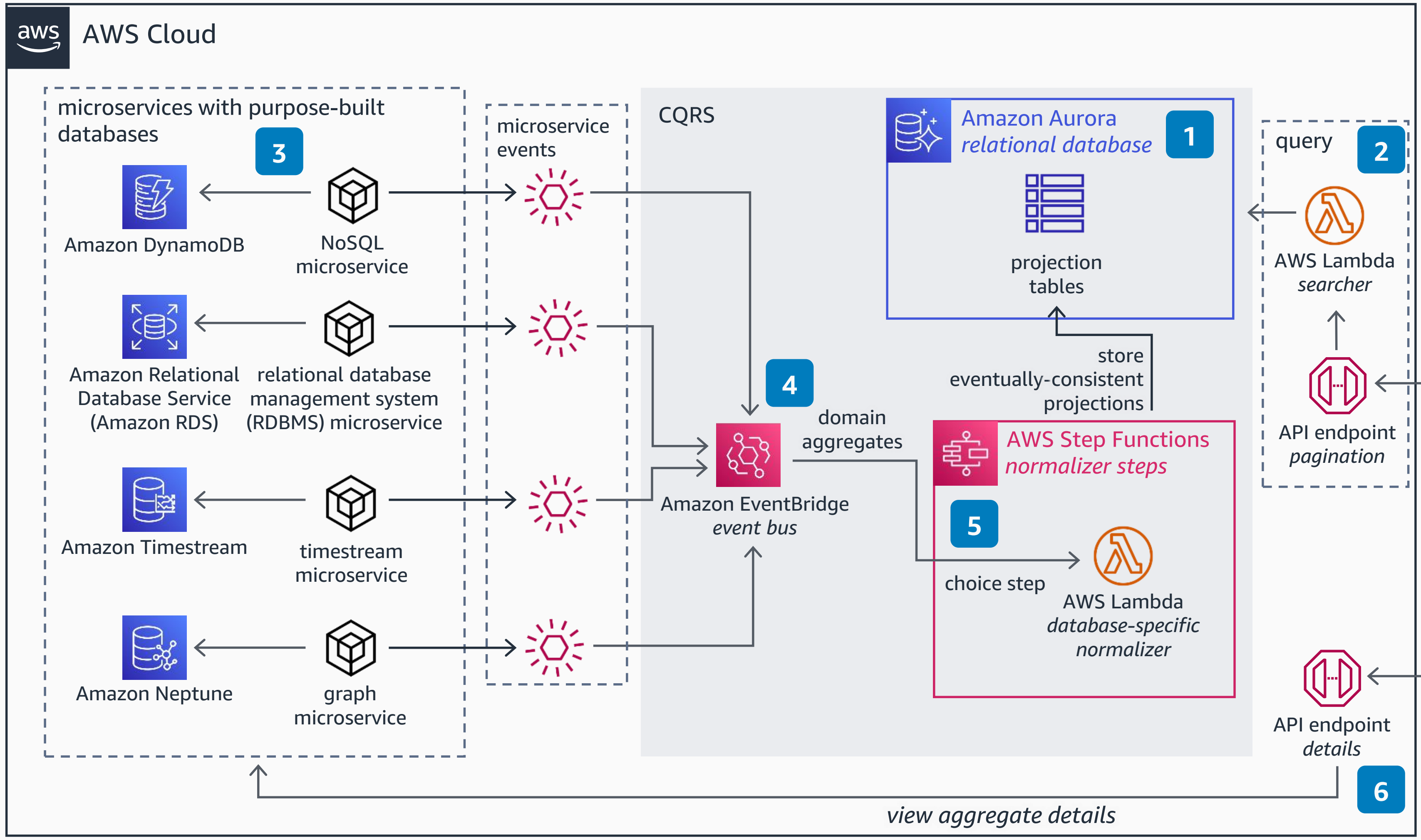
Paginated Search with Purpose Built Databases

This reference architecture shows how create a searchable, paginated list on domain aggregates from purpose-built databases with a step transformation and SQL queries. Apply the command query responsibility segregation (CQRS) pattern to an event-driven microservice architecture to enable searching, pagination, and sorting by querying eventually-consistent data projections in Amazon Aurora.


AWS Cloud
microservices with purpose-built databases
3
Amazon DynamoDB
NoSQL microservice
Amazon Relational Database Service (Amazon RDS)
relational database management system (RDBMS) microservice
Amazon Timestream
timestream microservice
Amazon Neptune
graph microservice
microservice events
CQRS
Amazon Aurora relational database
1
projection tables
store eventually-consistent projections
4
domain aggregates
Amazon EventBridge event bus
AWS Step Functions normalizer steps
5
choice step
AWS Lambda database-specific normalizer
query
2
AWS Lambda searcher
API endpoint pagination
clients
API endpoint details
6
view aggregate details

1. Create a relational database using **Amazon Aurora** to hold projections of the domain aggregates you need to search, paginate, and sort. Working backwards from the user interfaces, decide how to design the tables and columns.

2. Create an **AWS Lambda** function to invoke a SQL query against the relational database, then connect it to your pagination API endpoint to handle search requests. The function uses a 'Where' clause for searching, 'OrderBy' for sorting, and 'Limit, Offset and Count' for pagination. The paginated results include the unique keys of the returned domain aggregates.

3. Decoupled microservices raise events for changes in the domain aggregates they own. Each microservice selects the purpose-built database that suits their use case, while eventually keeping the projection database up to date consistently.

4. Use **Amazon EventBridge** to create an event bus to collect the domain events from your microservices.

5. Normalize domain aggregates with database specific normalizers to match the projection tables in the database and calculate additional analytics. To manage the workflow and direct domain aggregates to the right normalizer, create a choice step within **AWS Step Functions.** For complex normalizations, the state machine can integrate and wait for long running containers on **Amazon ECS.**

6. Clients can request details on a returned aggregate, referring to it by a unique key. This is managed by the relevant microservice through the API endpoint.

Reviewed for technical accuracy August 25, 2022

AWS Reference Architecture