Confidence\_Num:** A code or number representing the confidence in the accuracy of the data.

**Observation\_Blob:** XML data that includes partially structured and unstructured data about an observation.

**patient\_set:** data from the visit\_dimension table

**patient:** One row of data from the visit\_dimension table.

**patient\_id:** A choice between Patient\_Num, (if source is HIVE) or Patient\_Id if another source. A source with “\_e” at the end is encrypted.

**start\_date:** The date-time that the patient was born.

**end\_date:** The date-time that the patient died.

**vital\_status\_cd:** A code to represent the meaning of the date fields above.

**param:**

**patient\_blob:** XML data that includes partially structured data about a patient

**annotationGroup:** A group of fields that appear together at the end of all tables and store annotation or administrative information:

**Update\_Date:** The date the data was last updated according to the source system from which the data was obtained. If the source system does not supply this data, it defaults to the download\_date.

**Download\_Date:** The date that the data was obtained from the source system. If the data is derived or calculated from other data, then the download\_date is the date of the calculation.

**Import\_Date:** The date the data is placed into the table of the data mart.

**SourceSystem\_Cd:** A code representing the source system that provided the data.

**Upload\_Id:** Tracking number assigned to any file uploaded.