Universität Stuttgart

Partial Execution of Mashup Plans During Modeling Time

Pascal Hirmer, Michael Behringer, Bernhard Mitschang

IPVS
Uni Stuttgart

Partial Execution of Mashup Plans During Modeling Time

Motivation

Motivation (I)

The Big Data Problem

- Increasing amount of data
- Data sources are oftentimes heterogeneous, distributed, and dynamic
- Coping with the *Big Data Problem* enables higher efficiency, effectiveness for problem-solving and leads to higher profits

Motivation (I)

The Big Data Problem

- Increasing amount of data
- Data sources are oftentimes heterogeneous, distributed, and dynamic
- Coping with the *Big Data Problem* enables higher efficiency, effectiveness for problem-solving and leads to higher profits

How can we extract important information from a large amount of heterogeneous, distributed data?

Motivation (I)

The Big Data Problem

- Increasing amount of data
- Data sources are oftentimes heterogeneous, distributed, and dynamic
- Coping with the *Big Data Problem* enables higher efficiency, effectiveness for problem-solving and leads to higher profits

How can we extract important information from a large amount of heterogeneous, distributed data?

- Our solution: the **FlexMash** approach
  - Flexible modeling and execution of data integration and processing **driven** by user **requirements**

Motivation (I)

The Big Data Problem

- Increasing amount of data
- Data sources are oftentimes heterogeneous, distributed, and dynamic
- Coping with the *Big Data Problem* enables higher efficiency, effectiveness for problem-solving and leads to higher profits

How can we extract important information from a large amount of heterogeneous, distributed data?

- Our solution: the **FlexMash** approach
  - Flexible modeling and execution of data integration and processing **driven** by user **requirements**

Details: Poster Session ☺

Basic Concepts

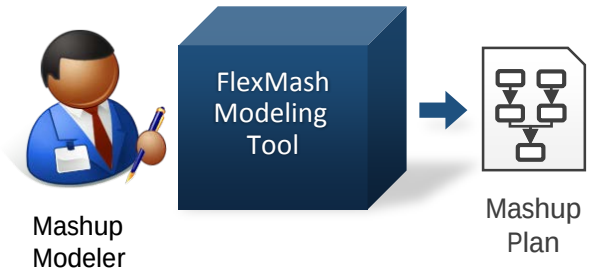
FlexMash:
How does it work?

FlexMash – How does it work? (I)

The FlexMash method

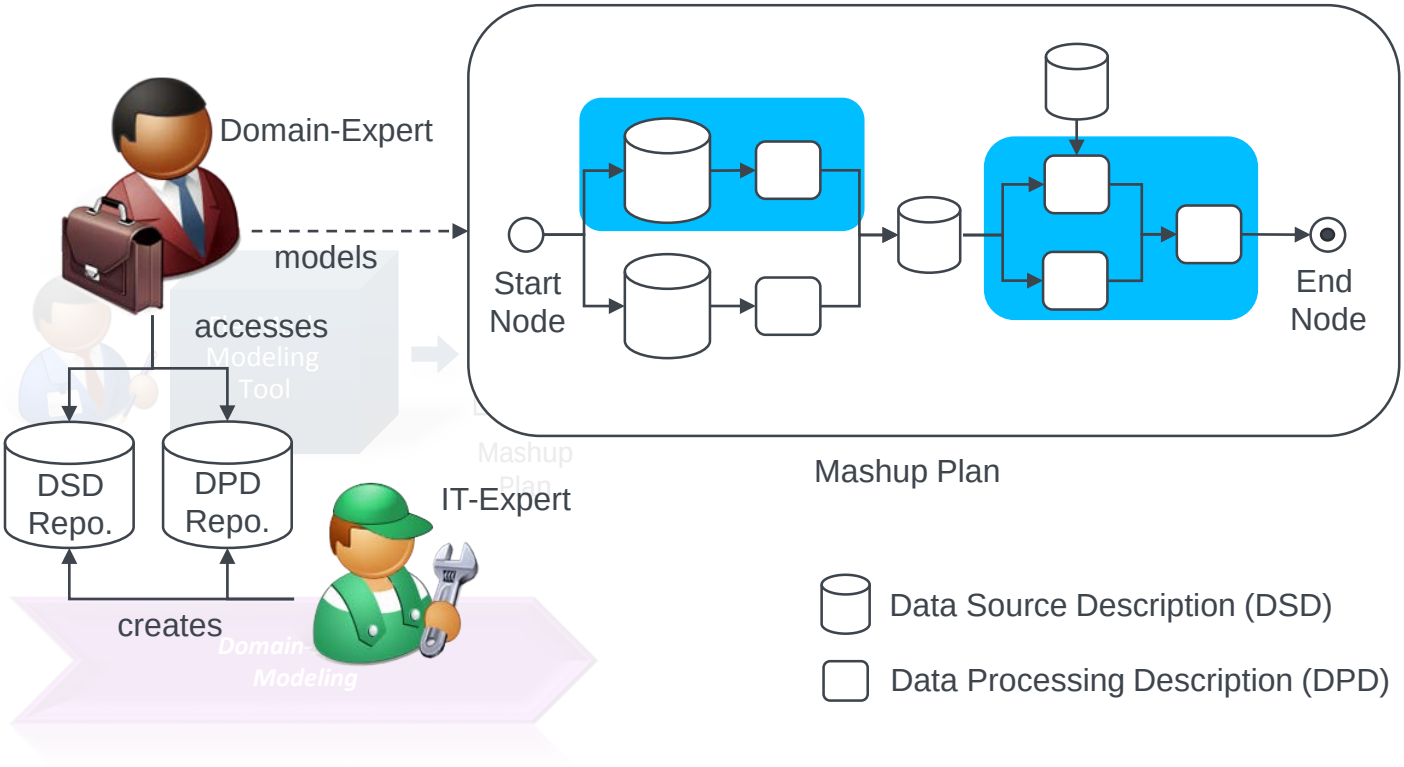
FlexMash – How does it work? (I)

The FlexMash method



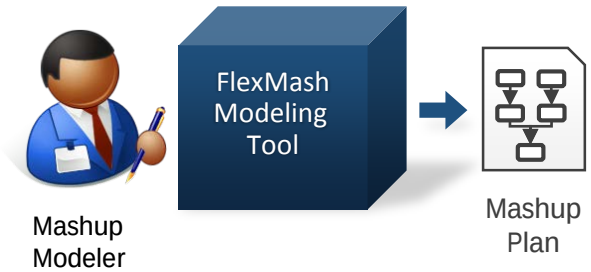
FlexMash – How does it work? (I)

The FlexMash method



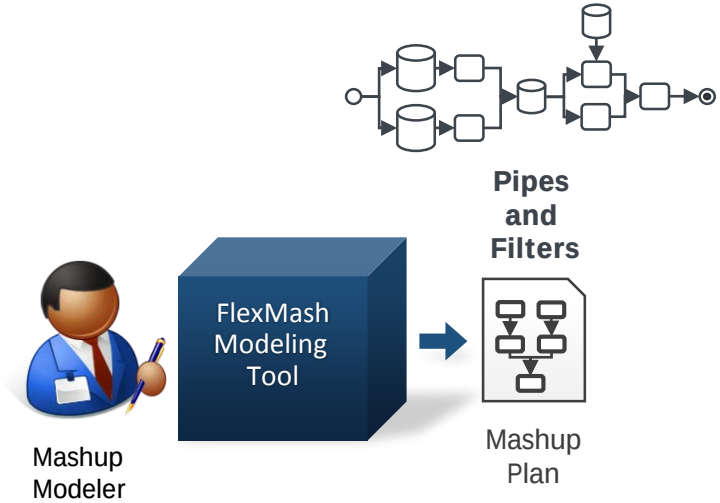
FlexMash – How does it work? (I)

The FlexMash method



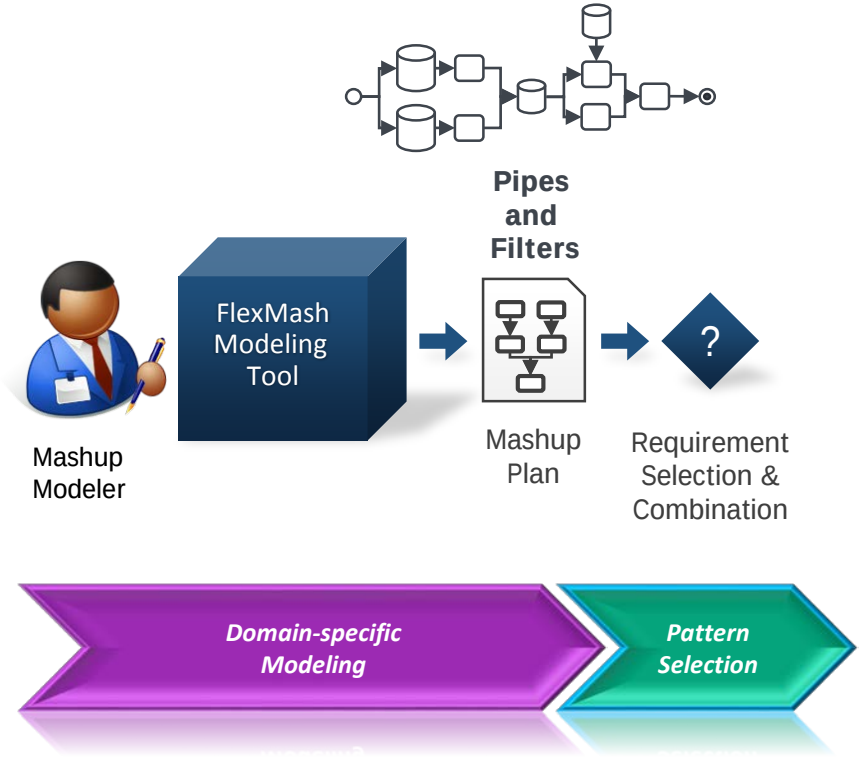
FlexMash – How does it work? (I)

The FlexMash method



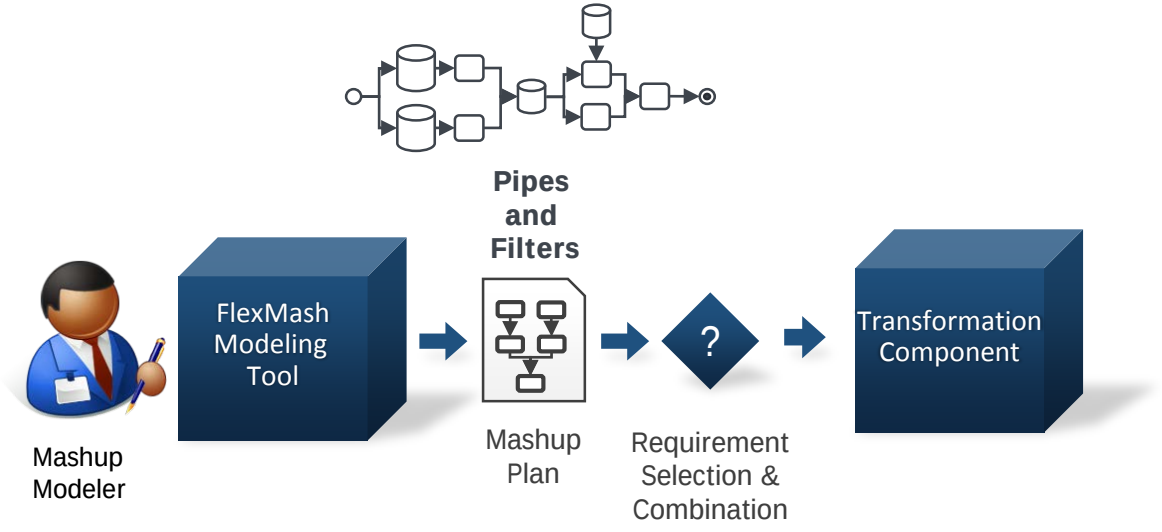
FlexMash – How does it work? (I)

The FlexMash method



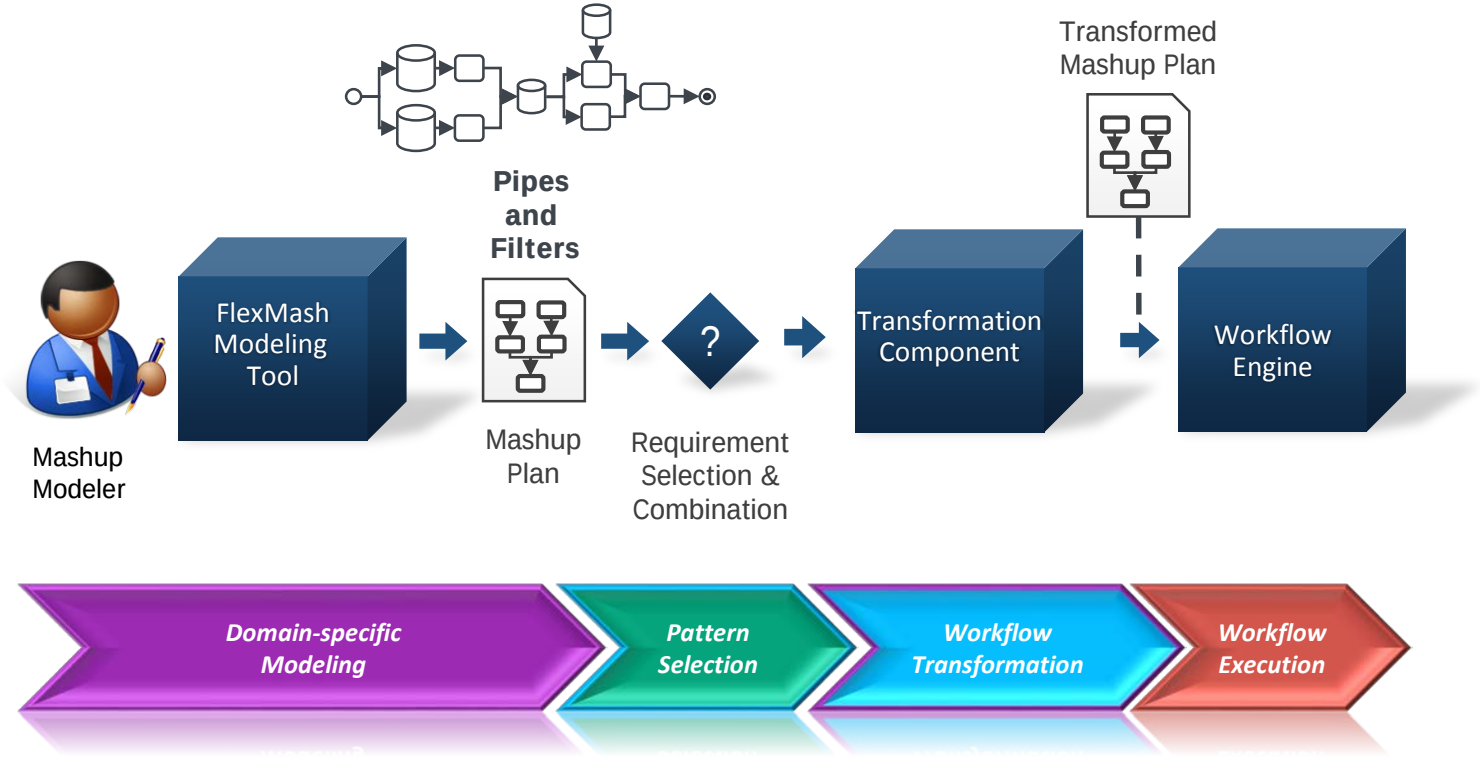
FlexMash – How does it work? (I)

The FlexMash method

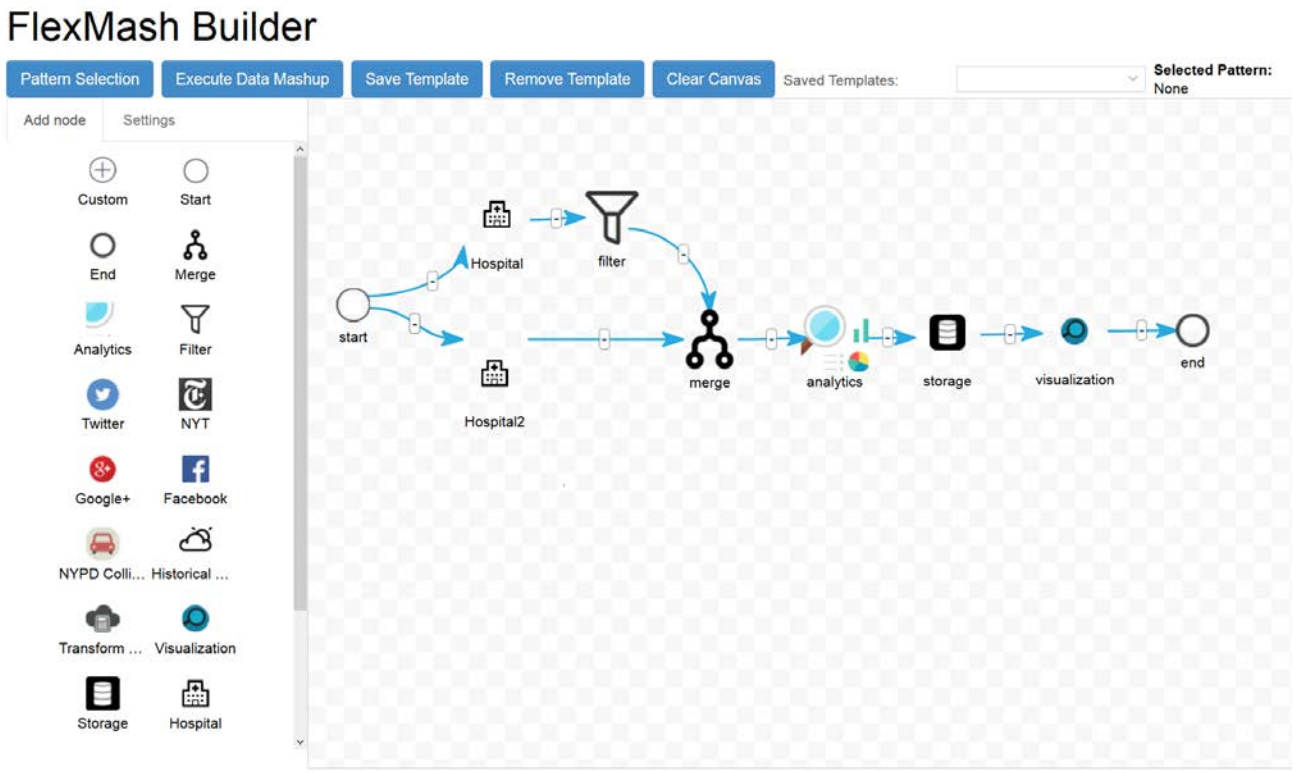


FlexMash – How does it work? (I)

The FlexMash method



Exemplary Mashup Plan



Partial Execution of Mashup Plans

Goals and Contribution

- FlexMash processes data based on workflows that are invoking **data processing services**

Partial Execution of Mashup Plans

Goals and Contribution

- FlexMash processes data based on workflows that are invoking **data processing services**

- Mashup Plans are transformed **as a whole** into executable workflows
  - ➔requires re-execution of the whole mashup plan for minor changes
  - ➔efficiency issues
  - ➔no possibility to show intermediate results

Partial Execution of Mashup Plans

Goals and Contribution

- FlexMash processes data based on workflows that are invoking **data processing services**

- Mashup Plans are transformed **as a whole** into executable workflows
  - ➔requires re-execution of the whole mashup plan for minor changes
  - ➔efficiency issues
  - ➔no possibility to show intermediate results
- **Goal of this paper:**
  - **partial execution** of Mashup Plans **during modeling time**
  - "increasing" the **efficiency**
  - enable to **show intermediate results**

Partial Execution of Mashup Plans

Goals and Contribution

- Contributions
  - Transformation and execution of single nodes
  - Showing states of execution (unprocessed, in progress, done)
  - Showing intermediate results

  ➔ First step for partial execution of Mashup Plans

Partial Execution of Mashup Plans

Goals and Contribution

- Contributions
  - Transformation and execution of single nodes
  - Showing states of execution (unprocessed, in progress, done)
  - Showing intermediate results

  ➔ First step for partial execution of Mashup Plans

- Challenges
  - Staleness of data
  - Constant changes in the Mashup Plan
  - Dependencies in the model

Related Work

Related Work

- Sonntag, M., Karastoyanova, D.: Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows. Journal of Grid Computing, 2013
- Sonntag, M.: Model-as-you-go - ein Ansatz zur flexiblen Entwicklung von wissenschaftlichen Workflows, PhD Thesis, 2016

Related Work

- Sonntag, M., Karastoyanova, D.: Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows. Journal of Grid Computing, 2013
- Sonntag, M.: Model-as-you-go - ein Ansatz zur flexiblen Entwicklung von wissenschaftlichen Workflows, PhD Thesis, 2016
- Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the kepler system, Concurrency Computing, 2006

Related Work

- Sonntag, M., Karastoyanova, D.: Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows. Journal of Grid Computing, 2013
- Sonntag, M.: Model-as-you-go - ein Ansatz zur flexiblen Entwicklung von wissenschaftlichen Workflows, PhD Thesis, 2016
- Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the kepler system, Concurrency Computing, 2006
  - ➔ Inspiration for this work
  - ➔ We apply these concepts to Mashup Plans, i.e., on an abstraction level on top of workflows
  - ➔ We further enable implicit execution ☺

Partial Execution of Mashup Plans During Modeling Time

Requirements for the approach

Requirements for the approach

- Reminder: Challenges
  - Staleness of data
  - Constant changes in the Mashup Plan
  - Dependencies in the model

Requirements for the approach

- Reminder: Challenges
  - Staleness of data
  - Constant changes in the Mashup Plan
  - Dependencies in the model

(R1) Definition of data expiration time

Requirements for the approach

- Reminder: Challenges
  - Staleness of data
  - Constant changes in the Mashup Plan
  - Dependencies in the model

(R1) Definition of data expiration time

(R2) Reduction of staleness of data

Requirements for the approach

- Reminder: Challenges
  - Staleness of data
  - Constant changes in the Mashup Plan
  - Dependencies in the model

(R1) Definition of data expiration time

(R2) Reduction of staleness of data

(R3) Dependency checks

Requirements for the approach

- Reminder: Challenges
  - Staleness of data
  - Constant changes in the Mashup Plan
  - Dependencies in the model

(R1) Definition of data expiration time

(R2) Reduction of staleness of data

(R3) Dependency checks

(R4) Split Mashup Plans into fragments

Requirements for the approach

- Reminder: Challenges
  - Staleness of data
  - Constant changes in the Mashup Plan
  - Dependencies in the model

(R1) Definition of data expiration time

(R2) Reduction of staleness of data

(R3) Dependency checks

(R4) Split Mashup Plans into fragments

(R5) Persistent storage of intermediate results

Requirements for the approach

- Reminder: Challenges
  - Staleness of data
  - Constant changes in the Mashup Plan
  - Dependencies in the model

(R1) Definition of data expiration time

(R2) Reduction of staleness of data

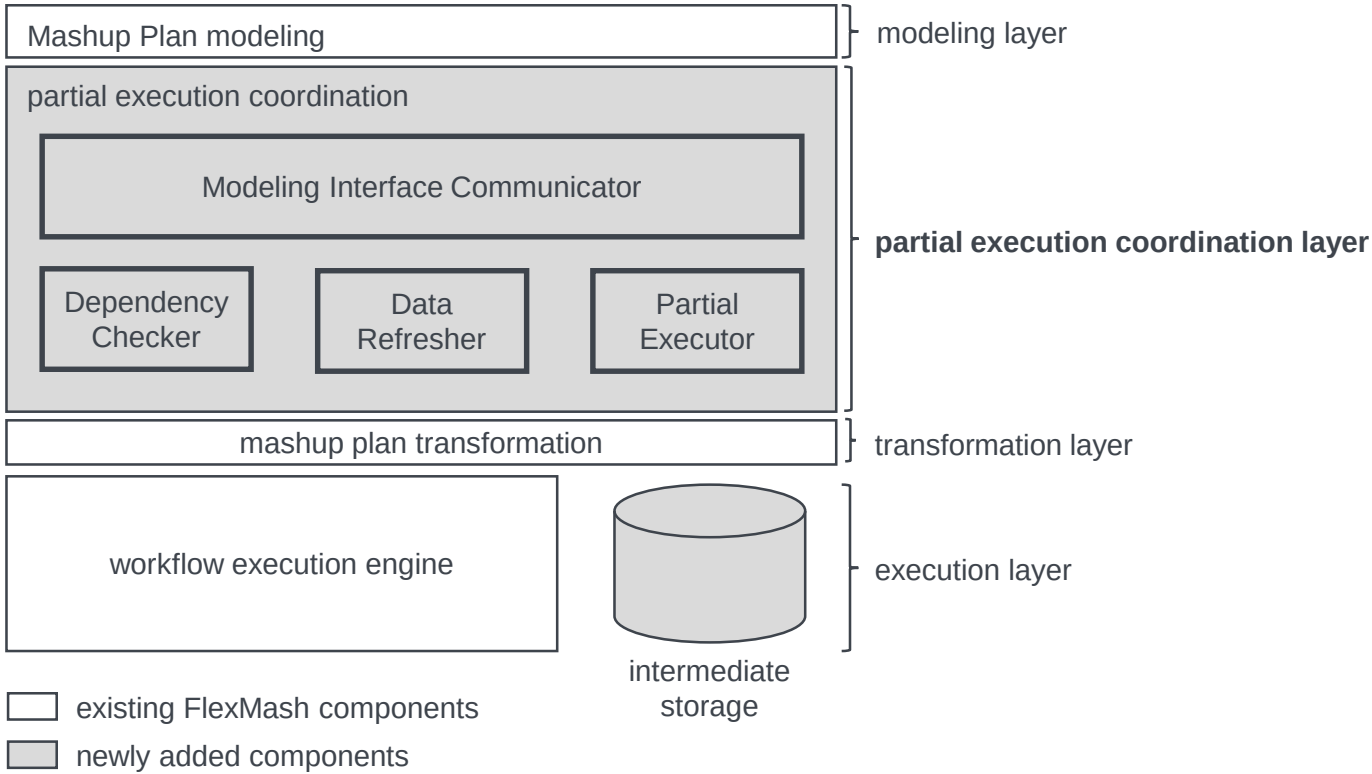
(R3) Dependency checks

(R4) Split Mashup Plans into fragments

(R5) Persistent storage of intermediate results

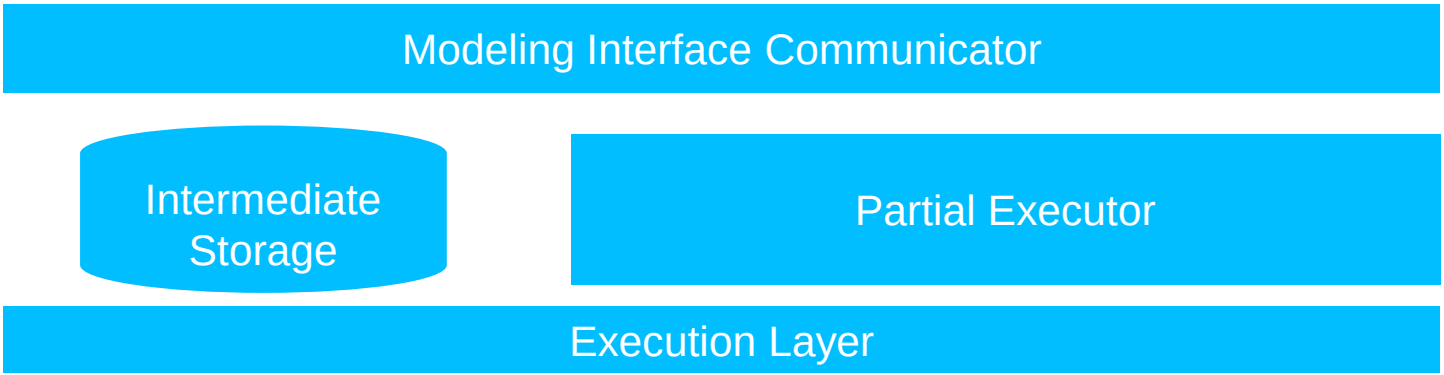
(R6) User feedback

Architecture



Simple Example

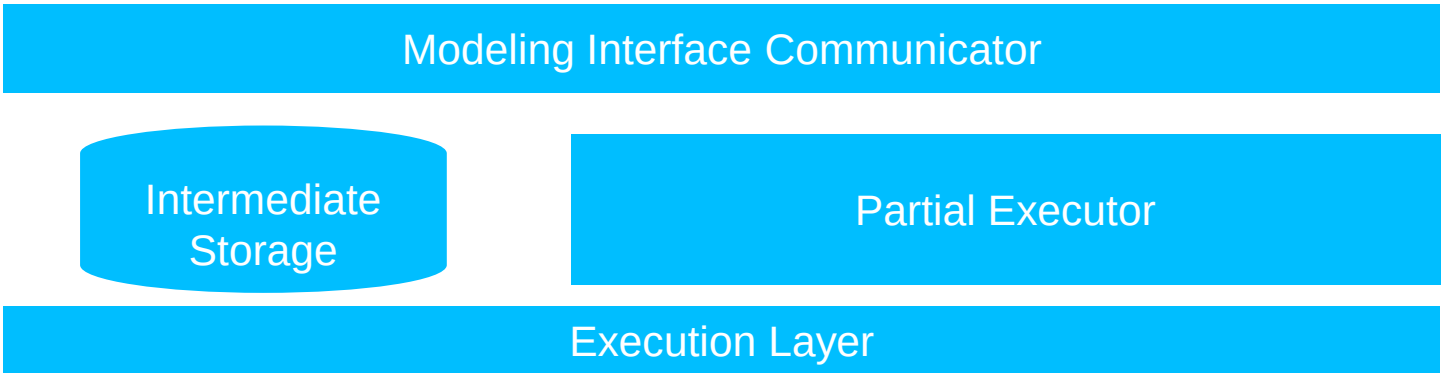
Simple Example



Simple Example

table: patients

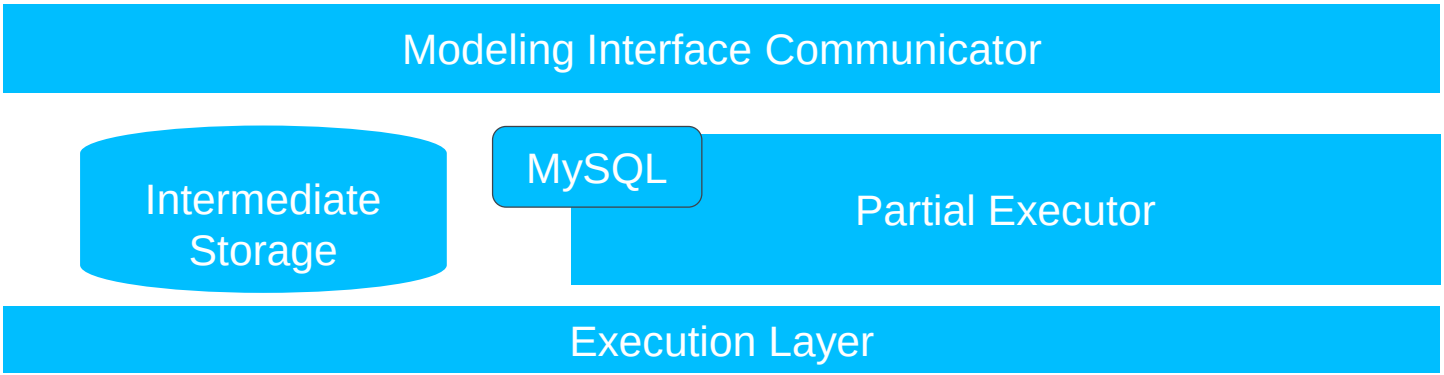
MySQL



Simple Example

table: patients

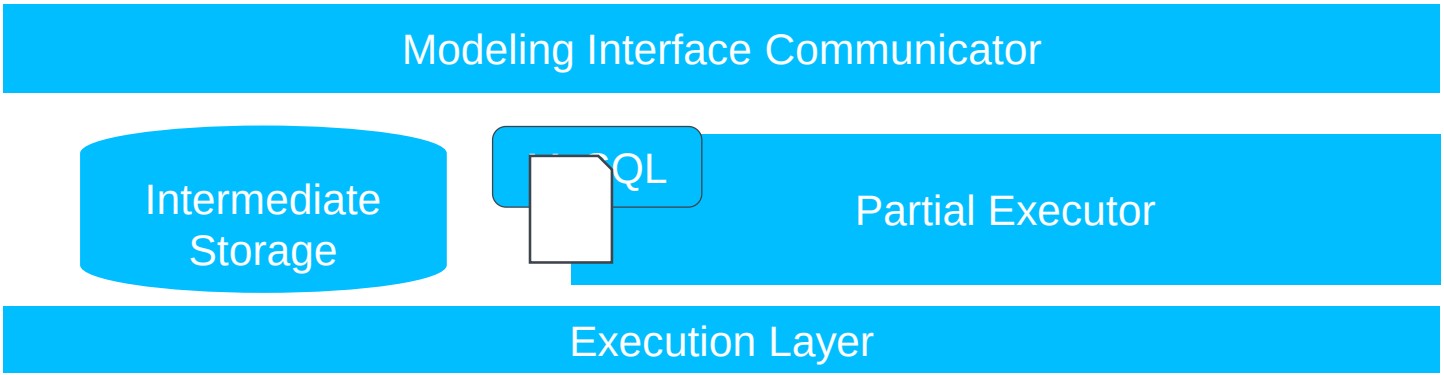
MySQL



Simple Example

table: patients

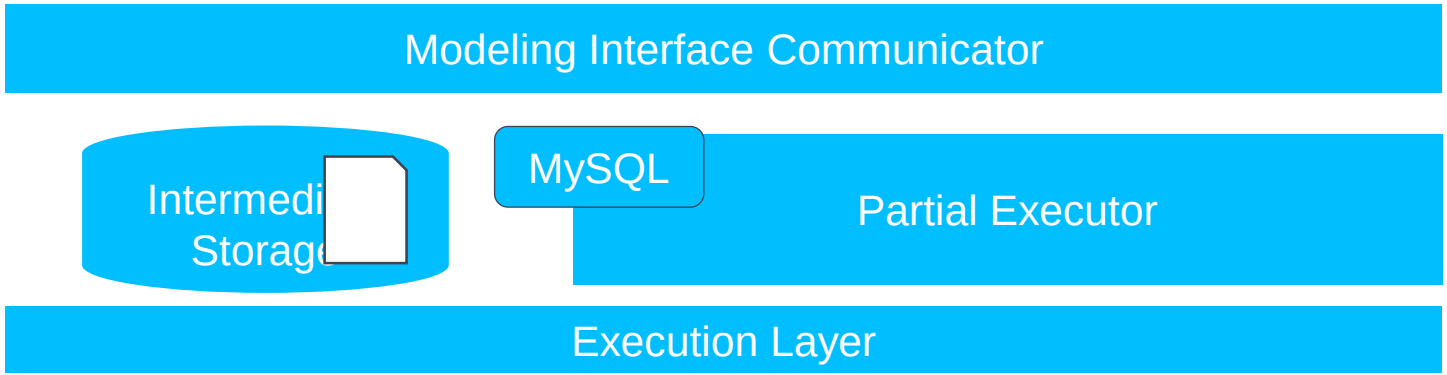
MySQL



Simple Example

table: patients

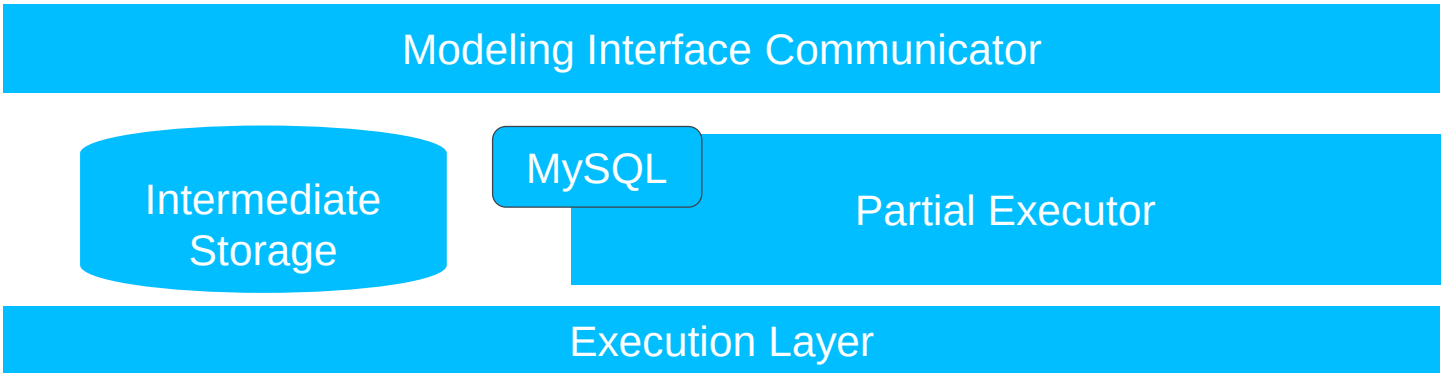
MySQL



Simple Example

table: patients

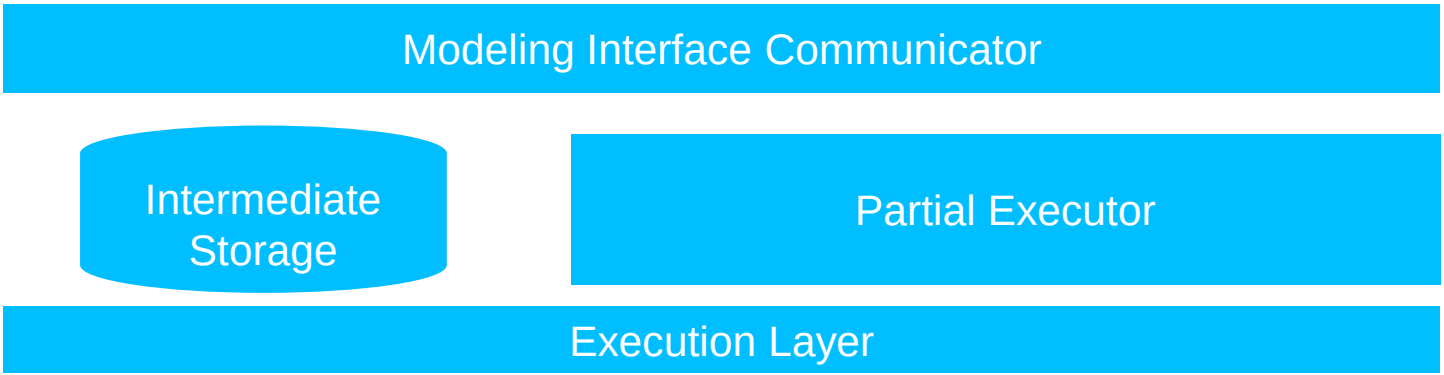
MySQL



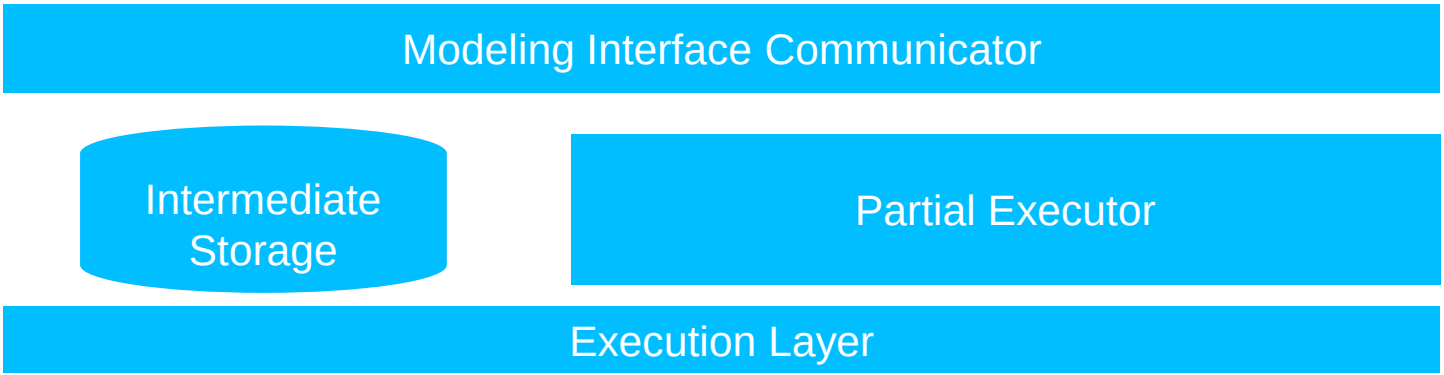
Simple Example

table: patients

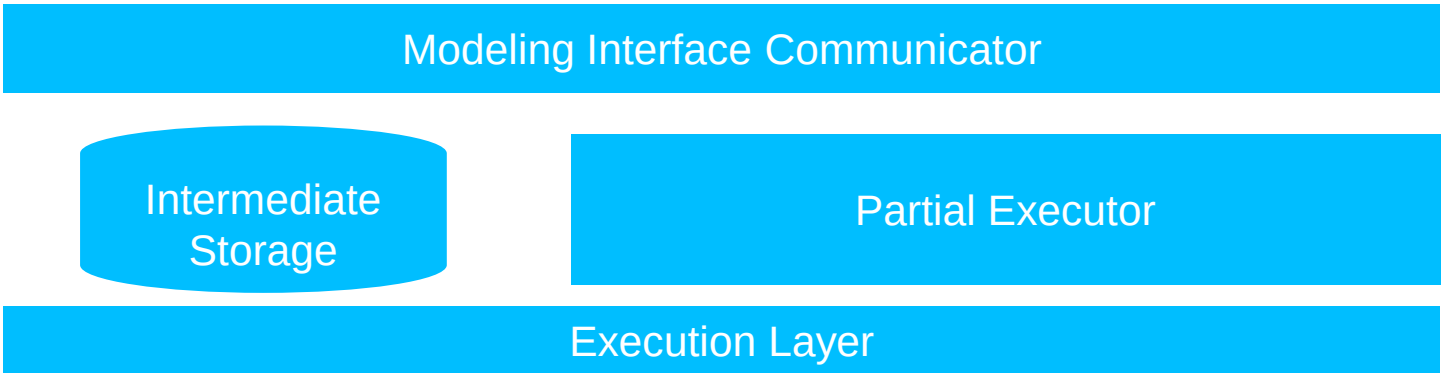
MySQL



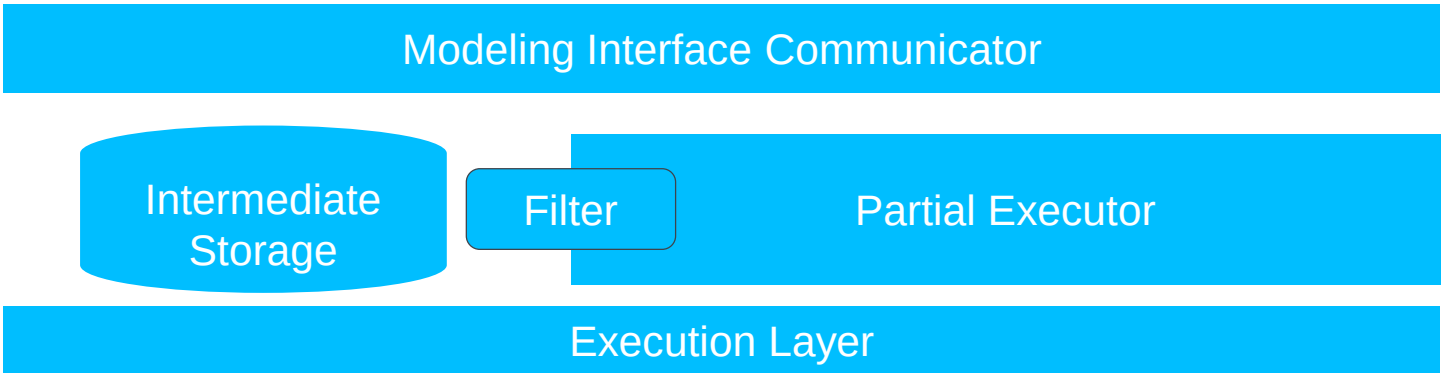
Simple Example



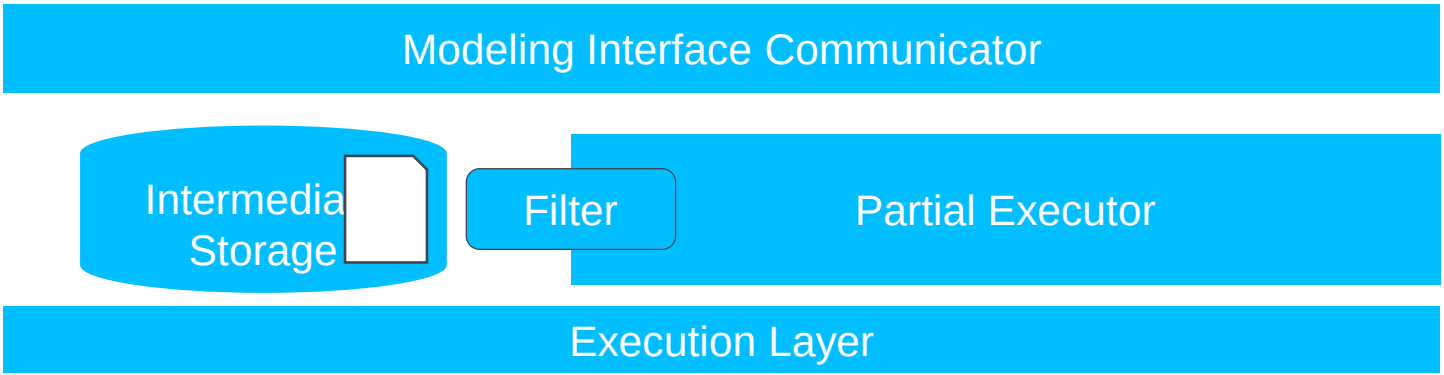
Simple Example



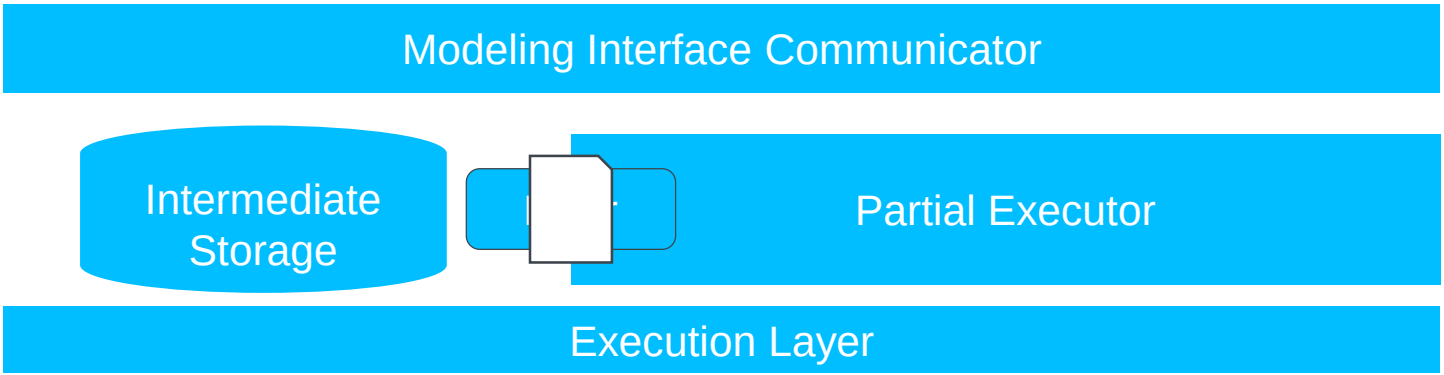
Simple Example



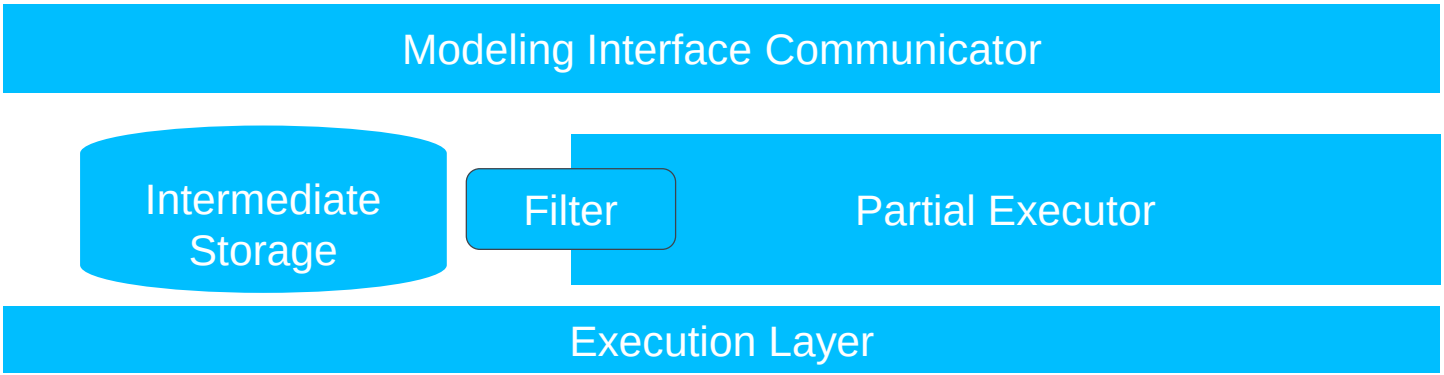
Simple Example



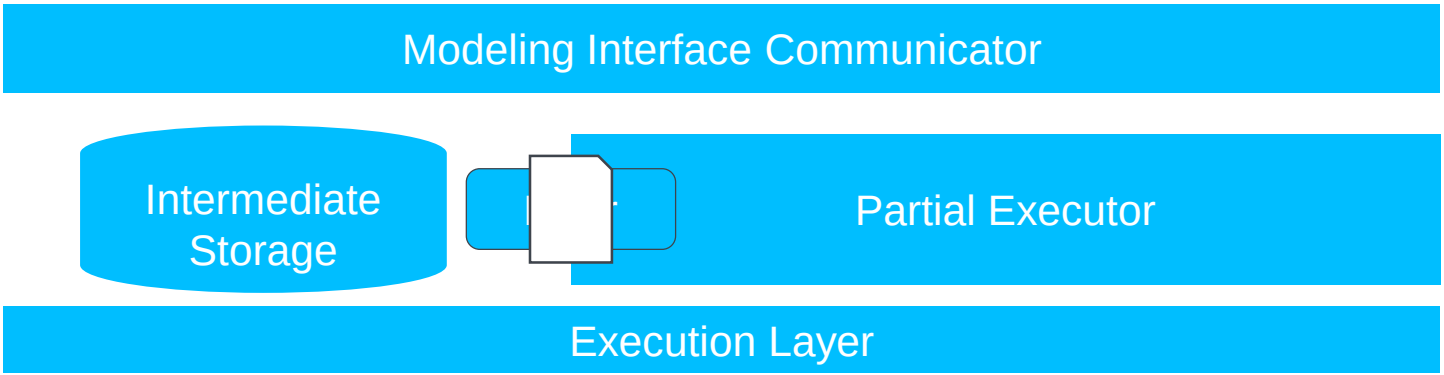
Simple Example



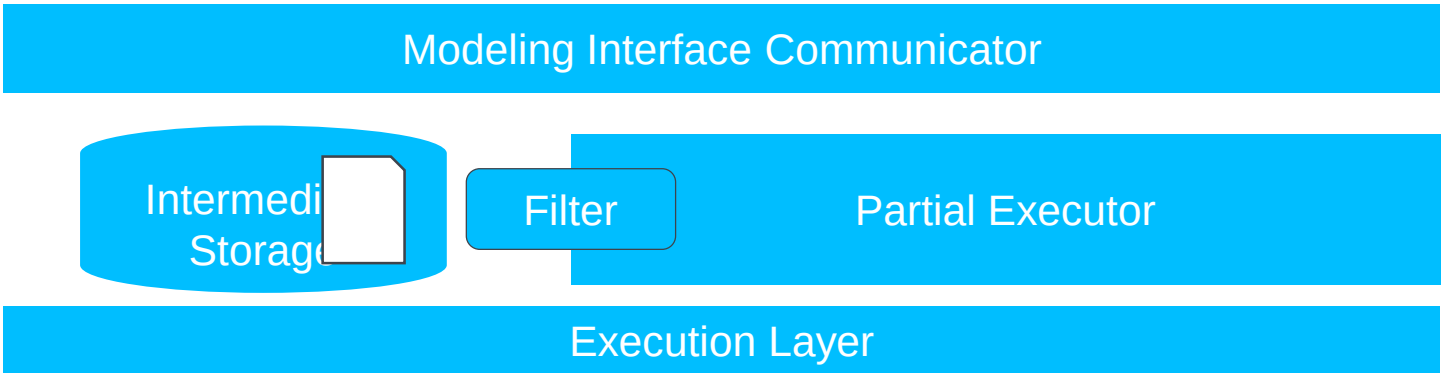
Simple Example



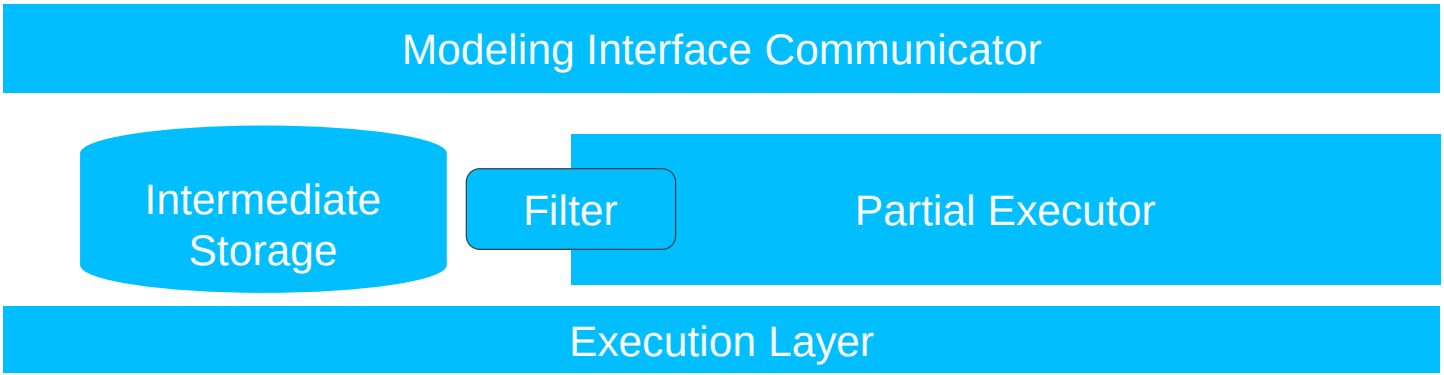
Simple Example



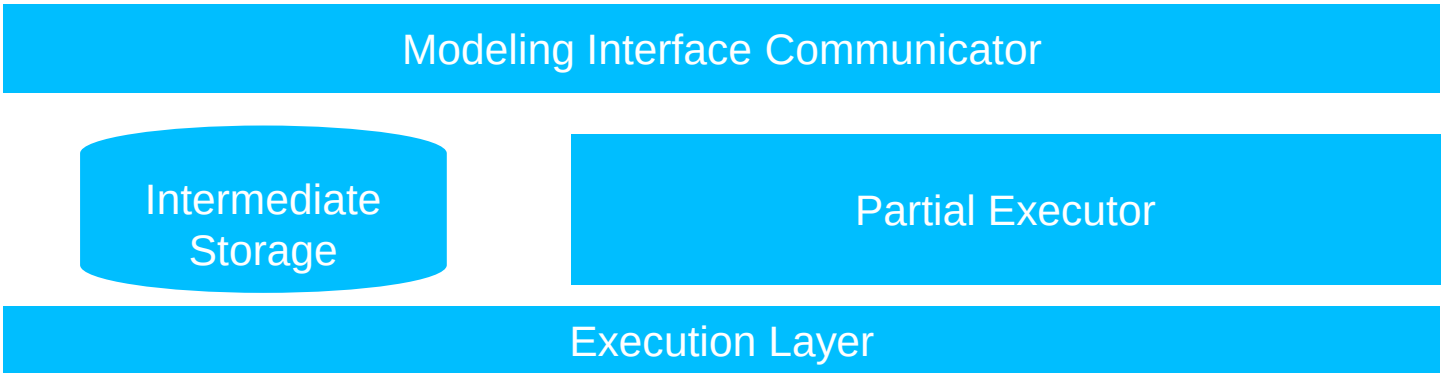
Simple Example



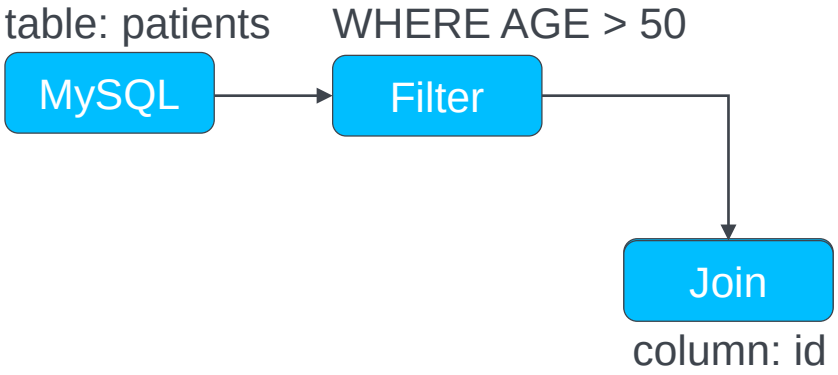
Simple Example

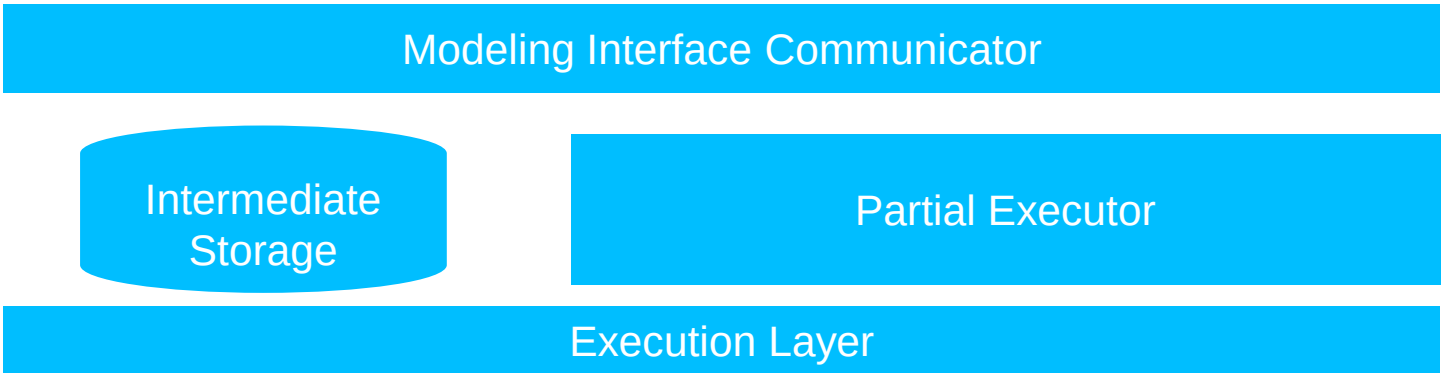


Simple Example

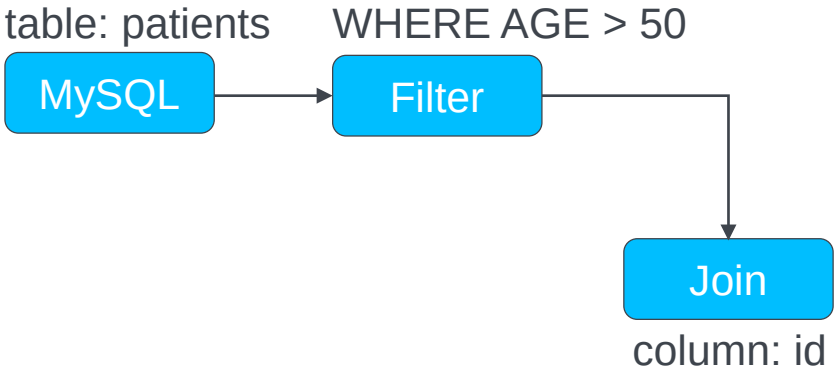


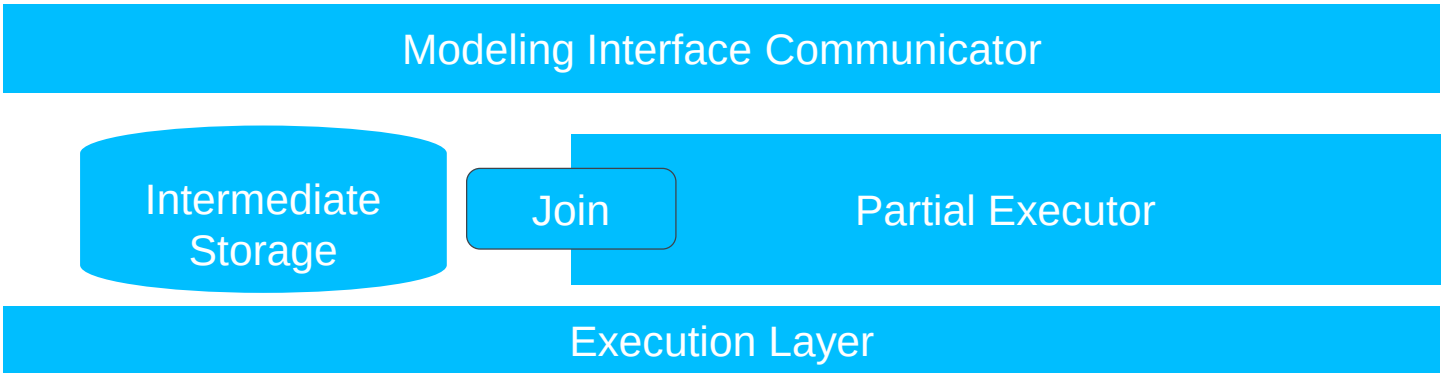
Simple Example



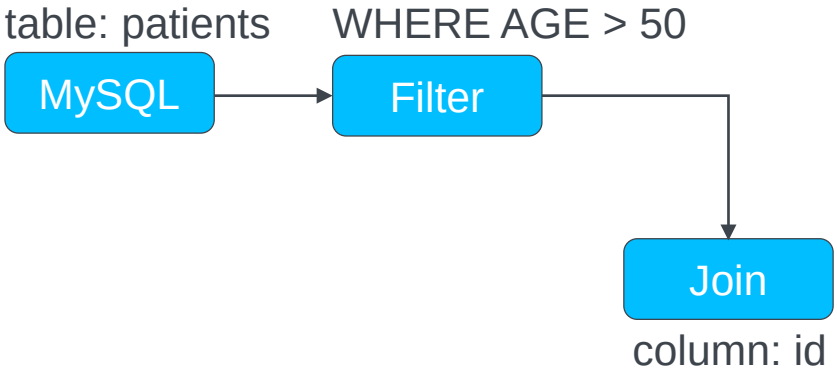


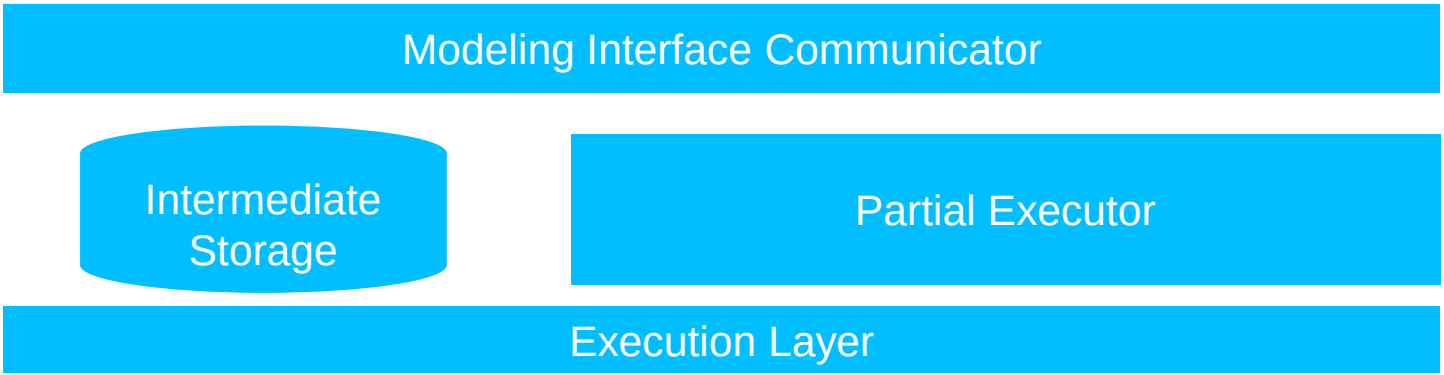
Simple Example



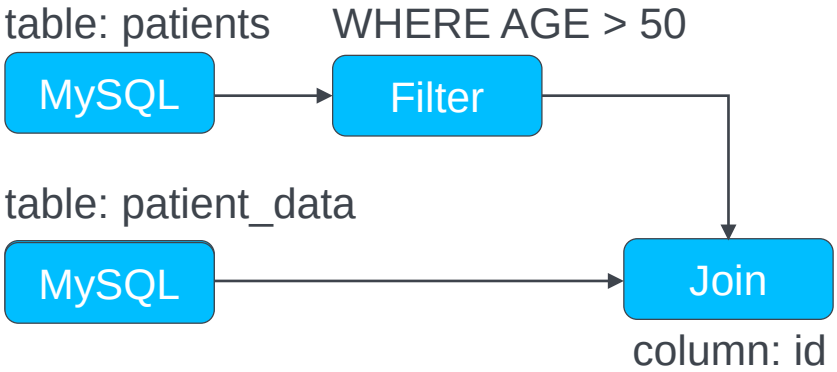


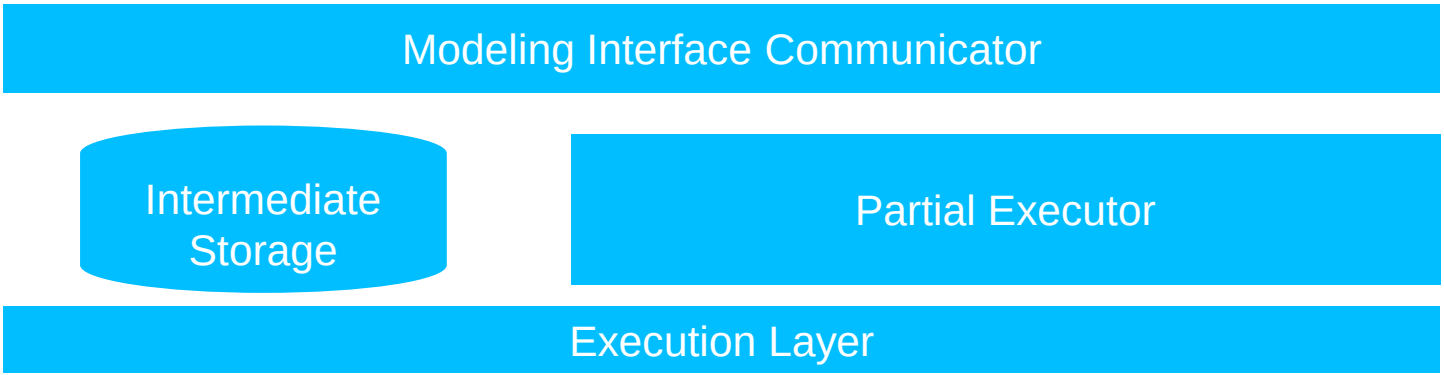
Simple Example



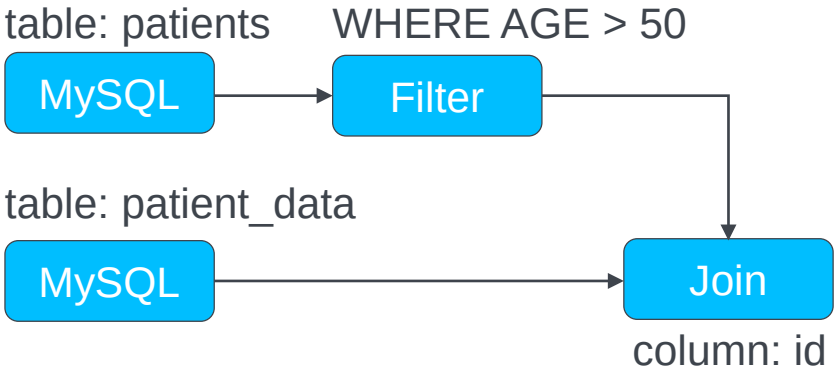


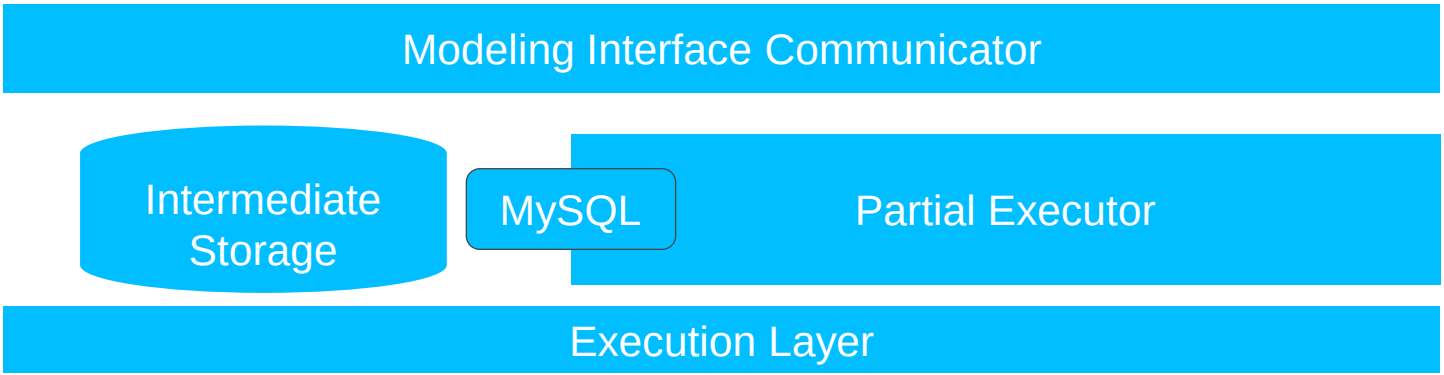
Simple Example



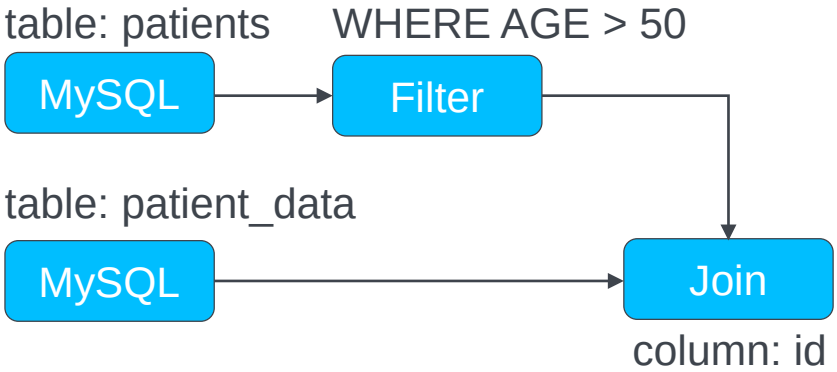


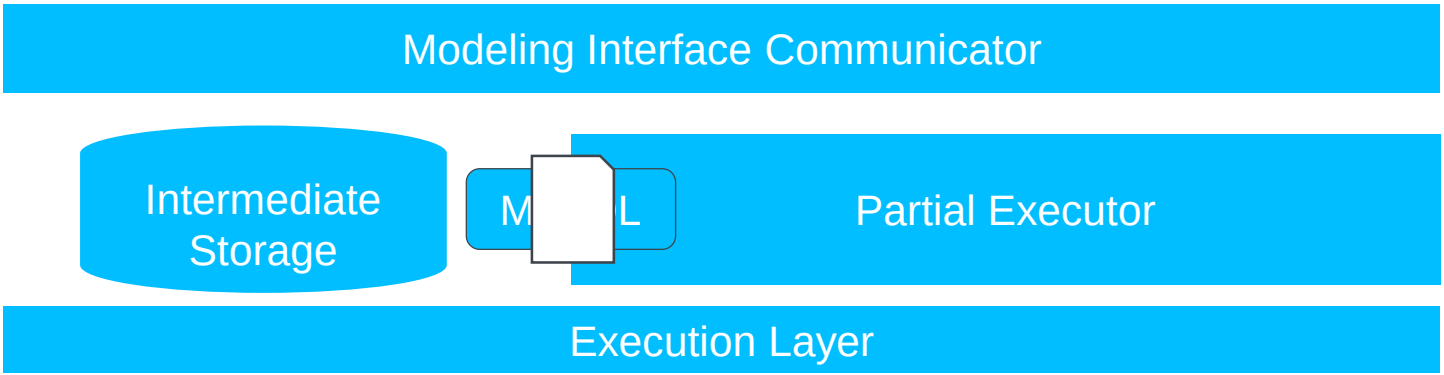
Simple Example



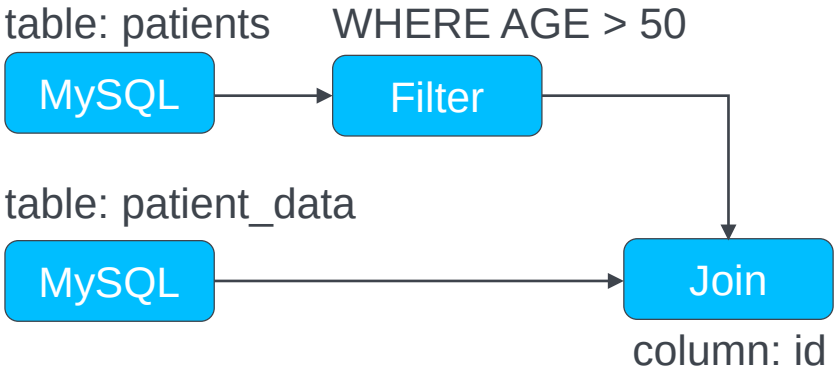


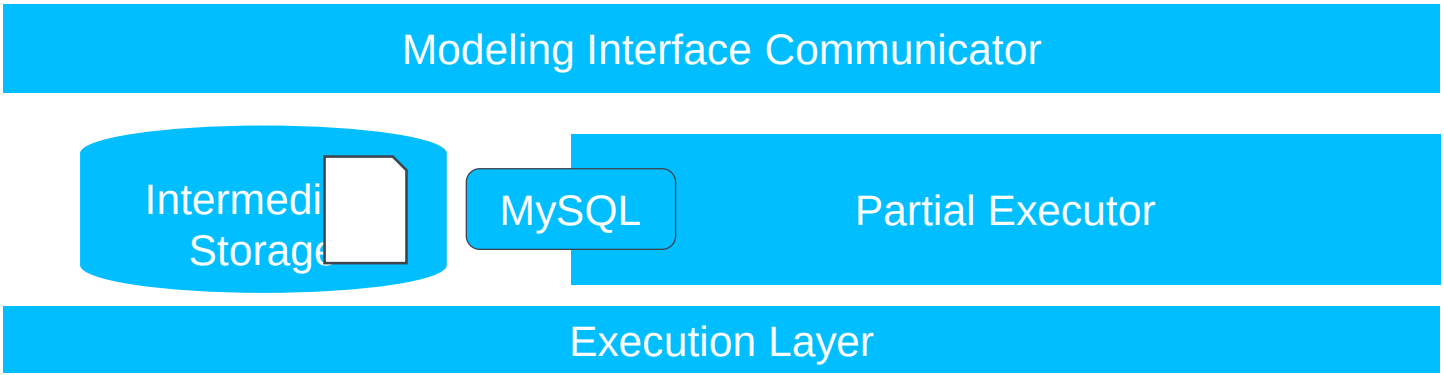
Simple Example



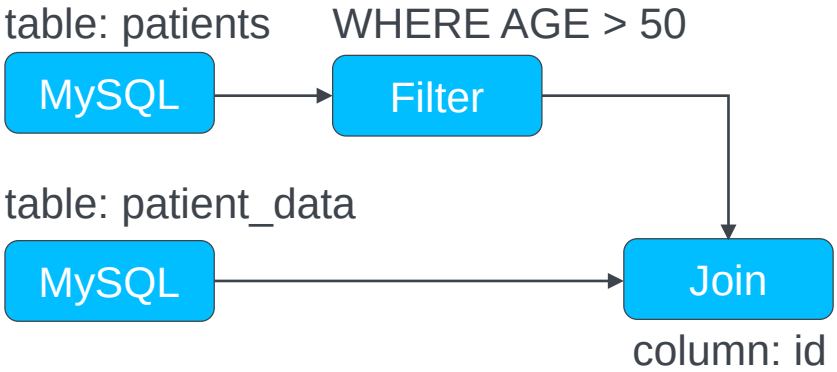


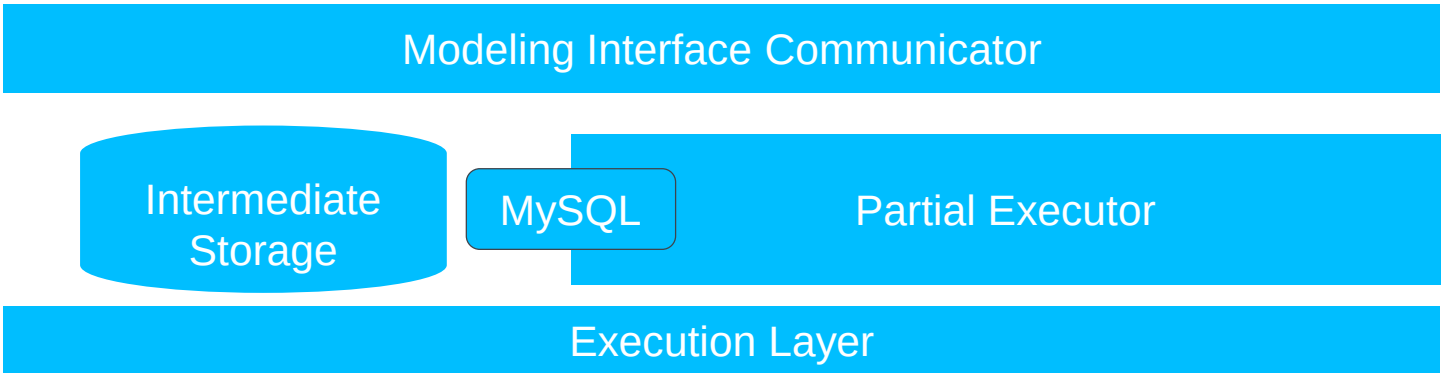
Simple Example



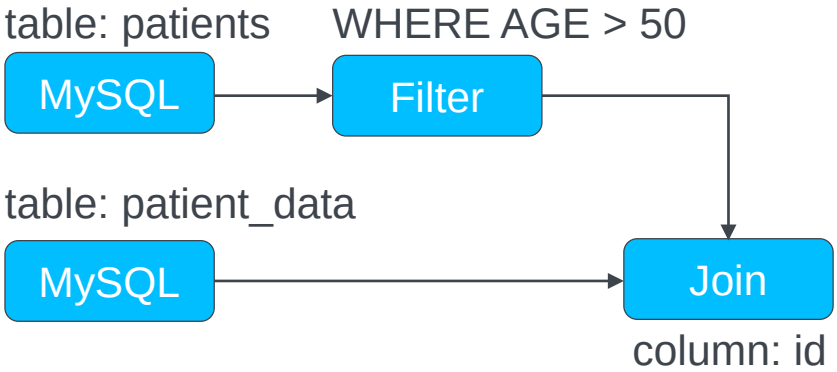


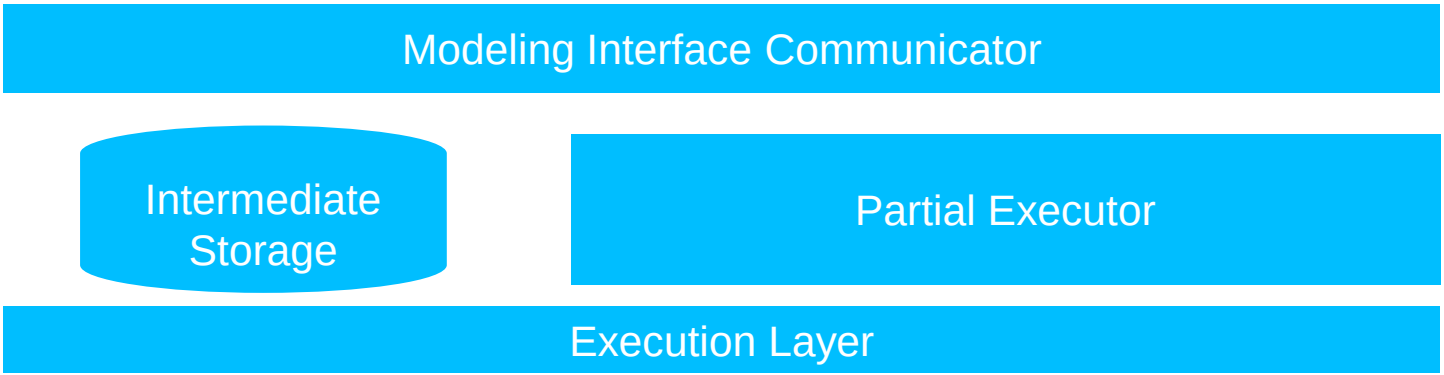
Simple Example



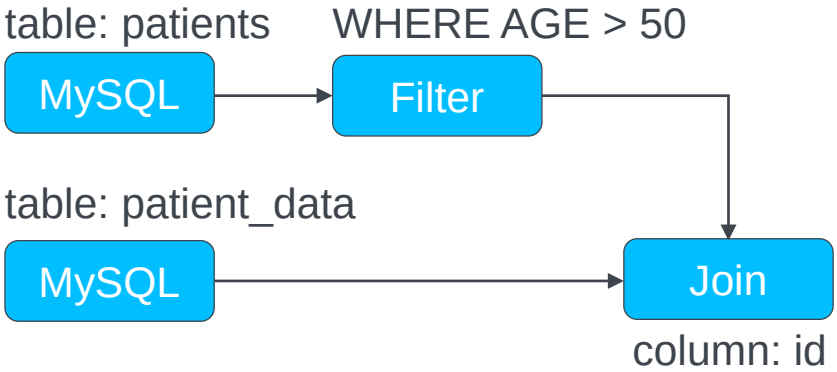


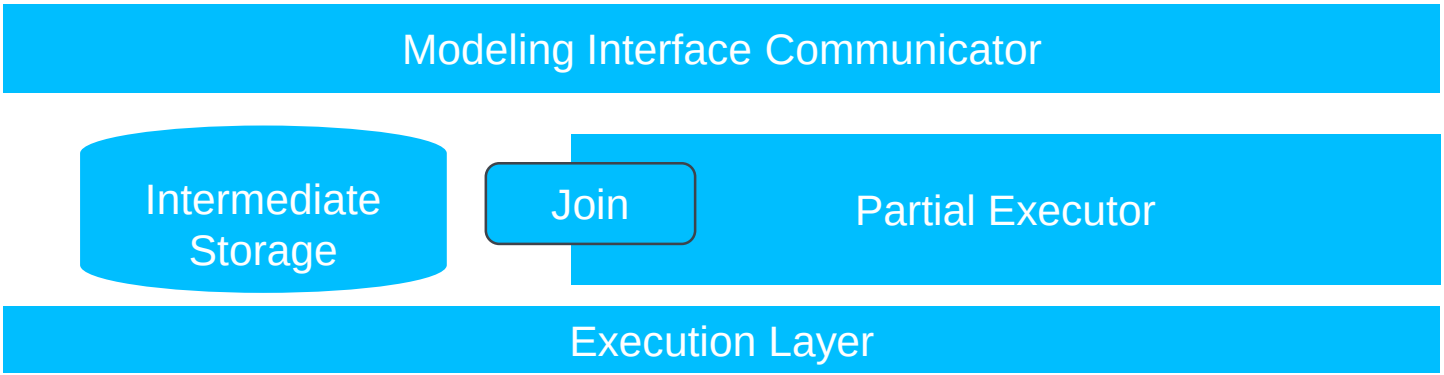
Simple Example



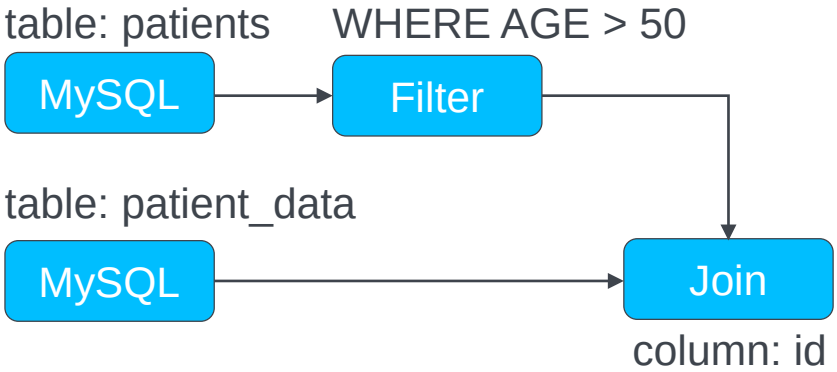


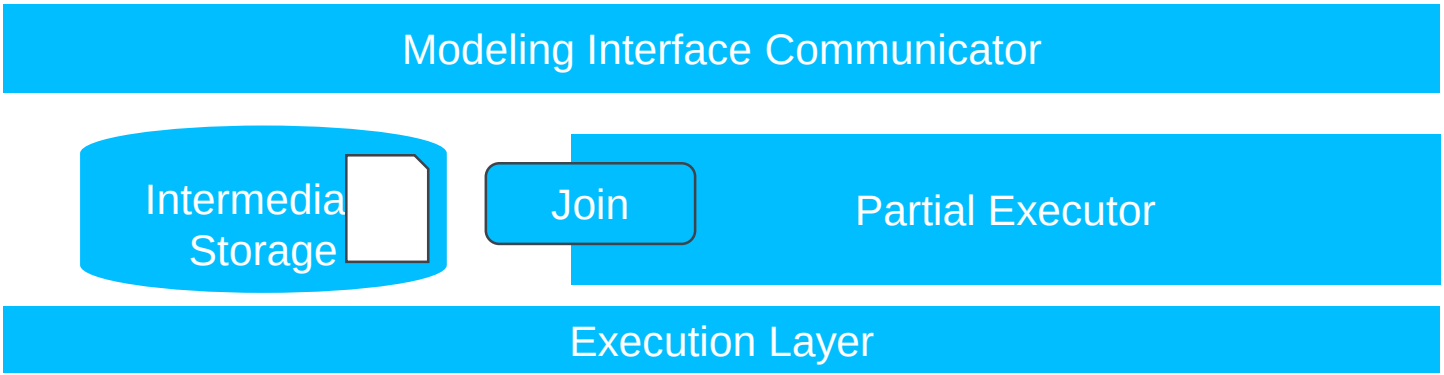
Simple Example



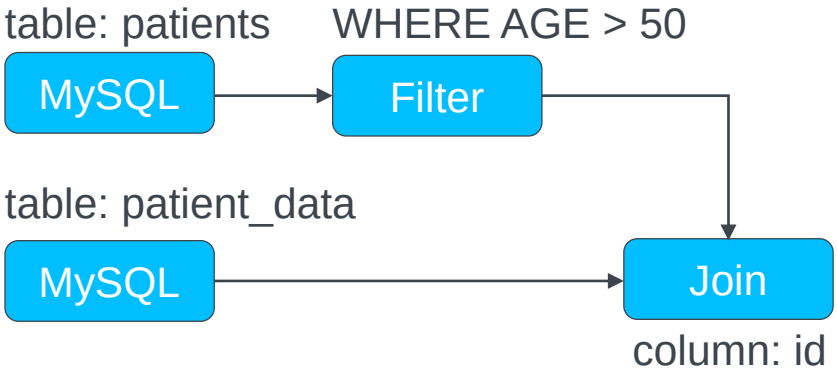


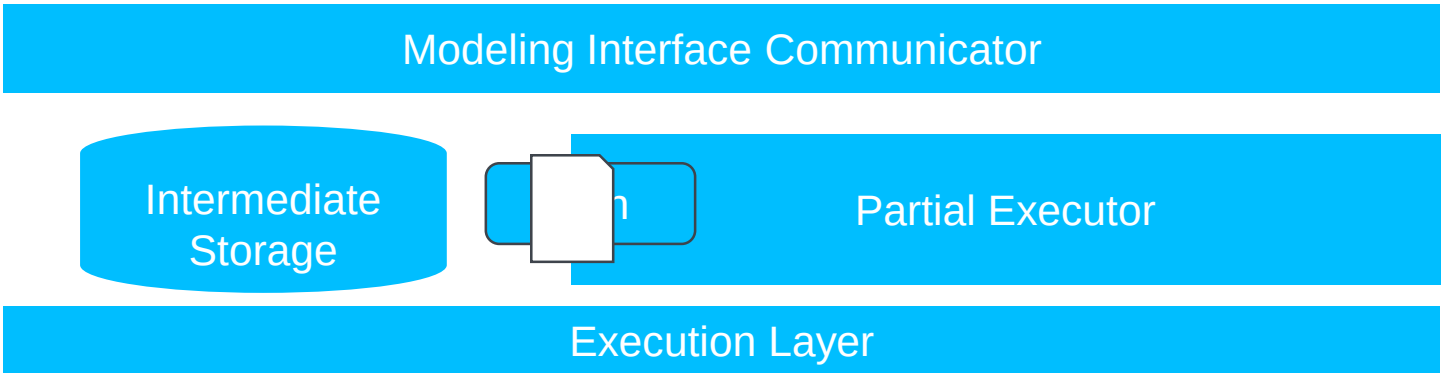
Simple Example



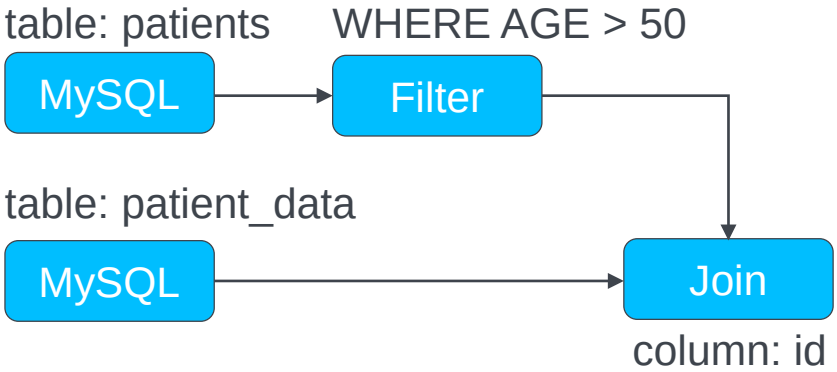


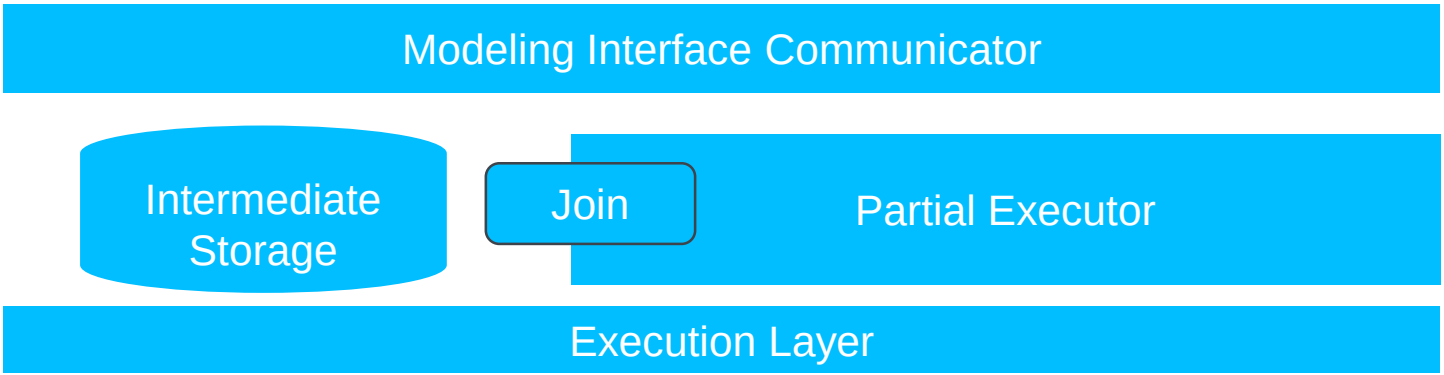
Simple Example



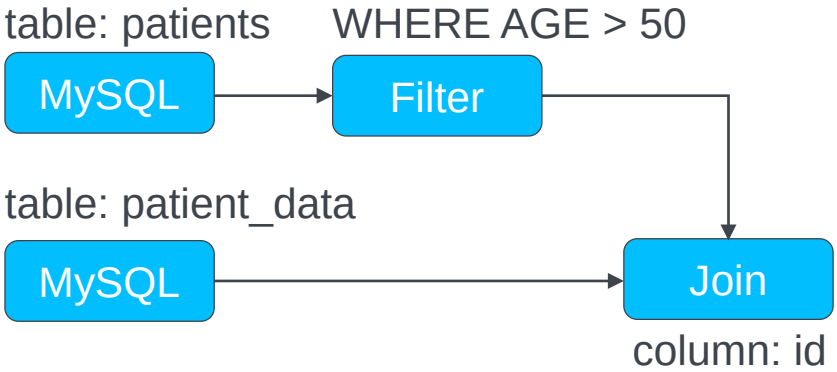


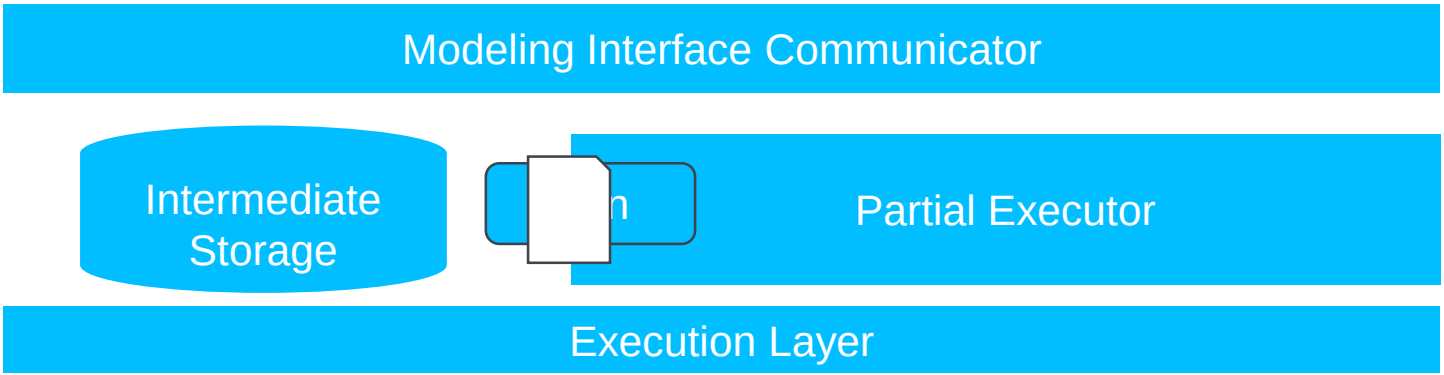
Simple Example



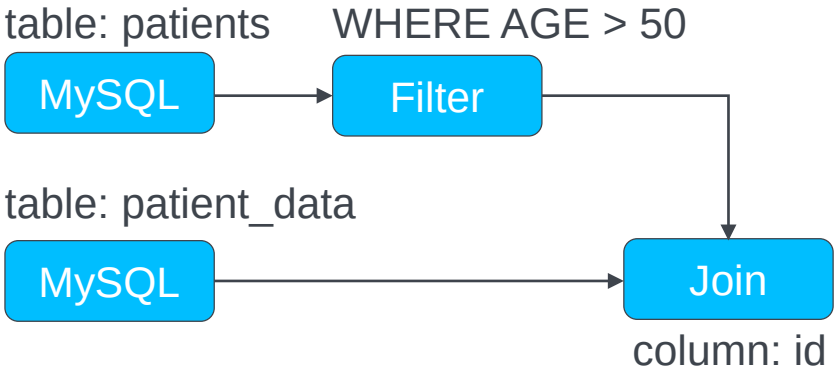


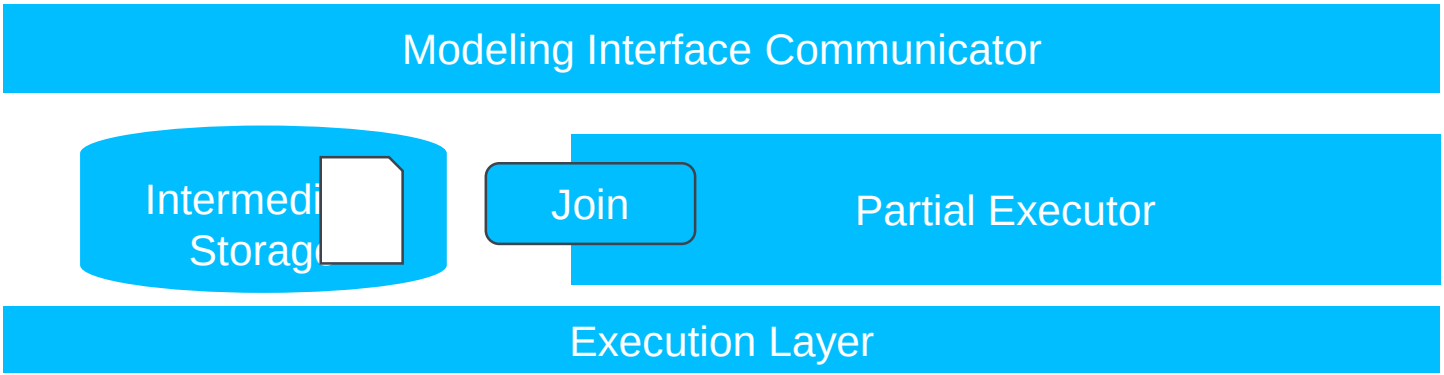
Simple Example



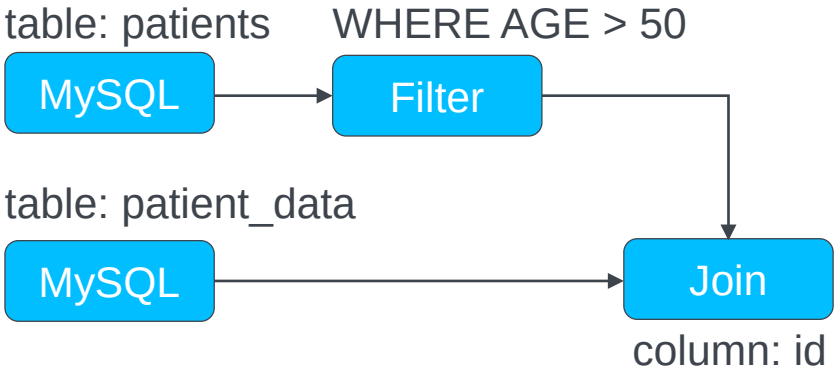


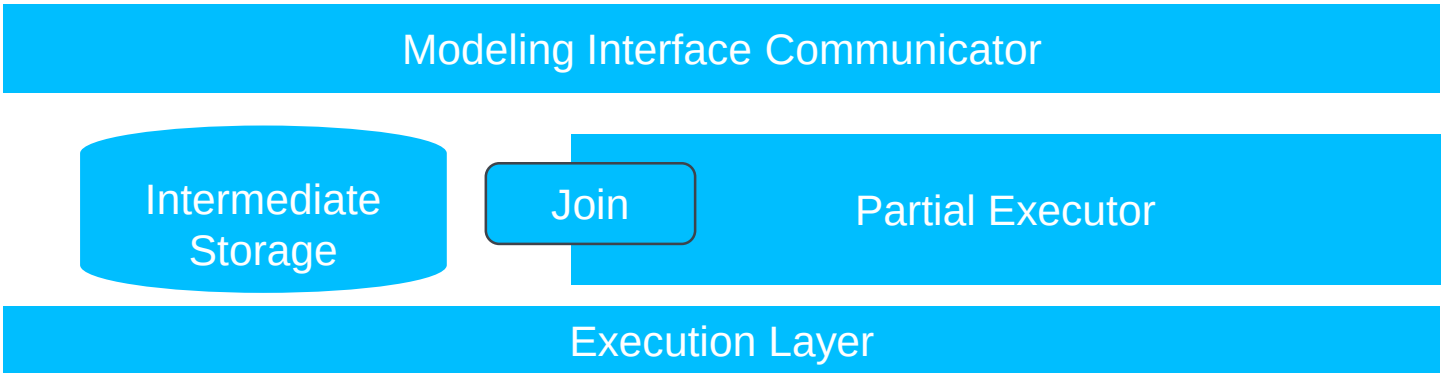
Simple Example



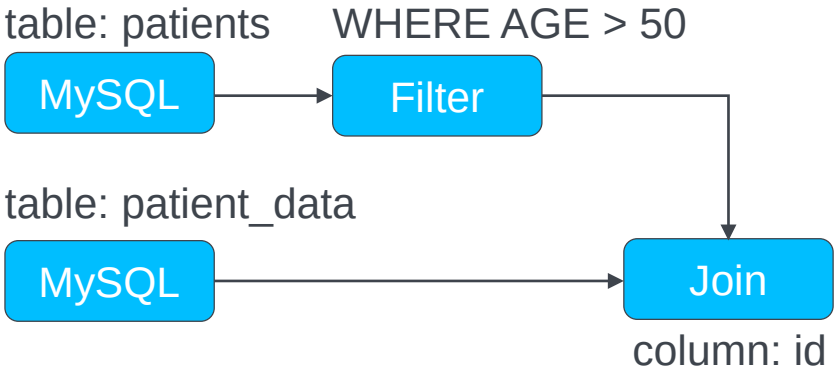


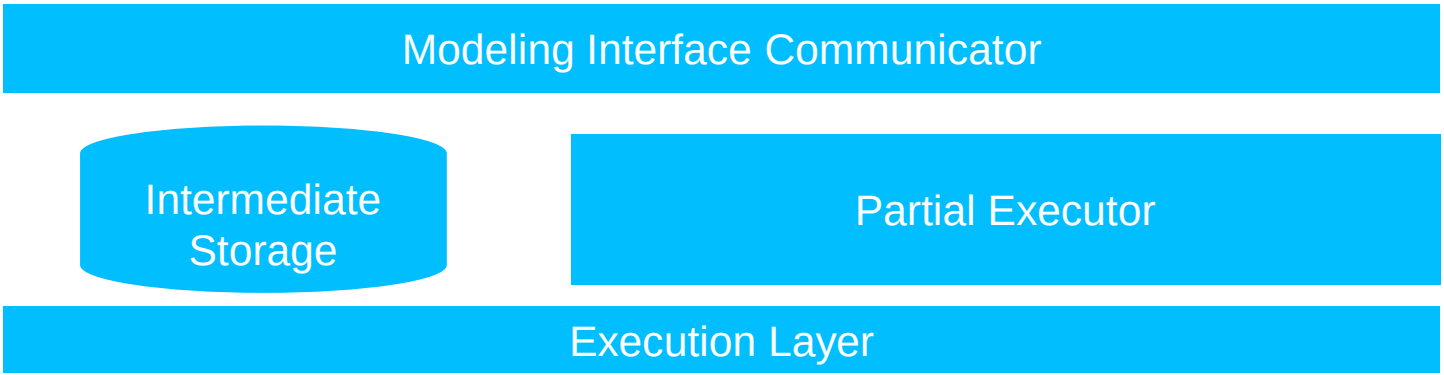
Simple Example



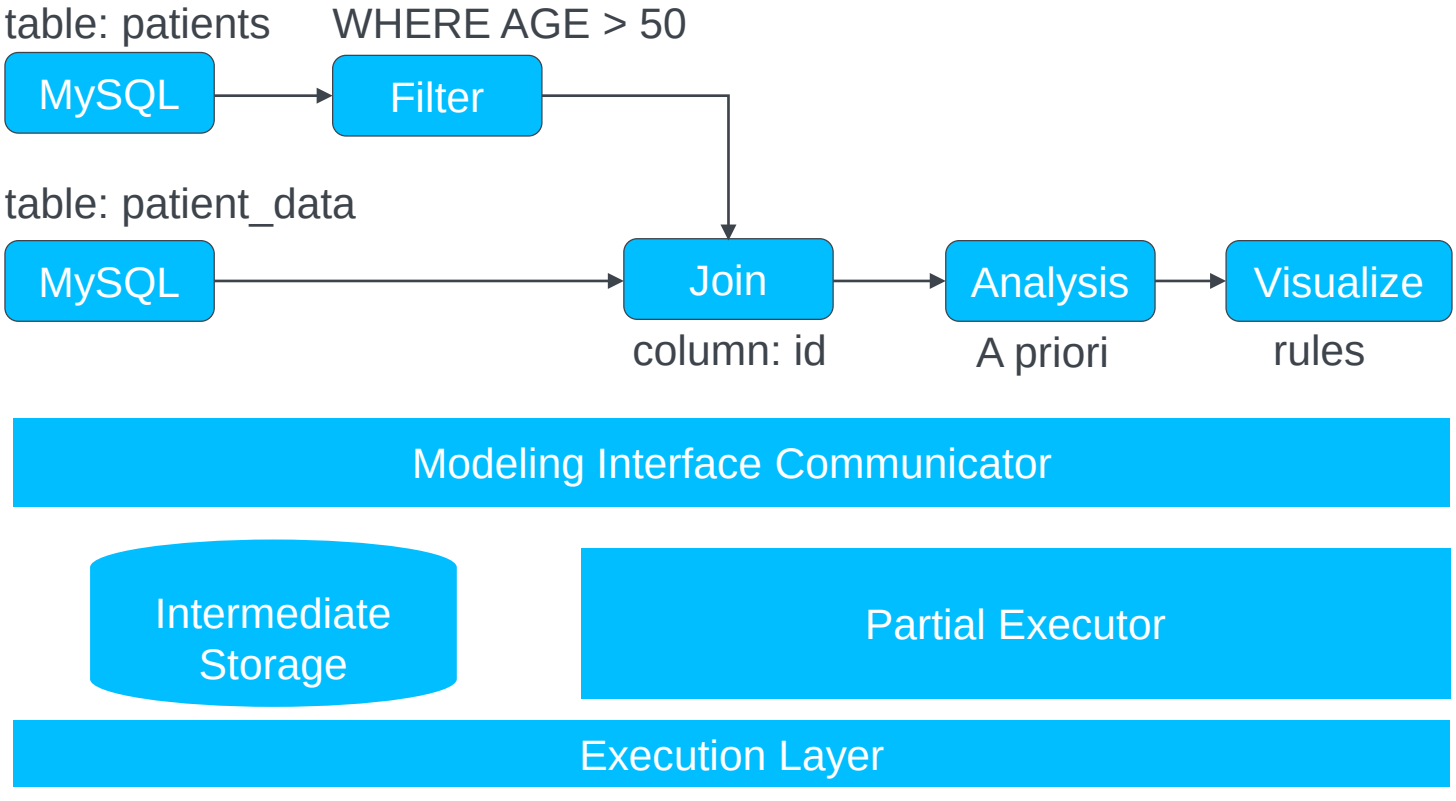


Simple Example

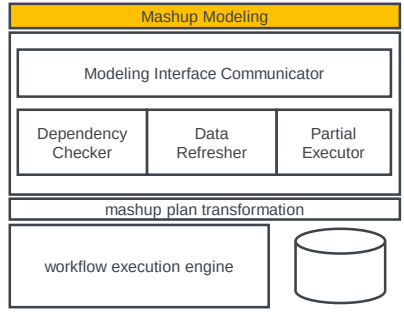




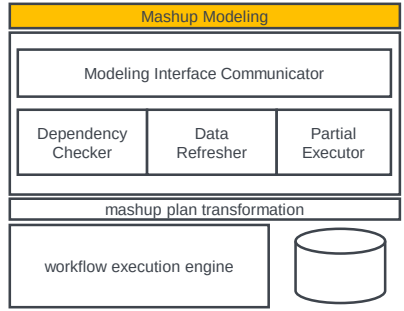
Simple Example



FlexMash Modeling Layer Adjustments



FlexMash Modeling Layer Adjustments



- Current state of execution

(a) recently placed without interaction

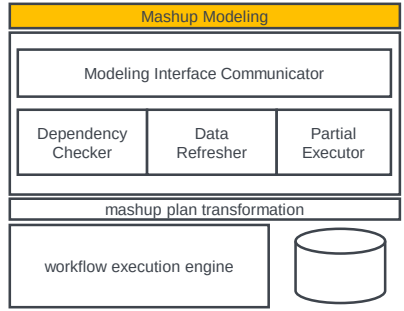
(b) not properly configured, outdated or error during execution

(c) ready to process or currently in execution

(d) 30 percent completed

(e) completed

FlexMash Modeling Layer Adjustments



- Current state of execution

(a) recently placed without interaction

(b) not properly configured, outdated or error during execution

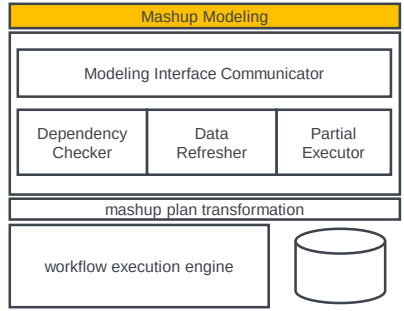
(c) ready to process or currently in execution

(d) 30 percent completed

(e) completed

- Showing intermediate results during model time (raw data)

FlexMash Modeling Layer Adjustments



- Current state of execution

(a) recently placed without interaction

(b) not properly configured, outdated or error during execution

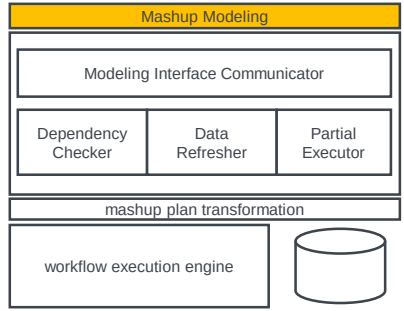
(c) ready to process or currently in execution

(d) 30 percent completed

(e) completed

- Showing intermediate results during model time (raw data)
- Implicit execution instead of manual triggering

FlexMash Modeling Layer Adjustments



- Current state of execution

(a) recently placed without interaction

(b) not properly configured, outdated or error during execution

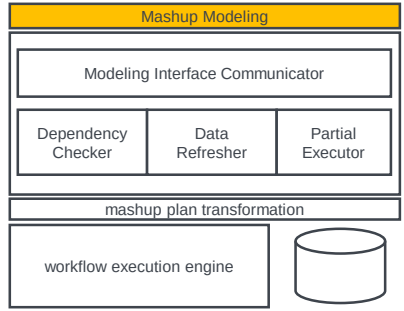
(c) ready to process or currently in execution

(d) 30 percent completed

(e) completed

- Showing intermediate results during model time (raw data)
- Implicit execution instead of manual triggering
- Means to define lifetime of data in the data source nodes of the plan

FlexMash Modeling Layer Adjustments



- Current state of execution

(a) recently placed without interaction

(b) not properly configured, outdated or error during execution

(c) ready to process or currently in execution

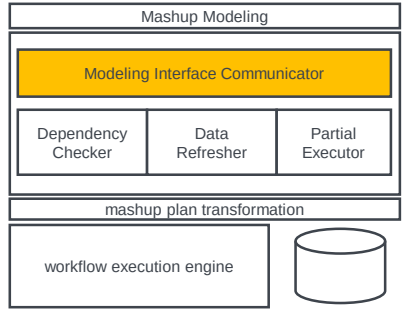
(d) 30 percent completed

(e) completed

- Showing intermediate results during model time (raw data)
- Implicit execution instead of manual triggering
- Means to define lifetime of data in the data source nodes of the plan

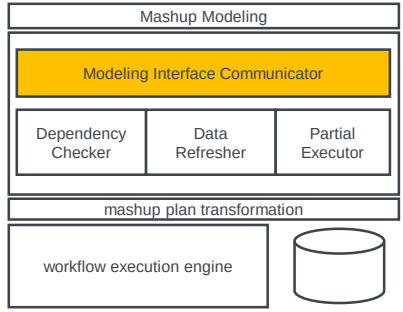
- State of execution is stored in the Partial Executor component

Modeling Interface Communicator

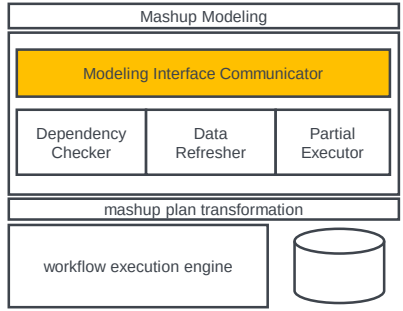


Modeling Interface Communicator

- Interface to the modeling layer

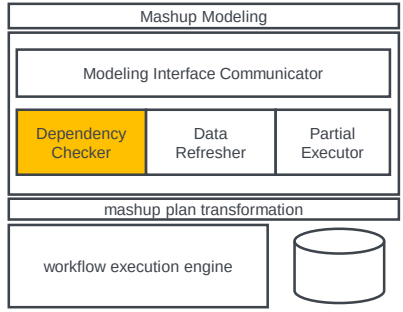


Modeling Interface Communicator

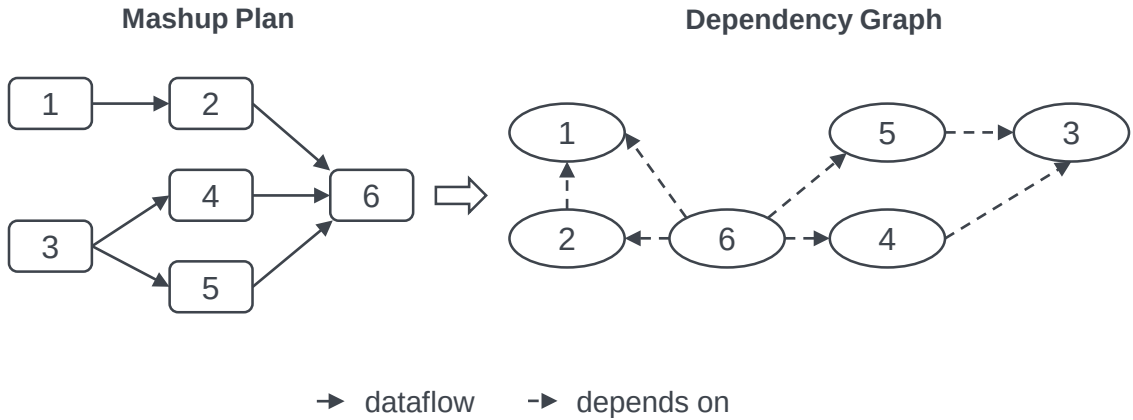


- Interface to the modeling layer
- Handles all the communication with the modeling layer
  - Delivers intermediate results
  - Offers information about the current state of execution
  - Coordinates all components

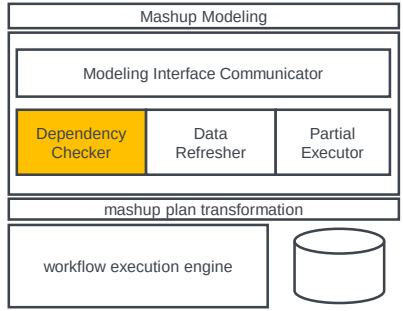
Dependency Checks



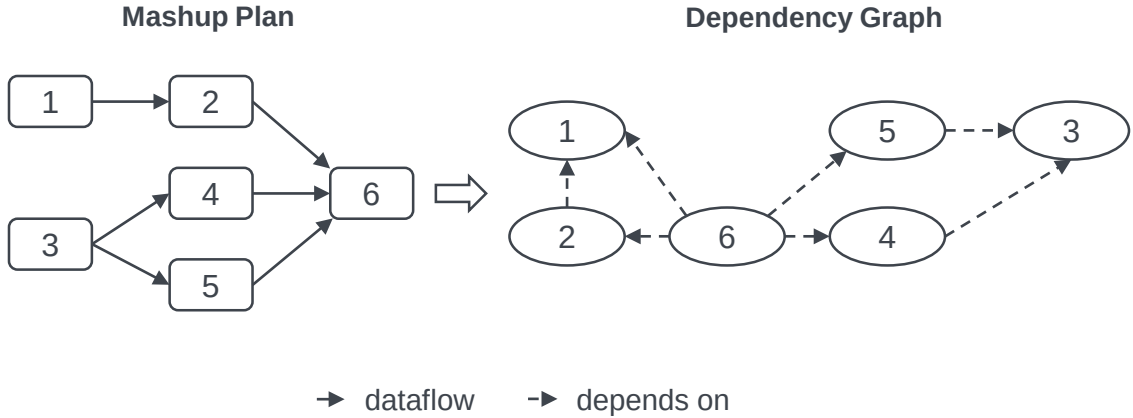
- Dependencies checked based on dependency graphs



Dependency Checks

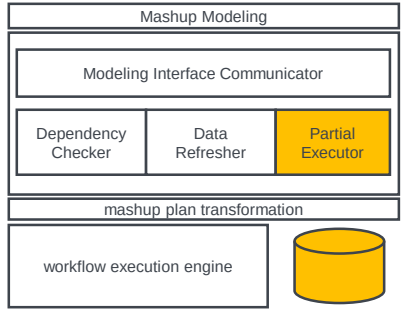


- Dependencies checked based on dependency graphs

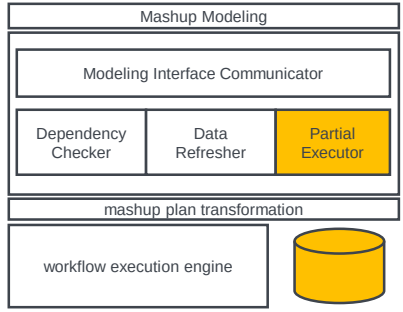


- Mashup Plan is transformed into dependency graph
- Constant updates of the graph necessary with changing Mashup Plans
- Dependency graph indicates which nodes can be executed

Partial Executor

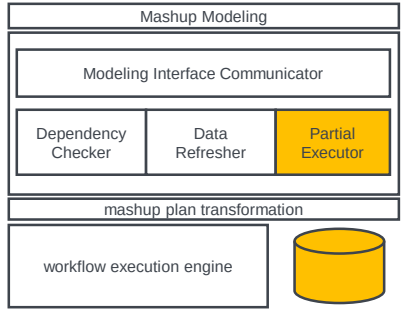


Partial Executor



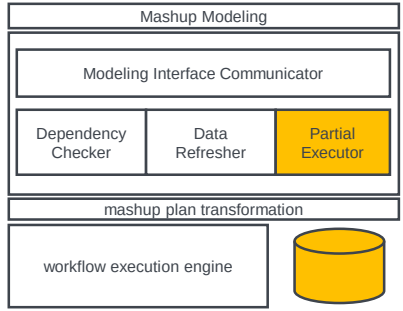
- Stores "twin" model of the Mashup Plan

Partial Executor



- Stores "twin" model of the Mashup Plan
  - Nodes, edges and current state of execution
  - Contains information about available intermediate results

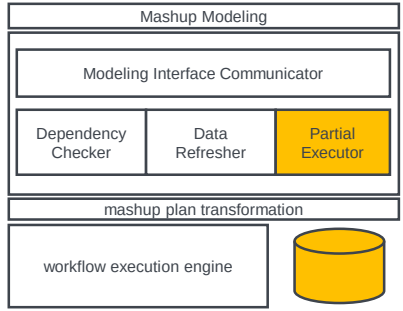
Partial Executor



- Stores "twin" model of the Mashup Plan
  - Nodes, edges and current state of execution
  - Contains information about available intermediate results

- Handles the partial execution

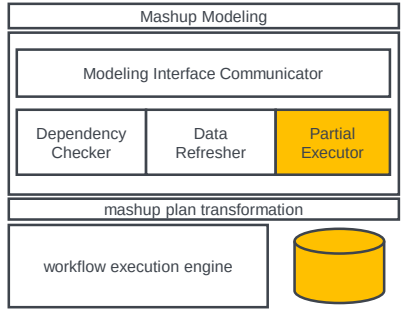
Partial Executor



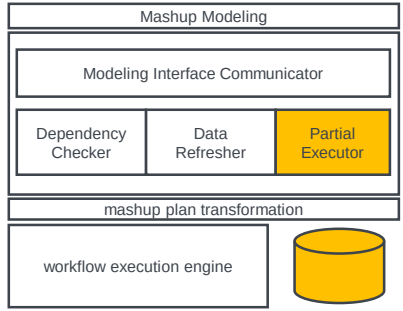
- Stores "twin" model of the Mashup Plan
  - Nodes, edges and current state of execution
  - Contains information about available intermediate results

- Handles the partial execution
  - Execution triggered by Dependency Checker based on dependency graphs
  - Nodes without dependencies handed to transformation
  - Single nodes are executed in the workflow engine
  - Available intermediate results are used as input for execution

Partial Executor

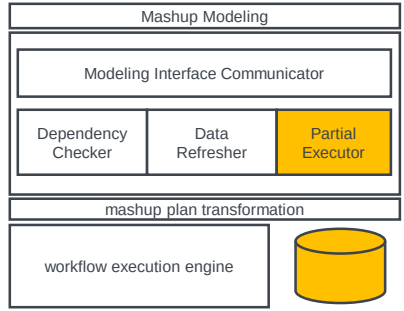


Partial Executor



- Results
  - stored into an intermediate storage
  - handed to the front end to display intermediate results
  - State of execution is stored and provided to the front end

Partial Executor

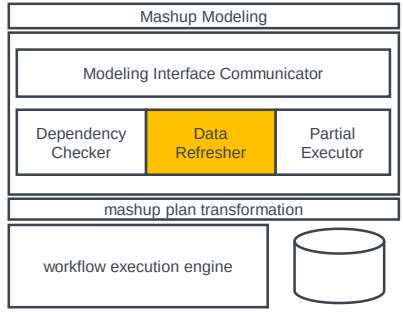


- Results
  - stored into an intermediate storage
  - handed to the front end to display intermediate results
  - State of execution is stored and provided to the front end

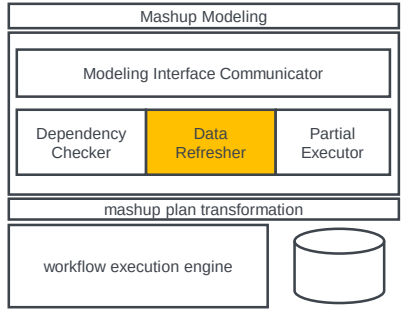
- Unsolved issue: How to cope with re-execution of currently executed (long-running) services?
  - Abort service execution and start over ➔ results are lost / costly ☹
  - Use current execution ➔ wrong results and/or stale data ☹

  ➔ Future work ☺

Data Refresher

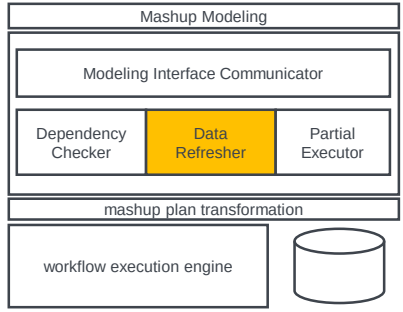


Data Refresher



- Intermediate data becomes stale over time

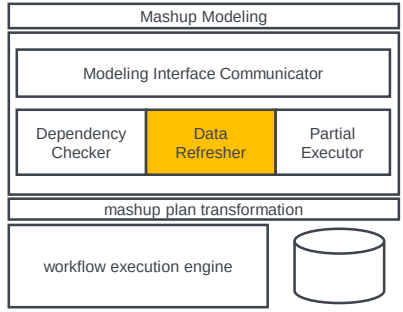
Data Refresher



- Intermediate data becomes stale over time

- We need a means to define the lifetime of data!
  - Example:
    - High frequent sensor data = low lifetime
    - Non-frequent events = high lifetime

Data Refresher



- Intermediate data becomes stale over time
- We need a means to define the lifetime of data!
  - Example:
    - High frequent sensor data = low lifetime
    - Non-frequent events = high lifetime
- Lifetime needs to be defined at the data sources on modeling level
  - Requires expert knowledge
- Data refresher removes stale data and triggers re-execution of services

Validation

Requirement Validation

Requirement Validation

(R1) Definition of data expiration time

➔ experts can define the life time in the data source nodes

Requirement Validation

(R1) Definition of data expiration time

➔ experts can define the life time in the data source nodes

(R2) Reduction of staleness of data

➔ constant refreshing of when lifetime expires

Requirement Validation

(R1) Definition of data expiration time

➔ experts can define the life time in the data source nodes

(R2) Reduction of staleness of data

➔ constant refreshing of when lifetime expires

(R3) Dependency checks

➔ solved through dependency graphs

Requirement Validation

(R1) Definition of data expiration time

➔ experts can define the life time in the data source nodes

(R2) Reduction of staleness of data

➔ constant refreshing of when lifetime expires

(R3) Dependency checks

➔ solved through dependency graphs

(R4) Split Mashup Plans into fragments

➔ done (easy ☺)

Requirement Validation

(R1) Definition of data expiration time

➔ experts can define the life time in the data source nodes

(R2) Reduction of staleness of data

➔ constant refreshing of when lifetime expires

(R3) Dependency checks

➔ solved through dependency graphs

(R4) Split Mashup Plans into fragments

➔ done (easy ☺)

(R5) Persistent storage of intermediate results

➔ solved by intermediate storage

Requirement Validation

(R1) Definition of data expiration time

➔ experts can define the life time in the data source nodes

(R2) Reduction of staleness of data

➔ constant refreshing of when lifetime expires

(R3) Dependency checks

➔ solved through dependency graphs

(R4) Split Mashup Plans into fragments

➔ done (easy ☺)

(R5) Persistent storage of intermediate results

➔ solved by intermediate storage

(R6) User feedback

➔ provided through the user interface

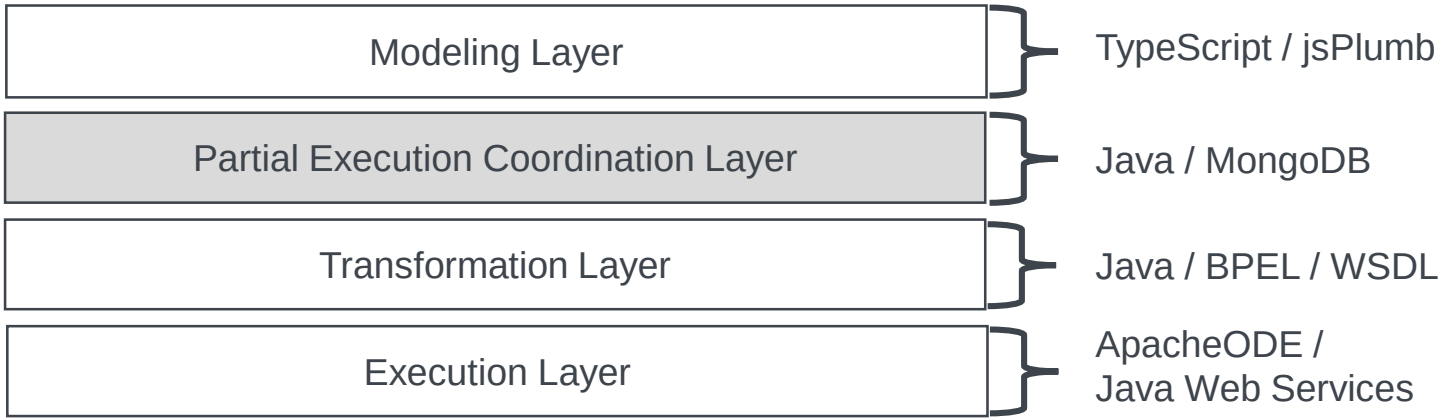
Implementation

Implementation

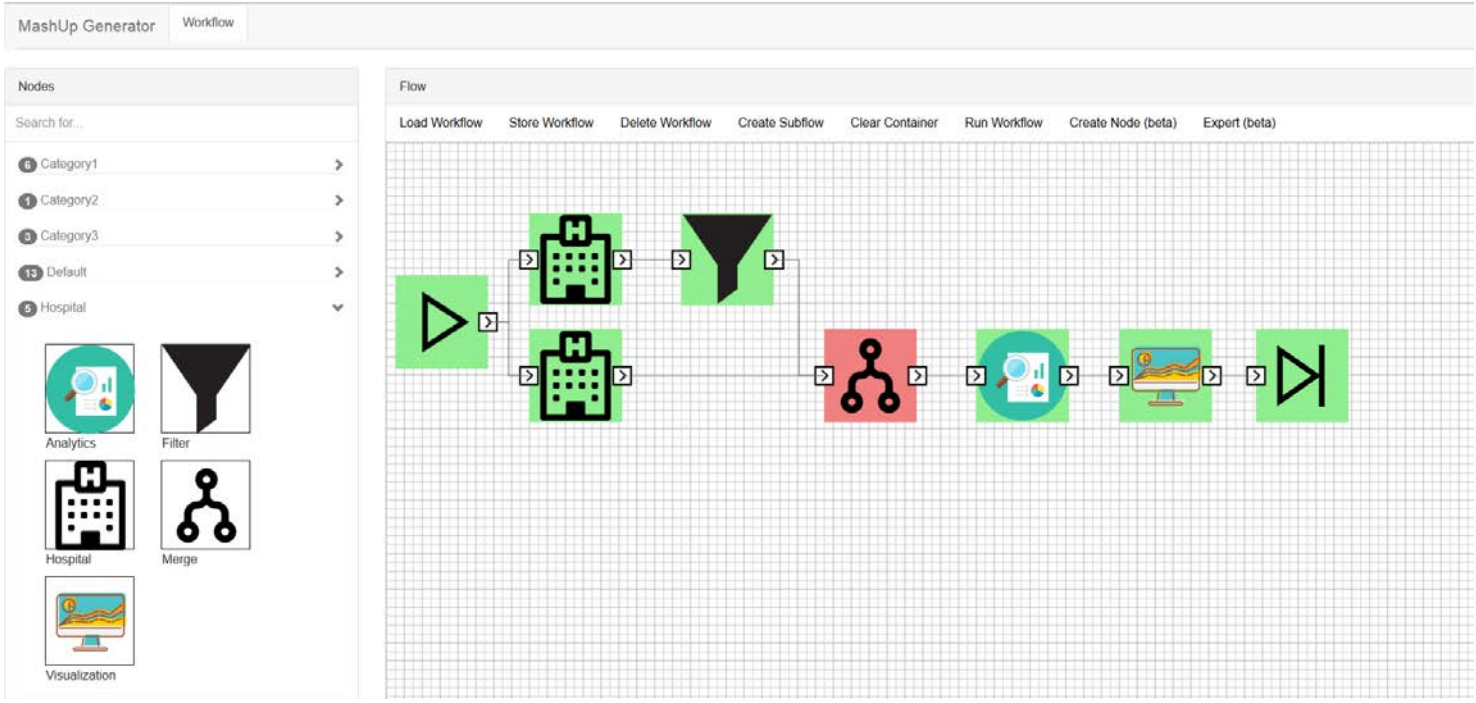
- Working prototype of the introduced architecture

Implementation

- Working prototype of the introduced architecture
- Technologies used:



FlexMash – New UI



Conclusion and Future Work

Conclusion and Future Work

Conclusion and Future Work

- We enabled partial execution of Mashup Plans
- Enables showing intermediate results
- "Felt" faster runtime of the Mashups

Conclusion and Future Work

- We enabled partial execution of Mashup Plans
- Enables showing intermediate results
- "Felt" faster runtime of the Mashups

- Future work
  - Execution of subgraphs instead of single nodes
  - User studies

CoMoRea´18

14th Workshop on Context and Activity Modeling and Recognition

- Workshop at the IEEE International Conference on Pervasive Computing (Percom)
- Focus of the workshop
  - advance the state of the art in context modeling and reasoning
  - discuss fundamental issues in context processing and management
  - identify concepts, theories and methods applicable to context modeling and context reasoning

- **Where?** Athens
- **When?** March 19th to March 23rd

- **Workshop paper submission deadline: Nov 11, 2017**

CoMoRea´18

14th Workshop on Context and Activity Modeling and Recognition

- Organizing Committee
  - Heiner Stuckenschmidt, Universität Mannheim, Germany
  - Matthias Wieland, Universität Stuttgart, Germany
  - Pascal Hirmer, Universität Stuttgart, Germany
- Advisory Board
  - Christian Becker, Universität Mannheim, Germany
  - Jadwiga Indulska, The University of Queensland, Australia
  - Daniela Nicklas, University of Bamberg, Germany
  - Daniele Riboni, University of Cagliari, Italy

Topics of interest

- Context modeling techniques and domain-specific context models
- Ontology-based approaches to context modeling and reasoning
- Ontologies of Activities and Context
- Hybrid context models and advanced issues in context modeling, including issues of information quality, ambiguity, and provenance
- Context reasoning algorithms, their complexity and accuracy
- Discovery, reuse, privacy, security and trust of context information
- Distributed and scalable context management
- Tool support for context modeling and development of context model-based applications
- Machine learning and reasoning techniques for context and activity recognition
- High Level Activity Recognition from Sensor Data
- Machine Learning and Computer Vision for Context and Activity Recognition
- Reference Datasets and Benchmarks for Activity Recognition and Context Reasoning

Universität Stuttgart

Vielen Dank!

Dipl.-Inf. Pascal Hirmer

E-Mail Pascal.Hirmer@ipvs.uni-stuttgart.de
Telefon +49 (0) 711 685-88297

Universität Stuttgart
Institute of Parallel and Distributed Systems
Universitätsstraße 38, 70569 Stuttgart