

Slide 1

CQL: Databases Done Right

Peter Gates
Conexus.ai

$\Sigma \dashv \Delta \dashv \Pi$

Outline

- CQL basics
- Computational foundation: The type side
- A unifying theme: Categories, enriched Graphs
- A brief overview of CQL constructions as they relate to familiar relational constructions.
- Demo of a CQL model which takes you from a set of related source databases, assembles them into a single target database and then migrates that target.
 - Schemas
 - Instances
 - Colimits
 - Queries
- Conclusion

9/10/2019 CQL Data Modeling 2

Here is an outline of the talk. We start with some user stories and a value proposition for CQL. We then provide a basic introduction to what CQL is and its conceptual and mathematical underpinnings.

The body of the talk will involve a series of dives into some of the core features of CQL with screen shots from working executable files or CQL models. We will then wrap up with some conclusions regarding where we think CQL fits into the data and technology landscape.

As a developer, CQL enables you to:

Evolve	Evolve databases using composable schema and data mappings.
Enrich	Enrich the expressive power of database schemas and data using path equations.
Integrate	Integrate data using graphs of data mappings.
Benefit	Benefit from improved data quality and reduced development cost by catching errors earlier in the development lifecycle.

9/10/2019 CQL Data Modeling 3

These are user stories. For the remaining part of this talk we will conduct a technical survey of specific CQL features to provide some grounding for these claims.

Value Proposition

High Assurance
Data Migration and Data Integration:
Better
Cheaper
Faster

9/10/2019 CDL Data Modeling 4

High assurance is achieved through the application of an imbedded theorem prover. This can be seen as intelligent assistance or IA. The theorem prover is able to catch errors that violate the underlying mathematics.

CQL Basics

- CQL manipulates schemas and instances and the mappings between them.
- CQL schemas and instances are designed to be related in a much more holistic way than other database systems you may be familiar with.

Schemas and instances are first class objects in CQL. What does that mean?

Mappings between schemas and instances treat those constructions as an encapsulated whole. What defines that whole?

CQL Shares Much in Common With SQL

- The nomenclature and syntax of CQL is designed to follow SQL whenever possible.
- A person familiar with SQL should be able to learn how to use CQL quickly.
- Throughout the talk we will make comparisons with SQL.

The name CQL suggests some connection with the tried and true Structured Query Language. CQL stands for Categorical Query Language. We will briefly discuss why later.

CQL syntax follows SQL syntax wherever possible and throughout the talk we will draw similarities between the two.

QOL is Based on the Mathematics of Category Theory

- Category theory is unlikely to be familiar to most of the audience so consider **Graphs**.
- For our purposes a Graph is a set of nodes connected by directional edges or arrows.
- Categories are enriched Graphs.
- Categories are a very flexible way of thinking about a set of related objects.

Let me ask the question: Does the audience feel comfortable with the concept of a directed graph?

The slide is titled "Type Side" in red text. In the top right corner, there is an orange rounded rectangle containing the text "Type Side" in white. The main content consists of two bullet points. At the bottom, there is a footer with three small items: a date "9/10/2019", the text "CQL Data Modeling", and a small number "8".

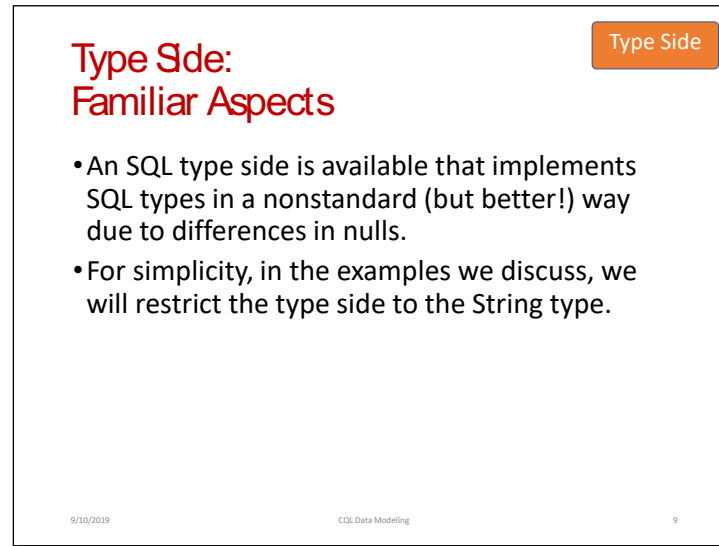
Type Side

- The type side is a configurable foundation that defines the set of types available for all other CQL constructions.
- Java types and functions can be exposed in the type side.

9/10/2019 CQL Data Modeling 8

The type side is the foundation upon which all CQL constructions are based. You can roll your own type side from scratch if you understand how to construct multi-sorted algebraic theories. Alternatively, for your convenience, you can just expose whatever Java types and functions you would like to use in your database.

Better still: Next slide



Type Side

Type Side: Familiar Aspects

- An SQL type side is available that implements SQL types in a nonstandard (but better!) way due to differences in nulls.
- For simplicity, in the examples we discuss, we will restrict the type side to the String type.

9/10/2019

CQL Data Modeling

9

The CQL type side is analogous to the built in SQL types typical to other database systems. We will explore one important difference involving the way null values are handled. CQL provides a database developer with a great deal of flexibility in defining the type side. In the examples we explore we will restrict the type side to the string type.

By the way, without digging too far into the details, the string type is a simple example of an algebra. Because an algebra is a well-defined mathematical concept, we can build other CQL constructions on the type side and apply an embedded automated theorem prover to verify the correctness of CQL constructions.

**Schemas:
Familiar Aspects**

Type Side
Schema

- A schema is a set of entities and two types of columns:
 - Attributes : Entities -> Types.
 - Foreign keys : Entities -> Entities

9/10/2019 CQL Data Modeling 10

Notice there is something implied in this slide. Can anyone see what that is? I will give a hint. What is the connection between schemas and graphs?

There is an additional column not mentioned here that you might expect based on your familiarity with relational databases.

Schemas: New Ideas

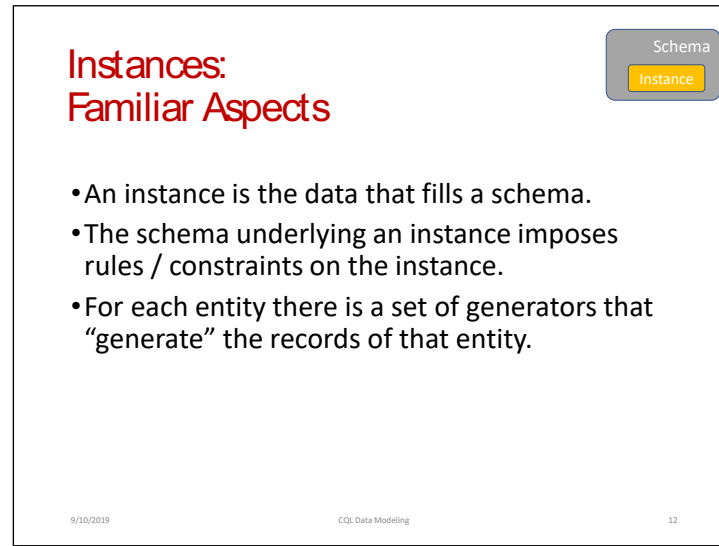
Type Side
Schema

- Schemas are first class objects that define a coherent structure with a well-defined mathematical foundation.
- Columns can be thought of as outgoing arrows and can be composed head to tail to define paths.
- Paths can be equated to extend the expressive power of a schema.

9/10/2019COL Data Modeling11

A schema is seen by the imbedded theorem prover as an intact whole with a well-defined structure. Think of a schema as a graph with some additional properties to include an identifier for each node and paths that can be used in equations.

Question: It is worth mentioning that you cannot equate any two paths. What might you want to require of a pair of paths for them to be equated?



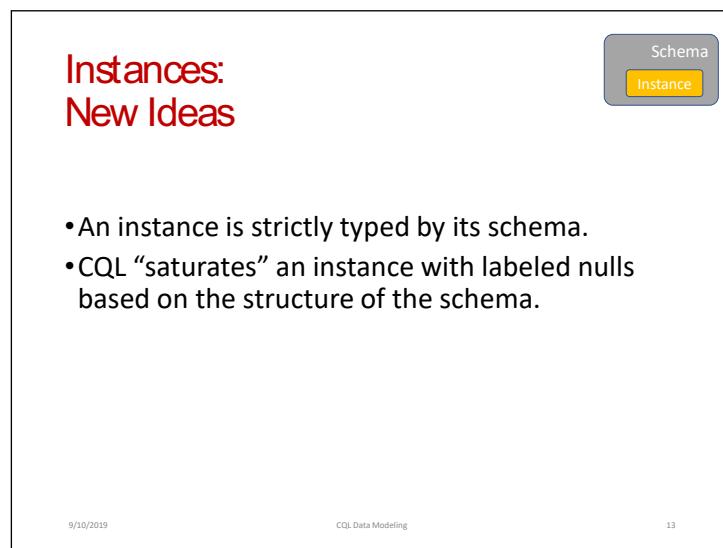
The slide content is enclosed in a rectangular frame. In the top right corner of the frame, there is a grey rounded rectangle labeled 'Schema' containing a yellow rounded rectangle labeled 'Instance'.

Instances: Familiar Aspects

- An instance is the data that fills a schema.
- The schema underlying an instance imposes rules / constraints on the instance.
- For each entity there is a set of generators that “generate” the records of that entity.

9/10/2019 CQL Data Modeling 12

CQL instances share much in common with SQL data. One important difference is that CQL instances are graphs (categories) much like RDF graphs.



**Instances:
New Ideas**

- An instance is strictly typed by its schema.
- CQL “saturates” an instance with labeled nulls based on the structure of the schema.

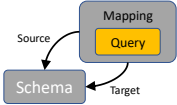
9/10/2019 CQL Data Modeling 13

CQL instances differ from SQL instances in that one can define equations. We will not explore this feature in this presentation. The other difference is that null values are much more strongly typed than what you are used to with relational null values. A CQL null is typed by both its generator (row identifier) as well as the column in which it appears.

So for example, if we had an attribute that documented a patient’s (say John Doe) body weight (column), the null value would be typed by the generator associated with John Doe’s record and the body weight column. This would be interpreted as John Doe’s unknown body weight.

The advantage of this is that labeled nulls can be used in equations like any other value. For example we might not know either John Doe’s body weight or height, but we might know that if we take John Doe’s unknown weight in kg divided by his unknown height in meters squared, that equation involving labeled nulls is equal to a BMI of 26.

Queries: Familiar Aspects

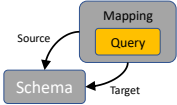


- An CQL query behaves similarly to an SQL query in the “simple” case where the target schema has a single entity.
- Because CQL queries move an entire source schema to a target schema they are functionally equivalent to an ETL workflow.

9/10/2019 CQL Data Modeling 14

A CQL query constructs single entities in a way that is very similar to SQL. CQL differs in that the query construction has a separate SQL like block for each entity in the target schema. These blocks are interrelated through variables that range over generators of the entities in the source schemas. Also each entity block includes a clause that constructs outgoing foreign keys of the target schema from generators associated with source entities. These two features provide a mechanism for mapping a source schema to a target schema that can be validated at compile time as preserving the data integrity of the source schema when mapped to the target schema.

Mappings and Queries: New Ideas



- CQL mappings are the mathematically obvious way to construct graphs of schemas.
- CQL queries are generalizations of mappings that are closely related to SQL queries.
- Mappings and queries can be composed to define paths.
- As such CQL schemas can be organized into graphs where schemas are nodes and mappings and queries are edges.
- Such graphs can be used to define data integration strategies and more generally model the enterprise landscape.

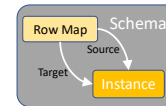
9/10/2019 CQL Data Modeling 15

CQL queries and mappings are related constructions that define ways of migrating data from one schema to another. Mappings are roughly functional in that they are many to one. Queries extend mappings by being many to many mappings. So CQL mappings are to CQL queries as functions are to relations.

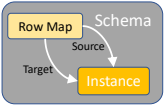
The consequence of this is that CQL mappings and queries can be used to define graphs (categories) whose nodes are databases and whose arrows are either mappings or queries. This defines an enterprise level picture of the data landscape.

A final point, we have now described schemas, instances and diagrams of related schemas as directed graphs each enriched with the additional structure of a category. This means we have three layers with which describe databases all of which are “categorified” and woven together into a mathematical rigorous tapestry.

Row Maps: That Which is Familiar



Row Maps: Are New



The diagram shows a central box with four nodes: 'Row Map' (top-left), 'Schema' (top-right), 'Target' (bottom-left), and 'Instance' (bottom-right). A curved arrow labeled 'Source' points from 'Schema' to 'Instance'. A curved arrow labeled 'Target' points from 'Instance' to 'Row Map'. A curved arrow labeled 'Target' also points from 'Row Map' to 'Instance'.

- Given a schema, an CQL row map is a mapping from one instance of that schema to another that preserves the instance structure enforced by that schema.
- Like schema maps row maps can be composed to define paths.
- As such the universe of instances **of a particular** schema can be organized into graphs where instances are nodes and row maps are edges.
- Such graphs can be used to integrate the universe of instances associated with a schema.

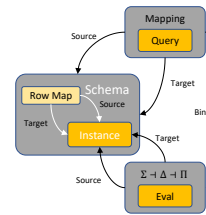
9/10/2019 CQL Data Modeling 17

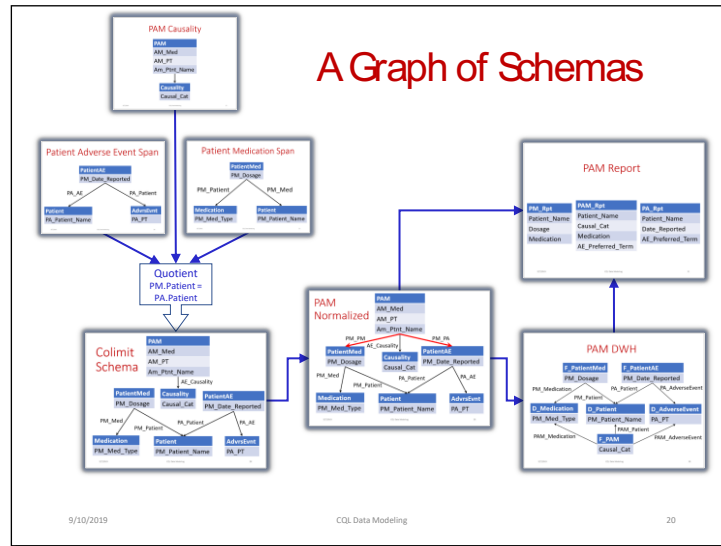
We will not discuss row maps in detail but mention them in passing as they are necessary to support the categorification of instance data. As already mentioned, instances are individually categories that look a lot like RDF graphs where the nodes are data elements and the arrows define how data elements are related.

One can also consider a category whose nodes/objects are each an instance of some schema. If instances of a schema define objects what are the arrows that relate them. The answer is row maps. Row maps define the arrows between instances specific to a schema.

But We Have Data In
Different Schemas!

Data Migration Demo

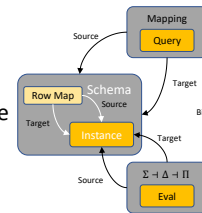




The diagram above is a schema level summary of the demo. The diagram includes the assembly of three source schemas into a single schema using the colimit quotient construction. The colimit schema in turn is then transformed several times illustrating various features of the query construction.

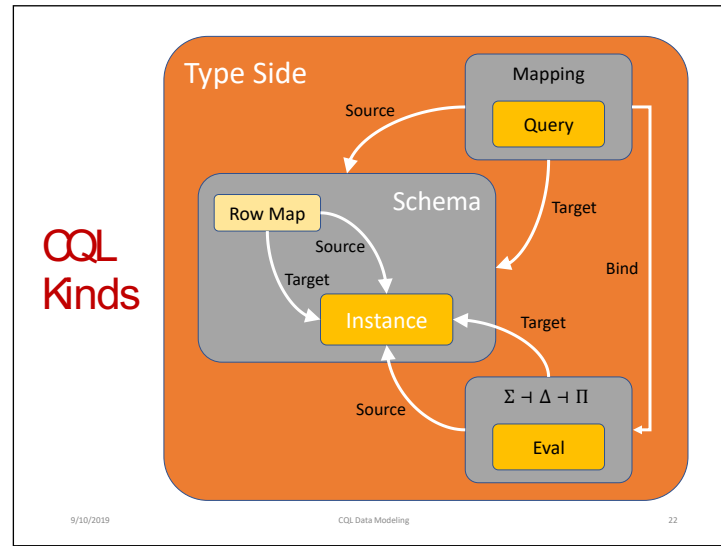
Data Migration

- Given a mapping from source schema to target schema we can push (Σ, Π) a source instance to a target instance or pull (Δ) a target instance to a source instance through the mapping.
- These three canonical ways to move data are connected by row maps: $\Sigma \rightarrow \Delta \rightarrow \Pi$
 - Σ – generalizes unions
 - Δ – generalizes projections
 - Π – generalizes joins
- Eval migrates data by first binding a query between a pair of schemas (source, target) followed by an instance of the source schema. Eval then returns an instance of the target schema. Similarly for $\Sigma \rightarrow \Delta \rightarrow \Pi$, except they bind mappings.



This slide is an overview of some of the technical features that are used to create the various ways CQL migrates data from one schema to another. The inset in the upper right-hand corner of the slide is a high-level picture of the CQL architecture.

Slide 22

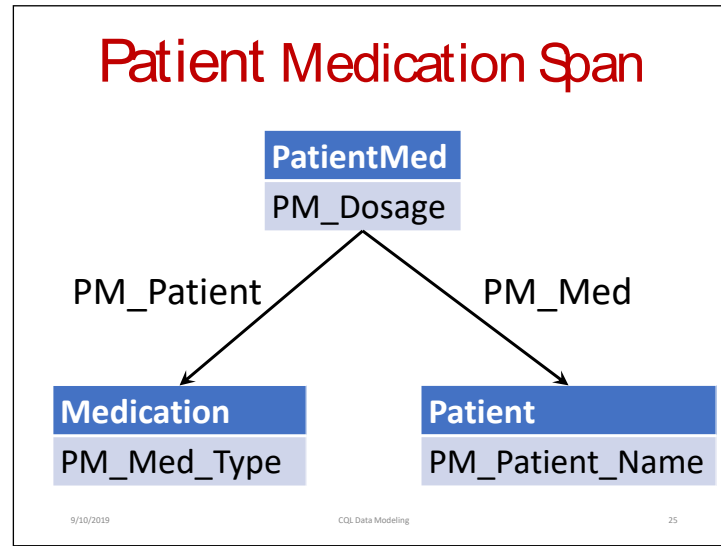


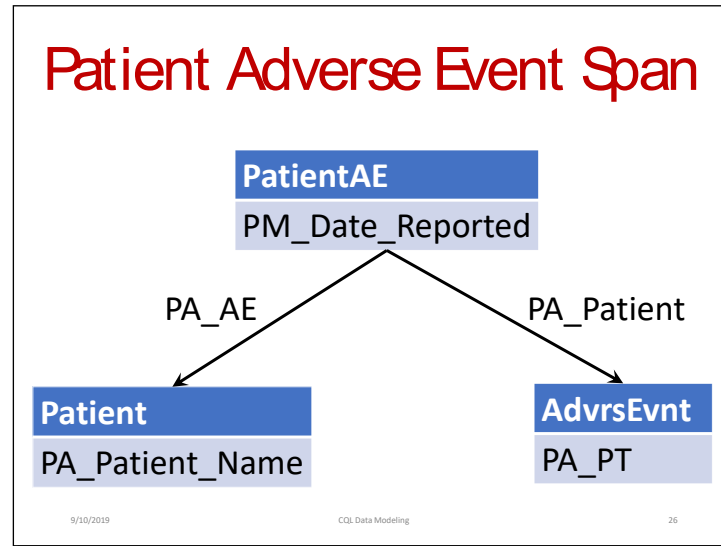
Architectural summary of CQL kinds.

Conclusion

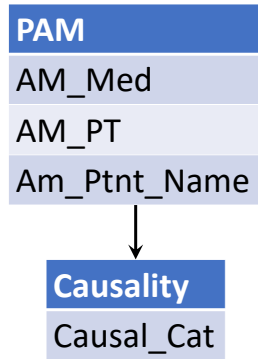
- Graphs of the universe of database instances with their maps between different databases are models of the enterprise.
- CQL's constructions from the type side foundation through schemas, instances, schema maps, row maps and enterprise maps provide a framework to bridge from business requirements to computational implementation.
- CQL is an intelligent assistant to a data architect.

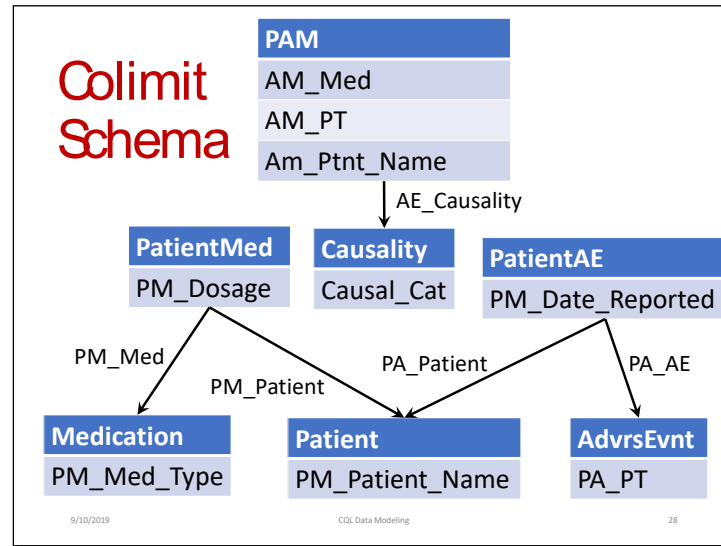
Slide 24

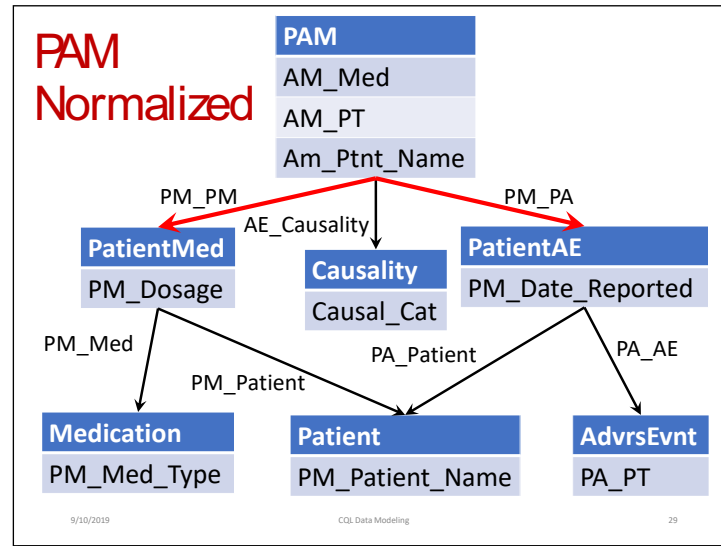


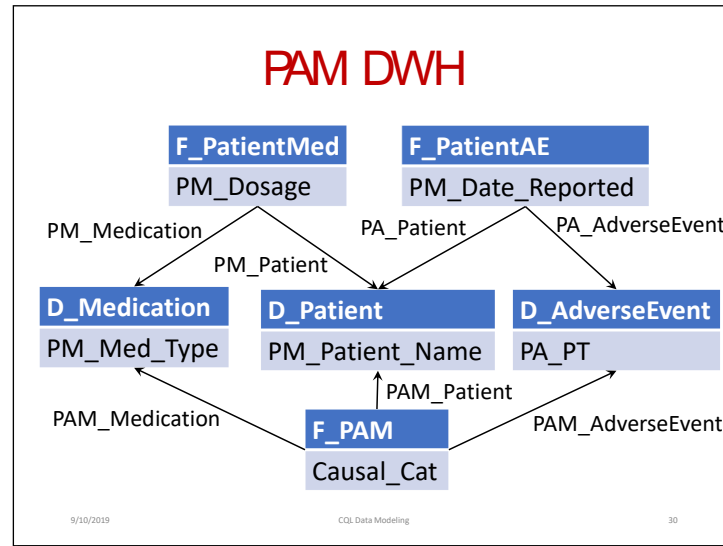


PAM Causality









PAM Report

PM_Rpt	PAM_Rpt	PA_Rpt
Patient_Name	Patient_Name	Patient_Name
Dosage	Causal_Cat	Date_Reported
Medication	Medication	AE_PREFERRED_Term
	AE_PREFERRED_Term	