Implementing a Question Story in Data Vault 2:

From Sticky Note to Info Mart

By Scott W. Ambler
ScottAmbler.com

Excerpted from Chapter 7 of the forthcoming book
Building a Continuous Data Pipeline
(Spring 2026)

About this White Paper

The material for this white paper is excerpted from the forthcoming book, *Building a Continuous Data Pipeline: One Team's Experience Enabling Corporate AI and Data-Driven Decision Making* by Scott W. Ambler, Spring 2026. The book is a novelization of a fictional data warehousing/analytics initiative applying Data Vault 2 to build a continuous data pipeline. For the most part the material has been taken verbatim, although I cut out several sidebars about candidate tools and added a few footnotes to describe context. I also cut a few sections of material that are out of scope for the context of this paper, and added a section at the end describing some ideas from Disciplined Agile™[1] that may be unfamiliar to you.

[1] "Disciplined Agile" is a trademark of Project Management Institute. See PMI.org/disciplined-agile.

Table of Contents

The Converge team, at the point where we join them, is following an agile approach. About six months later, they will evolve their way of working (WoW) into a lean continuous delivery approach. The team is currently at the end of their second construction sprint. They are building the first release of Converge, a continuous data pipeline based on Data Vault 2 (DV2) [1]. Converge will support better decision-making by management and the training and operation of enterprise-class artificial intelligence (AI) systems. The team is holding a demo and design review to show what new value they've produced and how they did so. They describe the implementation of a question story [2], a specialized form of user story that applies to data analytics/business intelligence (BI) scenarios.

The team works at BigFinCo, a large financial institution headquartered in Toronto, Canada. They are known as the Converge team, a second attempt at an enterprise data warehouse (DW). The first team, Utopia, took a traditional approach and, unfortunately, spent several years building a solution that proved insufficient to BigFinCo's needs.

What Has Come Before

Before we jump into the details of how the team works, we must first understand their context. The following points are critical in understanding that:

1. **The team invested in initiation**. A few weeks earlier, the team took the time to perform fundamental initiation activities, including forming the team, determining how they would work together, identifying the scope of the initiative, identifying a potential architecture strategy, and getting stakeholder sign-off to proceed into construction. Some people refer to the initiation phase [3] as 'Sprint 0,' some call it 'Start Up' or 'Inception,' and some agile purists naively believe none of it is required.
2. **The team collaborates closely**. Team members are currently specialists, although they work together to share their skills and learn from one another. They collaborate closely to do their work, including active stakeholder participation [4] of subject matter experts (SMEs).
3. **The organization has invested in enterprise metadata**. Enterprise data catalogue, where, among other things, they maintain their taxonomy (business concept definitions) and ontology (relationship between business concepts). The Converge team leverages and enhances this metadata whenever appropriate.
4. **Just-in-time (JIT) analysis and design**. Their initial scoping and architecture work is very high-level, just enough to guide their construction work, which is where they are now. They will explore requirements via JIT analysis and implementation details via JIT design [5], rather than capturing this information upfront as a traditional team would.

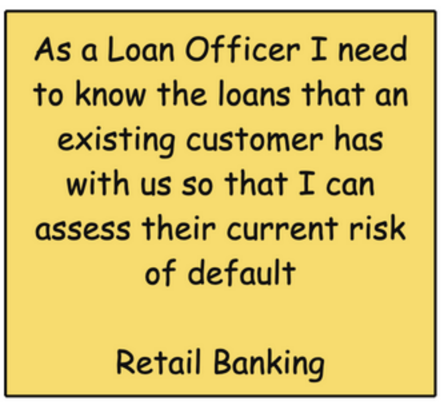
Taking the Opportunity to Learn Together

It is the end of the second construction sprint. The team has implemented their first question story and is ready to review their work. The review will consist of two parts: a demonstration of what they've built and a technical design review. The team distributed a link to a wiki page that overviews the artifacts to be reviewed, providing links to them as well so that people can prepare ahead of time if they wish. As usual, all stakeholders are invited to both sessions; only a few are interested in the design review portion.

The Demo

The team has moved their desks in their workroom[2] to clear out a space for the seven stakeholders who have chosen to attend the demo. They've borrowed chairs from an empty room down the hall and have set up a laptop and projector so that they can display their work on the whiteboard. Barbara, the backlog owner, welcomes people as they come into the room, inviting them to take a seat. Once everyone has arrived, Barbara closes the door and moves to the front of the room.

"Thanks for coming, everyone," says Barbara. "As we indicated in the agenda, we are here to demonstrate the work that we've done on this question story." She displays an image of the question story [2] as she speaks.



"We chose to work on this story first," explains Barbara, "because it was both high priority and required data from several data sources. It was sufficiently complex to show that our architecture strategy works in practice."

Andrew interjects, "If I may, I want to point out that this story only proves that the end-user reporting portion of our architecture works. We still have some more work to do, over the next two or three sprints, to show that the complete architecture works. At that point, we'll hold a milestone review with the steering committee."

"Thanks for that clarification, Andrew," replies Barbara. "Based on the question story, the team built the following report, which I'm displaying now, using Tableau[3] [6]. I'm currently showing this from my laptop, as you can see. As per our corporate user interface

---

[2] The team is co-located in a single, large Obeya room, see AgileModeling.com/essays/agilemodelingroom.htm. Nothing in this white paper requires this, and in fact the team could be fully distributed using virtual tools such as Miro and Zoom to collaborate.

[3] None of the advice in this paper is dependent upon the Tableau platform. You can easily substitute your business intelligence (BI) tooling of choice here.

guidelines, it also works on tablets and phones. However, given the amount of information, looking at this on a phone can be challenging."

Customer Loan History

Effective: July 27, 20XX

Customer: Harold Smithy
ID: 7869-6951-7

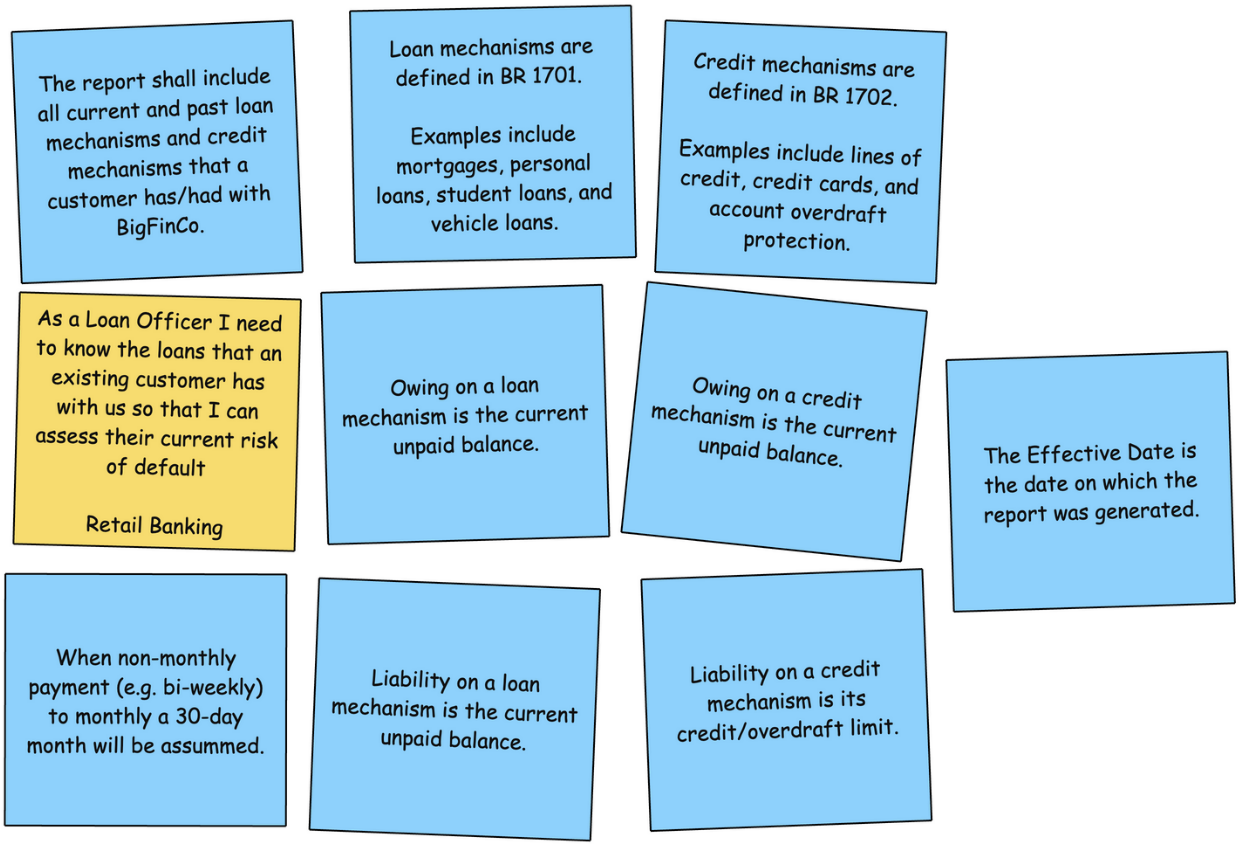
Total Owing: $681,624.34
Monthly Payments: $7,890.28
Total Liability: $682,278.80

Type	Inception	Limit	**Owing**
Mortgage: Beach House	2019-09-15	$750,000	$546,289.75
Mortgage: Residence	2001-04-01	$525,000	$43,174.13
Vehicle: 2020 Tesla	2019-12-07	$55,000	$34,093.05
Personal Loan	2016-01-26	$150,000	$26,721.87
Line of Credit	2004-03-02	$20,000	$20,000
Credit Card	1994-08-14	$12,000	$11,345.54
Vehicle: 2017 BMW	2016-11-31	$75,000	$0.00
Personal Loan	2012-10-02	$45,000	$0.00

“I really like the total owing versus liability,” says Brad, the subject matter expert (SME) representing Retail Banking. “That’s the critical information that I need when assessing risk. How are those figures calculated?”

“Let me walk you through the logic,” replies Barbara. “What I'm going to do is review the original requirements that we captured a few weeks ago and point out how they're implemented in the report. The yellow sticky, which I displayed an image of a few minutes ago, is the original question story. Now let's work through the acceptance criteria one at a time.” Barbara begins putting several blue stickies on the white board one at a time, pointing out on the screen where the corresponding calculation appears.

“OK, as you can see, all current and past loan mechanisms and credit mechanisms are listed for this customer. For each one, we indicate when the customer first opened that mechanism, the limit on that loan, and what is still owed on it. This customer, Harold Smithy, has two completely paid-off loans and six that are still owing. The total owing is calculated exactly as you would expect. Total liability is a little more interesting, as per the logic captured in the acceptance criteria. For a loan, the liability is the same as the owing amount, but for a credit mechanism, it is the overall credit or overdraft limit because that is what a customer can potentially borrow. The implication is that the liability of a customer will always be greater than or equal to their current owing amount.”



"I just wanted to point out that it is about more than just sticky notes," says Bill, a data tester. "Each of those blue sticky notes has been implemented as one or more executable tests [7]. The implication is that we will be able to automatically detect any changes to the requirements because one or more tests will break. This provides an important safety net for the Converge team."

"I have a concern," says Cathe, the Corporate Banking SME. "It took you two weeks to implement a single question story. This seems incredibly slow to me, particularly compared to our other software teams that are implementing several stories per sprint. Can you please explain this to me? And more importantly, when are we going to see something that supports my area?"

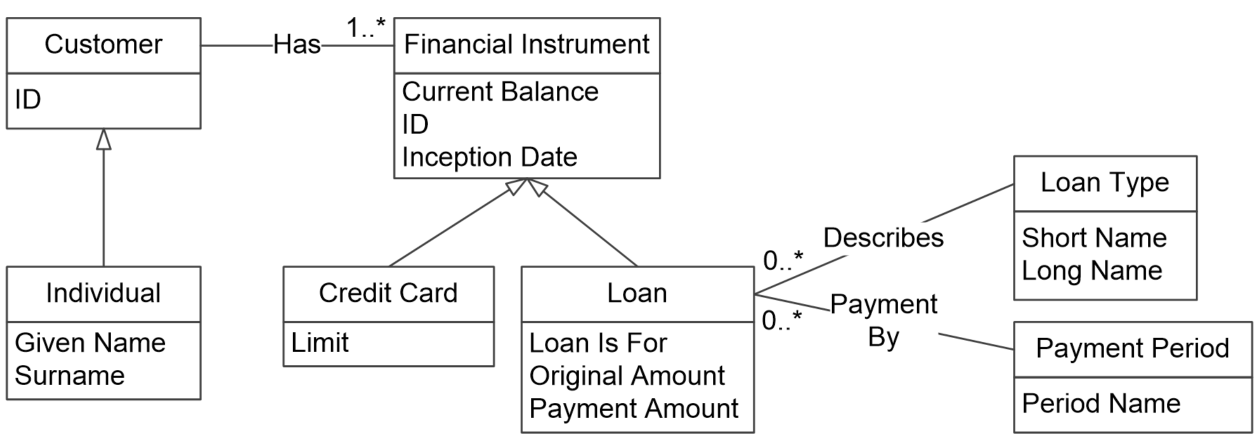
Andrew turns to her and says, "That is a completely fair question. The reason why it took the Converge team two sprints to put together such a straightforward report is that we are at the very beginning of Construction. At this point in time, we had a lot of infrastructure work to do. We needed to do the work to access the data from the appropriate legacy data sources; then we moved that data into our persistent staging area (PSA), which we needed to initiate; then we moved the data into the raw data vault and stored it in a Data Vault 2 (DV2) [1] compliant manner; then we implemented a subset of that data in our sparse business vault; we then normalized and moved the data into info marts; and finally we did the work to display the report in Tableau. This took two sprints because we were effectively starting from scratch. As time goes on, we will start to work faster because we have more foundation to build on, but right now it's the early days, and we have some initial infrastructure work to do, and that takes a bit of time. Having said that, we are only building infrastructure on a just-in-time (JIT) basis, and as we do so, we're delivering real business value, such as new reports like the one we're showing you now. We invested time during the Initiation phase, what some people call Sprint 0, to identify our target architecture, which guides us in building the infrastructure that we need when we need it. This is very different than the approach we took on the failed Utopia project, where it took over a year before we could show you anything like this."

"That means the team will start to deliver faster, as time goes on?" asks Cathe.

"Yes. That's correct," replies Andrew. "Agile people would say that our delivery velocity will increase."

"Can you please walk us through where all this data is coming from?" asks Wolfgang, an Enterprise Architect.

"Of course, let me show you." Andrew walks over to the laptop and soon displays a logical domain model using Unified Modelling Language (UML) notation [8]. "First, let's look at the required data for this report. Although we needed straightforward information about the customer, there is a bit of complexity around the loan and credit card information. As you can see, both of those entity types inherit from something called *Financial Instrument*. What we mean by this is that credit cards and loans are both financial instruments that include data about their current balance, their BigFinCo identifier, and their inception date."



Walter nods his head. "That reflects the reality of our business."

"Exactly," agrees Andrew. "We start by exploring the business needs, then we find the data to address those needs. To do so, we had to obtain data from several databases, as you can see in the list that I'm displaying now. This was because customer data is captured by multiple systems, as we all know, and because different loan mechanisms are managed by different systems. This table (see below) shows our mapping of the required business concepts to the source tables. We have performed a more detailed analysis to map the required attributes to the corresponding source columns in those tables."

Data Entity	Database	Table
Credit Card	Plutus	ClientCard
Customer	Hermes	Customer
Customer	Lending	Personal_Customer_TB
Customer	MDB	CUSTOMER
Customer	Plutus	Client
Line of Credit	Lending	LoC_TB
Mortgage	MDB	DWELLING_MORTGAGE
Personal Loan	Lending	Personal_Loan_TB
Vehicle	Lending	Vehicle_Loan_TB

"I'd just like to say that the Converge team worked closely with members of my team to identify the appropriate data sources," says Doug, the Director of Data Operations. "My people got them the access that they needed quickly, and we'll continue to do so in the future."

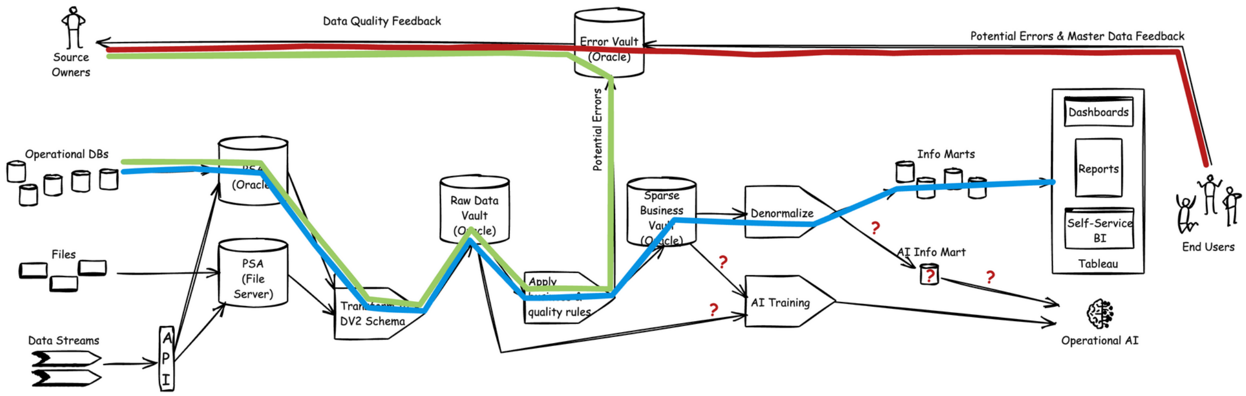
"Thanks, Doug. The team really appreciates this level of support that you're giving us," says Barbara. "Unless there's anything else, it looks like we're done with the demo."

Seeing that nobody had any more questions, Patty, Converge's Project Manager, speaks up. "Thanks, everybody. This was a short one. Immediately following this, the team is going to work through a technical design review. Everyone is welcome to participate, as usual. Let's take a quick break, and then whoever is interested should reconvene back here."

The Design Review

Ten minutes later, the team is back in the workroom. Only four stakeholders decide to participate—Wolfgang, Doug, Wayne, and Brad. Andrew sits at the laptop and displays the image of the deployment architecture diagram with the tracer lines[4] drawn over top of it.

[4] Each tracer line represents the logic required to implement a single requirement, captured in a question story or quality of service (QoS) requirement. Each tracer line goes through one or more architecturally significant portions of the Converge architecture. Implement the requirement corresponding to a tracer line means that you've proven that portion of the architecture works in practice.



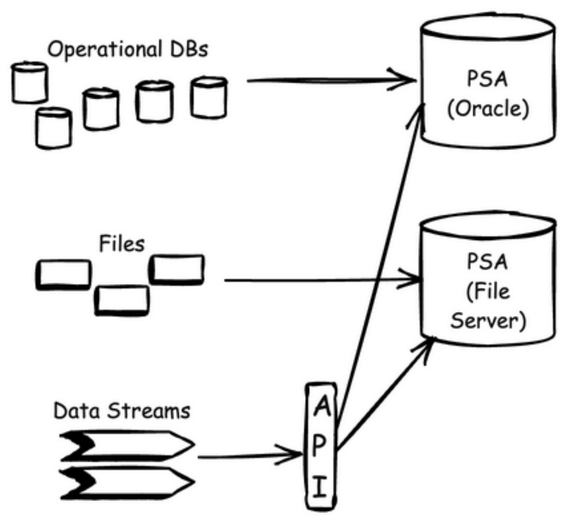
"The focus of the last two sprints has been proving the portion of the architecture called out by the blue line," says Andrew. "If you recall, we specifically identified the question story that we just demoed because it required us to implement this portion of the architecture. It also required us to work with multiple data sources, several of which had large amounts of data, which forced us to address volume."

The Persistent Staging Area (PSA)

Continuing, Andrew says, "First, we receive incoming data from operational databases and load it directly into our PSA. This portion of the PSA is implemented in Oracle[5]. File-based data, which we haven't done the work to support yet, will be loaded onto a file server. This work will occur as needed to support future question stories, which we have yet to get to. The reason why we have a PSA is to ensure that we do not lose any incoming data. This is a real risk when operating at scale as we do."

"I see you have something called an API for getting data into the database," says Brad. "What is that and why aren't we using it yet?"

"API is short for application programming interface," answers Andrew. "This is something that we built for Utopia to support real-time data, and we know that it works quite well. When we need it, which will likely be a few sprints from now, we will port over that code and use it for Converge."



[5] The team has decided to implement in Oracle on premises. Nothing in this white paper depends on that decision. You could apply any reasonable database technology and deploy part or all of your data vault to the cloud if you like.

Andrew continues. "Here's the physical data model[6] for the schema of the Oracle portion of our PSA. I want to point out several things that we're doing, all of which are best practices from Data Vault 2 (DV2) [1]. One practice is that whenever we work with a source table for the first time, we bring the entire table into our data vault. As you can see in the data model, we have brought over far more data than we need to implement the question story. The thinking is that if we need one or more elements from a source table now, chances are pretty good we're going to need more at some point in the future."

Hermes.Customer
Cust_ID: NUMBER(9)
GivenName: VARCHAR2(40)
Surname: VARCHAR2(40)
+ 27 other data elements

Plutus.Customer
ID: NUMBER(9)
FirstName: VARCHAR2(25)
LastName: VARCHAR2(30)
+ 23 other data elements

MDB.CUSTOMER
CUSTOMERID: NUMBER(9)
GIVEN_NAME: VARCHAR2(30)
SURNAME: VARCHAR2(30)
+ 32 other data elements

Lending.Personal_Customer_TB
Customer_ID: VARCHAR(11)
First_Name: VARCHAR2(30)
Last_Name: VARCHAR2(30)
+ 29 other data elements

Plutus.ClientCard
CardNumber: NUMBER(16)
CardInception: DATE
CardLimit: NUMBER(8,2)
CurrentBalance: NUMBER(8,2)
+ 8 other data elements

Lending.LoC_TB
Loan_Number: NUMBER(12)
Credit_Limit: NUMBER(8,2)
Current_Balance: NUMBER(8,2)
Loan_Inception_DT: DATE
+ 13 other data elements

MDB.DWELLING_MORTGAGE
MORTGAGE_NUMBER: NUMBER(12)
AMOUNT: NUMBER(8,2)
BALANCE: NUMBER(8,2)
INCEPTION: DATE
PAYMENT_PERIOD: CHAR(2)
+ 19 other data elements

Lending.Personal_Loan_TB
Loan_Number: NUMBER(12)
Current_Balance: NUMBER(8,2)
Loan_Inception_DT: DATE
Original_Amount: NUMBER(8,2)
+ 17 other data elements

Lending.Vehicle_Loan_TB
Loan_Number: NUMBER(12)
Current_Balance: NUMBER(8,2)
Loan_Inception_DT: DATE
Original_Amount: NUMBER(8,2)
Payment_Period: CHAR(1)
+ 21 other data elements

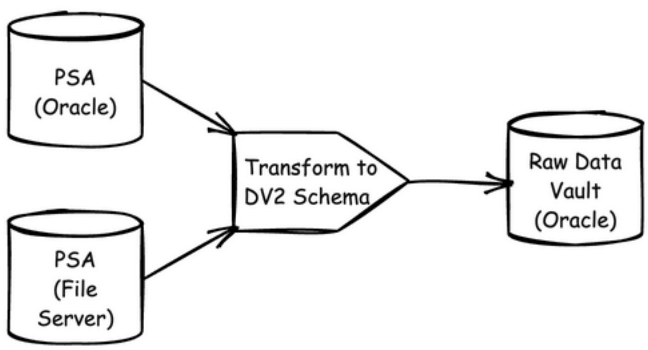
[6] See next page.

"Another practice is that we retain the names used in the data source. The only exception is that we append the name of the data source to the name of the table so that we know where it came from. You can see in the model that we're already dealing with different naming conventions across the source databases. This is nothing new, I just wanted to point it out."

"And we're following the practice of not transforming the source data at this point. For example, you can see that there are differences in business keys between data sources. Issues such as that get dealt with in the raw vault, which we will review next."

The Raw Vault

"From the PSA, we take the data and load it into the raw data vault," states Andrew. "It's at this point that we begin basic structural transformations following DV2 design practices."



"Here's the design data model[7] for the raw vault as of the end of this sprint. Every table name begins with 'rv_' to indicate it is within the raw vault. There are three types of tables in this model: hubs, links, and satellites. I've highlighted an example of each–*rv_loan_h*, *rv_customer_loan_l*, and rv_lending.personal_loan_s respectively–in the data model. Let's work through each type of table one at a time."

"A hub is where we store business keys for a business entity to resolve different key strategies in different data sources. For example, in the customer hub, we see that sometimes customers are identified by a nine-digit number and sometimes by an eleven-character ID, reflecting the two different business key strategies used in the source tables. Each hub table is indicated with '_h' appended to its name."

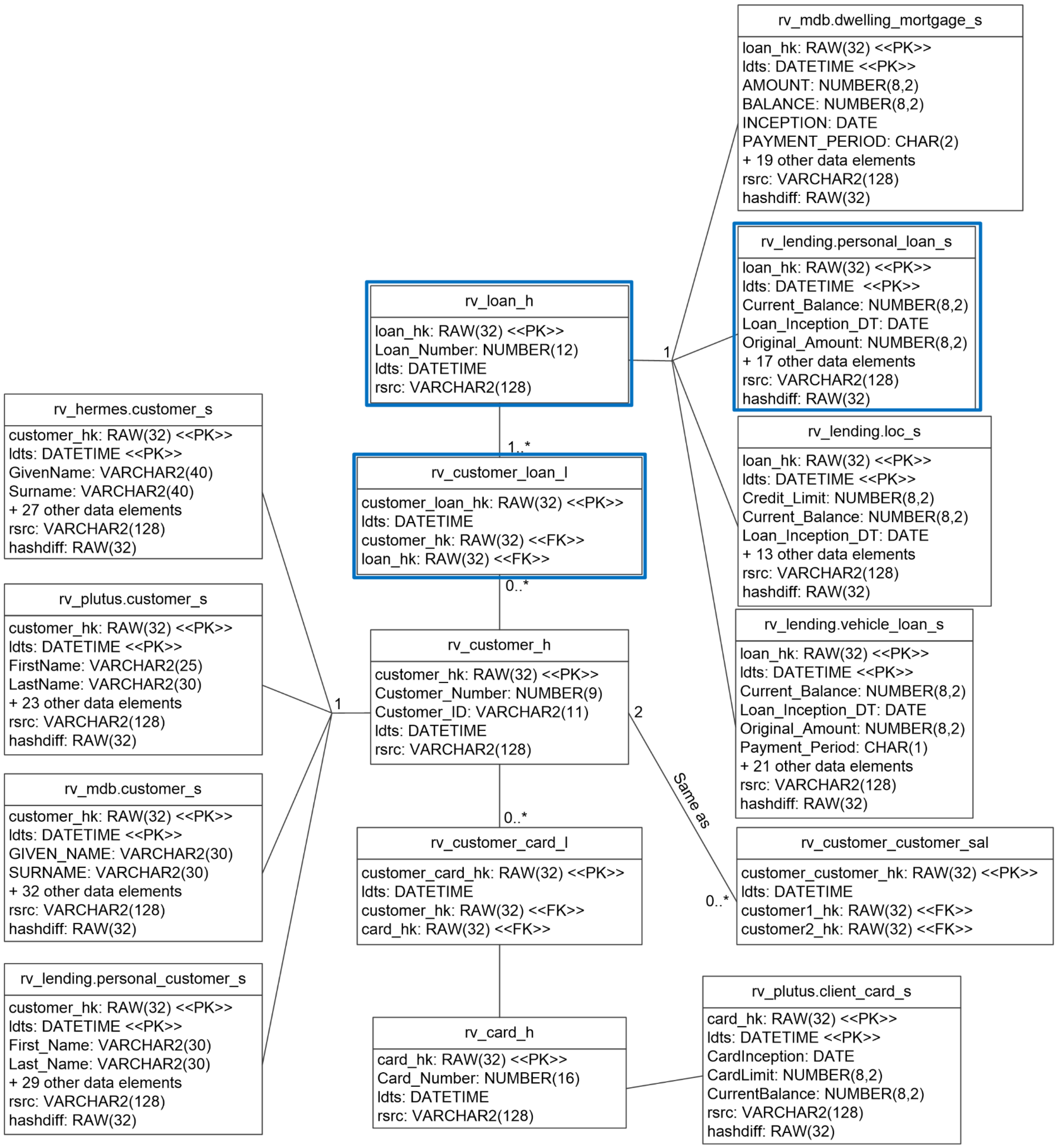
"Wait a minute," says Doug. "The primary key of the customer hub looks to be a hash key. *Customer_Number* and *Customer_ID* are both non-key columns."

"Good catch," says Esther, one of the data engineers on the team. "This is another DV2 best practice. Hash keys are used as the primary key for all tables, ensuring scalability and extensibility. Those two columns store business key values used by the source tables. The hash key for a given row is generated from one of those two values. You see that we're using RAW(32), a 32-bit binary data column, for the hash keys. We need 32 bytes to store a SHA-256 hash key value."

---

[7] See next page.

"And what are these other two columns that I see in all of the tables, *ldts* and *rsrc*? And *hashdiff* in the satellite tables?" asks Doug.



"First, *ldts* and *rsrc* are load date timestamp and record source, respectively," replies Andrew. "They are required for data lineage [9] to support audit control. Second, *hashdiff* is short for 'hash difference'. When a row is inserted into a satellite table, we calculate a hash of the column values so that we can quickly determine if a future incoming row has changed values from what we believe to be the most current value for that business concept. We basically compare the hashdiff values, and if they're different, we know that an update in the source system has occurred. For this to work, we must use the same ordering of column values each time we calculate the hash, something we capture using metadata."

"We have link tables," continues Andrew. "Link tables maintain relationships between hubs. Each link table is indicated with '_l' appended to its name. As you can see, there is a link table between the customer hub and the loan hub. Link tables are the DV2 equivalent of associative tables or relationship tables in relational data modelling [10]. We have a specialized link table, *rv_customer_customer_sal*, which captures a same as link. This table, which has the postfix of '_sal', indicates when two hub records represent the key values for the same entity. For example, one record has Customer_Number 987654321, whereas another record has customer ID '987-654-321' - different key values, but the same real-world customer."

"Finally, we have satellite tables which have '_s' at the end of their name. Satellites are where we store the business contextual data. As you can see, we have one satellite table per source table. Just like with the PSA, we bring all source data into the raw vault. We have the choice to only bring in the data that we need at this point, but we decided that it was very likely we would need most, if not all, the data because these tables capture fundamental business data for critical business concepts."

"I gotta say, this is a pretty big model for such a straightforward question story," points out Wolfgang. "I thought you said we're taking an agile modelling [5] approach and doing just enough modelling on Converge. This looks like you've done more than that."

"That's a great observation," says Andrew, "but your conclusion is wrong. Because we are following DV2 design best practices, we do in fact, end up with a schema that is more complicated than one that would result from just following standard normalization rules. We've adopted the DV2 approach because we need something scalable, auditable, and most importantly, evolvable. Following standard data modelling conventions, or the classic Kimball modelling conventions [12], won't give us these things the way that DV2 does. From an agile purist point of view, we are, in fact, over-modelling at this point. From a disciplined point of view, we are still doing just enough modelling. And you'll need to take me at my word for this, but we have not done any extra modelling yet, for any future question stories, other than the first story that we have just implemented."

Beware of "Best Practices"
Practices are contextual in nature – they work well in some situations but poorly in others [13]. I rarely use the term "best practice" and often push back against it. Yet, I readily use it with DV2. This is because DV2 is very clear about the context in which its advice is applicable – building enterprise data warehouses and the accompanying data pipeline and analytics functionality around them. That is a well-defined, narrow context. More importantly, in this context, DV2 practices are the best that I've run into in over four decades actively looking for such.

Implementing Change Data Capture (CDC)

Wolfgang nods his head, but he's still not completely convinced. "Andrew, a few minutes ago, you mentioned CDC [14] but didn't go into the details as to how DV2 supports that. Could you walk us through that?"

"Of course, good catch," replies Andrew. "A DV2 best practice is called insert only. As the name implies, whenever you have new data or a change to existing data, you insert new records into the vault; you don't update existing records. This is for scalability reasons – inserts are quick, updates are slow. The two standard columns, *ldts* and *rsrc,* enable us to distinguish the order of new data arrival and the source of the update. By comparing successive records describing the same entity, we can see how the data has changed over time and where the changes came from. This enables data audit control, which in turn supports regulatory compliance."

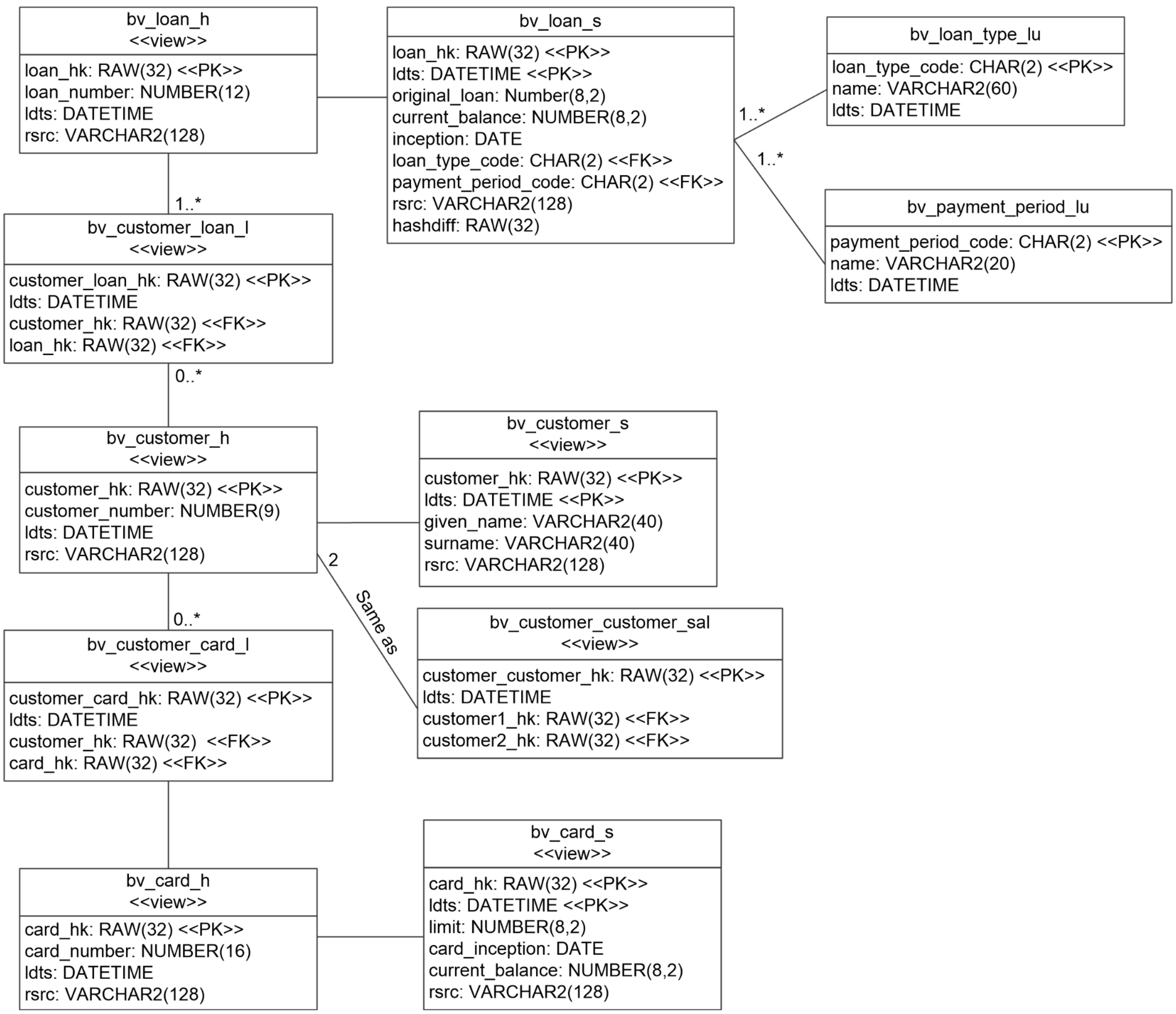
"Doesn't this make querying a bit harder?" asks Wolfgang.

"It does, but luckily it is straightforward as long as we consistently apply the DV2 design rules, which we do," replies Andrew. "You may have noticed that the link tables include the *ldts* column as part of the primary key, which is unusual from a relational modelling point of view. It is performant, which is critical in a data warehouse."

The Sparse Business Vault

"From the raw data vault, data is brought into the sparse business vault," continues Andrew. "It's at this point that we apply soft business rules to transform data into common formats as well as detect data quality problems in the incoming data. These problems are reported back to the data source owners, something we will review in a few minutes."

"I'm now displaying the data model for the sparse business vault. As you can see, only the data is needed to support the question story. It is considered 'sparse' because the only new data that is added to this vault is that which has gone through some sort of business logic/rule that has filtered it, cleaned it, performed a calculation on it, and so on. The business vault only contains data requiring the application of that business logic to ensure a correct data outcome. If that data in the raw vault is already correct, we don't touch it. Of the one hundred and twenty-three business attributes currently in the raw vault, we're only bringing eleven of them in right now. We have also introduced three new ones to support lookup tables."



"I'd like to point out two important things about this model. First, many of the business concepts are implemented as views into raw vault tables. We do this to reduce the amount of data that is being copied throughout the overall pipeline – we only want to bring data into the business vault that has been affected by some sort of business logic. The views, such as *bv_customer_h*, are on corresponding tables in the raw vault, in this case, *rv_customer_h*. Nothing has changed, so no need to copy that data. However, *bv_loan_s* is a new table rather than simply a view on the loan satellite tables in the raw vault. In this case, we're applying logic to combine data from several tables, we apply formatting rules to ensure consistency, and we add columns for the loan type (mortgage, vehicle loan, …) and payment period (bi-weekly, monthly, …). Second, notice how the names of the views and tables are preceded by 'bv_'. We do this to distinguish business vault components from raw vault components."

"This will make it easier for us if we ever decide to migrate portions of the data vault to another platform, such as Snowflake [14]," adds Doug. "This is a potential data architecture option that we are currently considering, as I've mentioned before. It is good to see design decisions like this that will ease a potential architectural refactoring in the future."

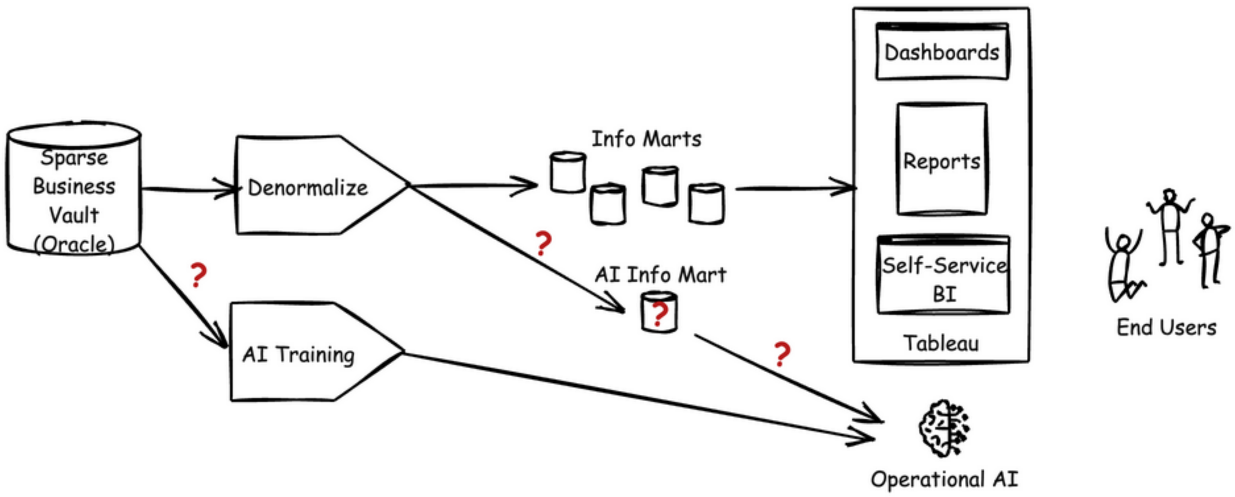
"Also, are there separate instances of Oracle, one for the raw vault, one for the business vault, and so on?" asks Doug.

"Good question," replies Andrew. "We haven't been clear about that. There's a single instance of Oracle, both for financial and performance reasons. When we say 'the business vault' or 'the raw vault', what we really mean are the portions in the database which we call the business vault or the raw vault, respectively. The data models that we've been showing are portions of the overall data model for the entire data vault. Another reason to apply naming conventions, such as consistently applying the suffix 'bv_' to all business vault tables and views."

Supporting Better Decision Making

"Let's continue and review the design of our info marts to support better decision-making by our end users. I think it would be best if Esther took over from me," says Andrew, "seeing as she knows this portion of our work best."

"I'd be happy to." Esther walks over and sits down at the laptop. She begins by displaying the portion of the deployment architecture diagram that focuses on the business intelligence (BI) aspects. "Let's begin with a quick overview of our architecture strategy. We denormalize the data from the sparse vault for reporting purposes and capture that in our info marts. To do this we apply common dimensional modelling strategies from the Kimball methodology. The data is then consumed by our end users using either Tableau or artificial intelligence (AI). Via Tableau they may use dashboards that we develop for them, self-service BI if they are sophisticated enough to do so, or access pre-defined reports that we create. In the case of this question story, we built them a predefined report."



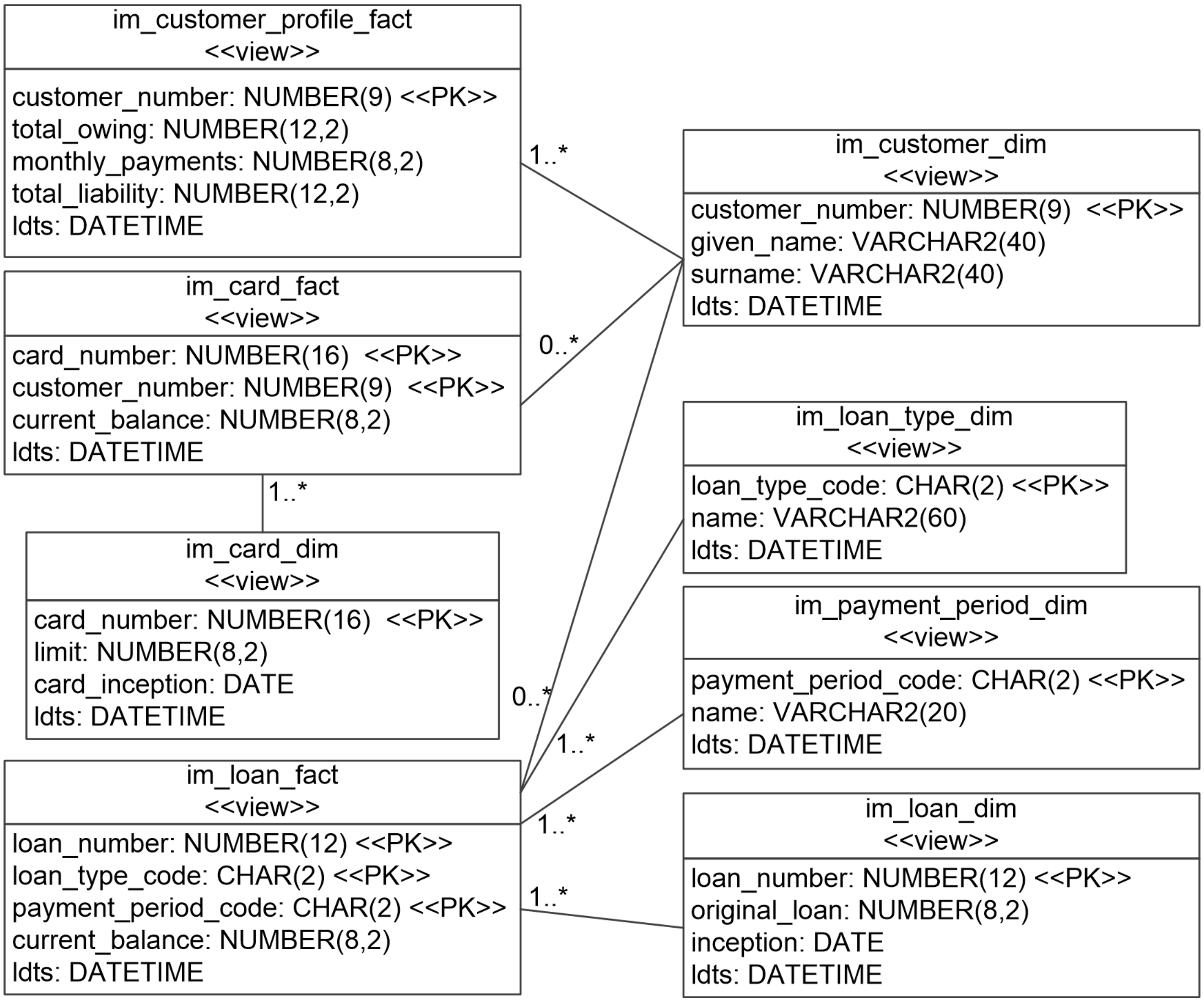
"I thought Converge wasn't supporting the AI teams yet?" asks Wayne, an independent tester.

"That is correct," answers Esther. "That's why the architecture diagram includes those red question marks; we have not yet identified how we intend to do this. We highly suspect that the AI teams will require us to build specialized info marts for them, probably using dimensional modelling[8], just as we do for the Tableau platform."

"If I may interject," says Wolfgang, "we have several AI proof-of-concept (PoC) projects underway. We should soon be moving to pilot projects within the next few months, and at that time, those teams will likely need data produced by Converge."

Esther then shows the data model for the info mart schema that supports the question story[9]. "This is the current state of the model for the info mart. As we've seen with the sparse business vault, it only contains the data required to support this question story. As we implement more question stories, this model will expand. We also intend to build several info marts over time as the business need arises. You've also likely noticed that the name of each table is preceded by 'im_', similar to what we've seen in the other data models."

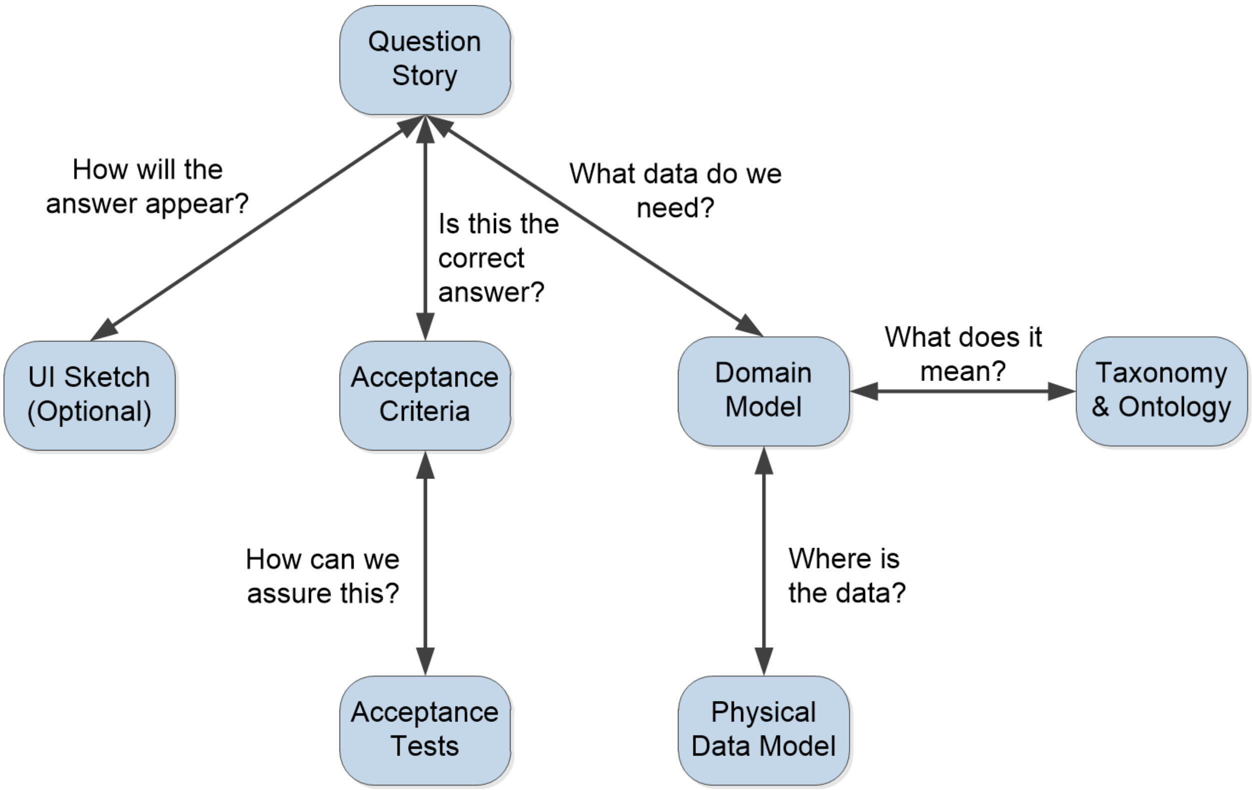
[8] Later in the book the Converge team collaborates with the AI teams to identify their data needs. They require an info mart specific to them, one that is even more denormalized that what is produced by a dimensional approach.
[9] See next page.



"Most importantly, we're taking a Kimball dimensional modelling approach. The reason why we do this is that many of the reporting and analytics tools, including but not limited to Tableau, work well with such models. This strategy also works well for predefined reports. For example, the customer profile fact table, *im_customer_profile_fact*, captures the summary information for each customer, which appears at the top of the profile report. The loan and card fact tables, *im_loan_fact* and *im_card_fact* respectively, capture the calculated current balance for those financial mechanisms."

Parting Thoughts

In this white paper you've seen the process for how the team starts with a question story, identifies acceptance criteria for that story, sketches[10] what the UI should look like to answer that question, and the resulting data models for various data stores in the pipeline architecture. This process, from the point of view of major artifacts, is depicted in the following diagram.



I hope that you have found this white paper useful. I also hope that it has piqued your interest in my forthcoming book, *Building a Continuous Data Pipeline*, which I hope to release in the Spring of 2026. I will be posting progress announcements on my LinkedIn feed and links to the book itself at AgileData.org and ScottAmbler.com.

[10] What we showed was the final 'screen shot' of the report, rather than the hand-drawn sketch created long before the report was built. This is because during demo you would show what was built.

References

1. **Data Vault 2 (DV2)**. DV2 is a proven, well-defined methodology for building an enterprise-class data warehouse. It describes key technical patterns and shows how to apply Disciplined Agile WoW. Learn more at DataVaultAlliance.com.
2. **Question stories**. Question stories describe thin slices of value for data products. See AgileData.org/essays/questionStories.html.
3. **Agile phases.**
4. **Active stakeholder participation**. This is an agile participatory design strategy, see AgileModeling.com/essays/activestakeholderparticipation.htm
5. **Agile Modeling (AM)**. AM is a practice-based methodology for modelling and documentation. See AgileModeling.com for details.
6. **Tableau**. The Tableau site is Tableau.com. Joshua N. Milligan's Learning Tableau 2025 is a great resource to learn the fundamentals of this platform.
7. **Executable specifications**. Ken Pugh's book, *Lean-Agile Acceptance Test-Driven Development: Better Software Through Collaboration*, is a great resource for anyone wanting to learn the fundamentals of capturing requirements as executable specifications.
8. **UML data modelling**. The article *UML Data Model Profile* at AgileData.org/essays/umldatamodelingprofile.html describes the notation and its semantics.
9. **Data lineage**. The blog *What is Data Lineage?* At DataCamp.com/blog/data-lineage is a great overview of this important data governance practice.
10. **Relational data modelling**. Steve Hoberman's Data Modelling Made Simple, 2nd Ed., is a great introduction.
11. **Kimball modelling**. Ralph Kimball's *Data Warehouse Toolkit Classics* is a great resource for all business intelligence (BI) professionals.
12. **Best practices**. The article *Best Practices? Practices are 'Best' Only in Context* at Ambysoft.com/essays/bestpractices.html argues that 'best practice' is a questionable term in practice.
13. **Snowflake**. The Snowflake product site is Snowflake.com. A great resource is Maja Ferle's book Snowflake Data Engineering as is Data Modelling with Snowflake 2nd Ed. by Serge Gershkovich.
14. **Change data capture (CDC)**. The article What is Change Data Capture (CDC)? A Beginner's Guide at DataCamp.com/blog/change-data-capture is a great overview of this practice.

About the Author

I have other writings about AI in my blog on ScottAmbler.com and writings about software development at Ambysoft.com. In my consulting practice, I focus on helping teams and organizations to improve their ways of working (WoW) and ways of thinking (WoT), particularly around data activities that are critical for AI/ML development. I also give presentations at conferences, to user groups, and to teams within organizations about a range of topics, including this one.

I freely share a lot of ideas and information at the following sites:

Agile Data — AgileData.org

Agile Modeling — AgileModeling.com

Ambysoft — Ambysoft.com

ScottAmbler.com