Data Factory Dictionary

Helping you talk like a cool kid 😎

Person

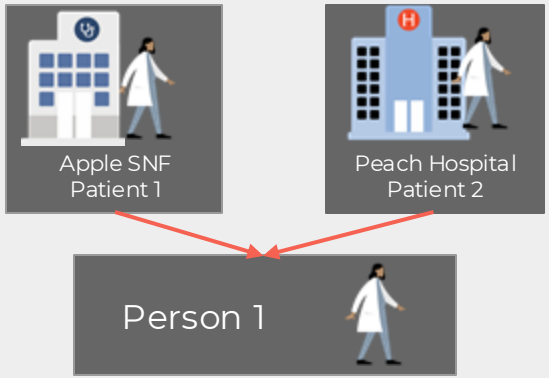
- Person is a collection of patients that share the same PMPI
- PMPI is also known as a cluster of patients

**In the database "pmpi" represents the person*

Patient

- A patient is a person's representation within the context of a single Admitting Facility (AF) or Owning-Org (OO). We will have as many patients in our system as the number of AFs the person has gone to and the number of Roster Submitters that want to coordinate care for that person

**In the database "view_patient_id", "roster_patient_id" and "patient_id" represents the patient*



eg. Mario goes to Apple SNF and Mario is a patient at Apple SNF (patient 1). Mario goes to a Peach Hospital and Mario is a patient at Peach Hospital (patient 2). But Mario is one Person (person 1, pmpi 1)

Encounter

- A patient can have multiple encounters at the same facility. Each encounter is made up of multiple events
- When a patient is admitted for the first time to a admitting facility, it means it's the patient's first encounter with the facility

**In the database "visitNumber" should remain the same for the same encounter*

Event

- Events makeup encounters
- There can be multiple events in a single encounter
- Events can be categorized as ADMIT, TRANSFER, PRESENTED, PAYOR_CHANGE, DISCHARGE etc.

**In the database “visitNumber” remains the same for different events that are a part of the same encounter).*

Event

- An action taken on a patient not seen by any other facility except by the facility who has taken the action.

**In app_state.group_encounter_view If an encounter has encounter_group_id equal to view_group_id then that following encounter is an event*

Ping

- A notification sent about a patient's current care seen by facilities who purchase pings and also seen by the facility where the care was provided.

**In app_state.group_encounter_view if an encounter has encounter_group_id not equal to view_group_id then that following encounter is a ping*

Fun Facts

Enabling successful crossovers

- **Identity Value** = MRN
- **Scheme** = comes from the MSH4 message header (sending facility, receiving facility information), we take that sending facility info in MSH4 and concatenate it with facility info to build the OID
- **Same Schema** - we will **not** do demographic matching if there are two different identity values, this will result in 2 patients
    - So if we specify the Schema more we will be more likely that we match correctly
- After we match the patient we use the visit number/account number

When visit number/account number will be consistent - when cutting over make sure the schema is different so we rely on demographic matching not just the identity value.

When visit number/account number will not be consistent - TBD

Service/ Database Name		General Use Case	How PatientPing Uses
Mirth	Service	A cross-platform interface engine used in the healthcare industry that enables the management of information using bi-directional sending of many types of messages. https://patientping.atlassian.net/wiki/spaces/MO/pages/806159441	The integration engine used to allow for real-time transmission of the following: • HL7 ADTs over TCP/IP (VPN) - Mirth receives HL7 messages and parses them into specific JSON structures • HTTPS (web service) • FTP/SFTP
HTTP	Protocol	Defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands	In the PatientPing application, we have HTTP endpoints and this is what can be found after the / • Eg. http://my.patientping.com/login the red part is an endpoint that results in the login page being called
Terminus	Service	Termius is a (Secure Shell) SSH client, it's a complete command-line solution. Used to organize, access, and connect to your servers. SSH is important because it is a protocol for creating encrypted network connections on insecure networks, such as the Internet.	A secure way to send information between different servers/services. • HL7 Pipeline - Mirth posts HL7 messages to Terminus that writes them to Kafka • CCD files - Mirth posts to Terminus the content of the CCD file to the /data endpoint • Consent files - the watcher script on our sftp server posts to Terminus the consent files to the /file endpoint • SFTP jobs- admin posts to Terminus a message to /message to trigger an sftp job
Kafka	Service	Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system, store streams of records in a fault-tolerant durable way, and process streams of records as they occur. This helps when building **real-time streaming data pipelines** that reliably get data between systems or applications and building real-time streaming applications that transform or react to the streams of data	This is where we queue data to be processed. Kafka is made up of different topics and each topic is its own queue. Messages from mirth get to Kafka and spit across 9 topics. The topic the message enters is dependent on the data source of the HL7 message (https://patientping.atlassian.net/wiki/spaces/EN/pages/25788540/Topic+Distribution+-+Kafka).

Service/ Database Name		General Use Case	How PatientPing Uses
Elastic Search	Database	Elasticsearch is a highly scalable open-source full-text search and analytics engine. It allows you to store, search, and analyze big volumes of data quickly and in near real-time.	It is one table, document store that does not allow for SQL “joins.” It is slower than a database if we were to use it everywhere, so we only use it for specific columns/tables. It indexes many columns and allows us to do fuzzy-search ● Used in parts of PMPI matching (eg. First Name) ● Used in filters since it allows for faceting
app_state	Database	a database made by PatientPing	This table is used to populate a lot of what we see in the UI of PatientPing. The most important tables in app_state are ● Group_encounter_view ○ This is the biggest table ○ This is the data pipeline's materialization of what a specific group can see about a specific encounter parsed by group_view_id ○ Great 1st step when debugging visibility issues with what can be seen in our features. full roster patient (FRP), exports, outbound, and notifications are all derived off of this table. When writing new features that access encounter data, this table should source that data. ● Patient_pmpi_map (provides the rpid to pmpi link
patientping	Database	a database made by PatientPing	this is the “raw data” and some need to know tables are: ● roster_patient ● encounters ● census_data ● groups ● roster_patient_group_rel (provides the patient to group link) ● patient_identites (we use for matching)
Janus & Anahit	Service	No general use case we made the Janus and Anahit!! Goooo Team!	The web-based platforms for our managing care providers and our point of treatment providers