How MySQL REST Service ends the middleware tax

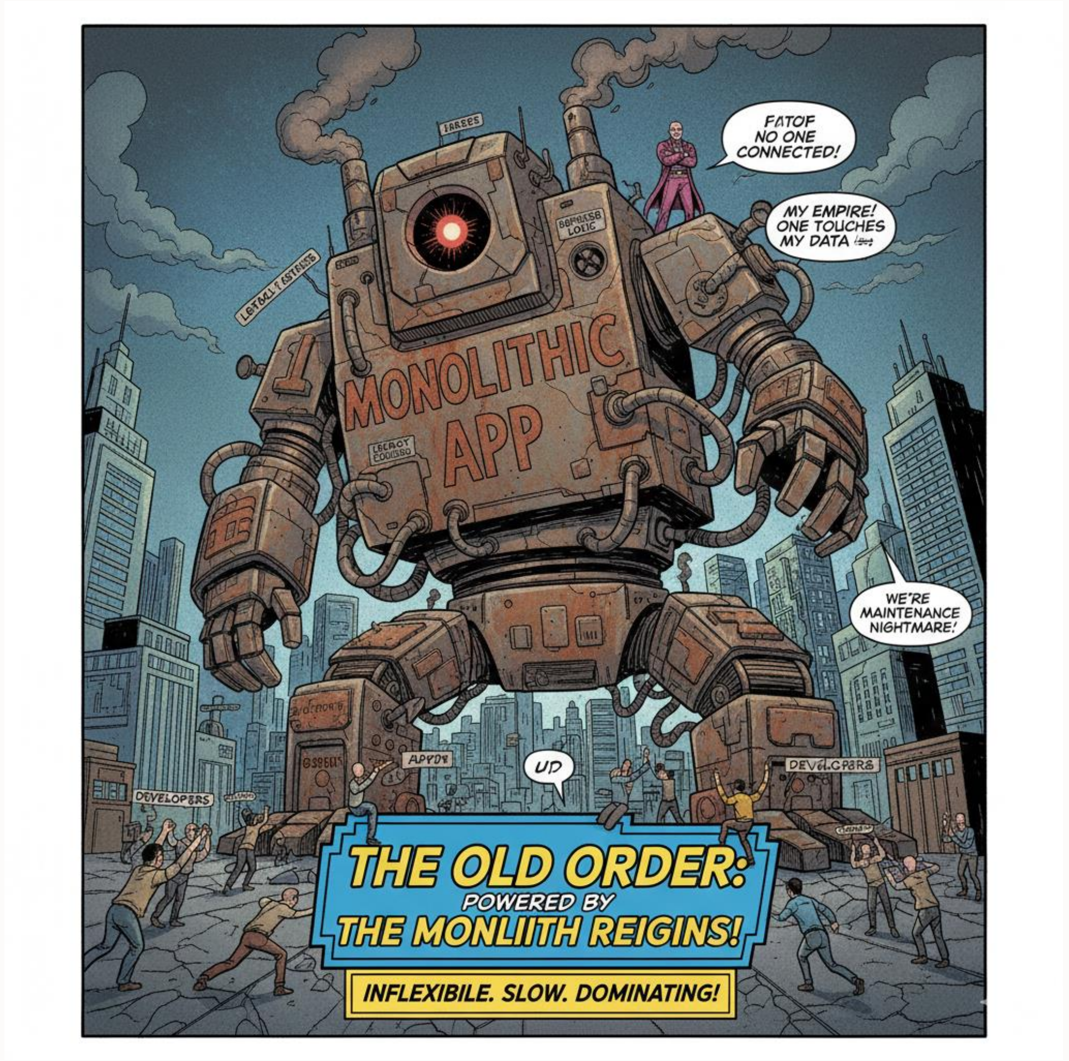
Vittorio Cioe
MySQL Solutions Engineer
vittorio.cioe@oracle.com

Agenda

- The middleware “tax” (…or ”dependancy”)
- MySQL REST Service joins the party
- Eliminating the “tax”: key functionalities
- Summary and Q&A

The middleware “tax” (…or ”dependency”)

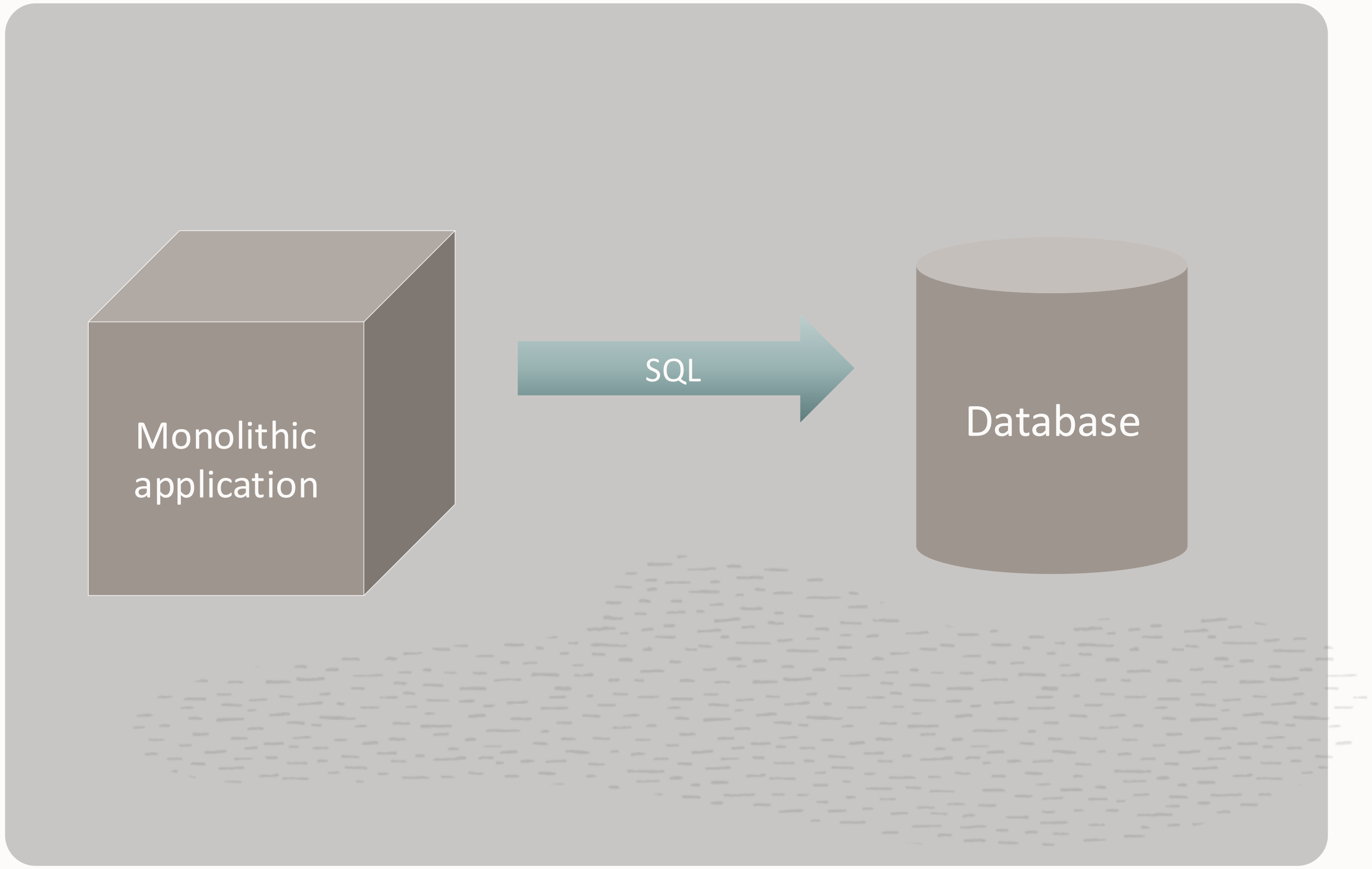
The digital transformation story (1/2)


FATOF NO ONE CONNECTED!
MY EMPIRE! ONE TOUCHES MY DATA
MONOLITHIC APP
WE'RE MAINTENANCE NIGHTMARE!
UD
DEVELOPERS
THE OLD ORDER: POWERED BY THE MONLIITH REIGINS!
INFLEXIBILE. SLOW. DOMINATING!

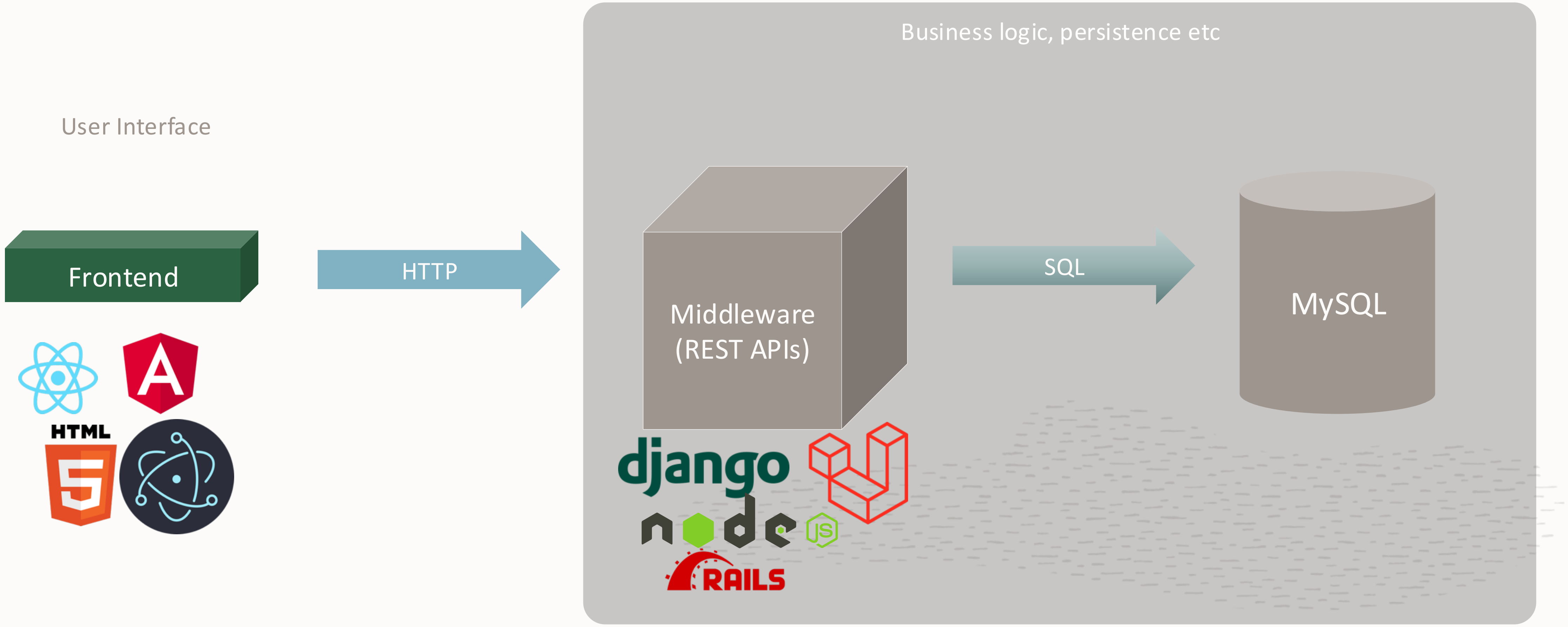
The digital transformation story (2/2)

…and now let’s see the real story…

- Technology evolution
- New development trends
- Faster internet
- Mobile application
- Companies opening up data
- Application modernization need
- From monolithic to multi tier

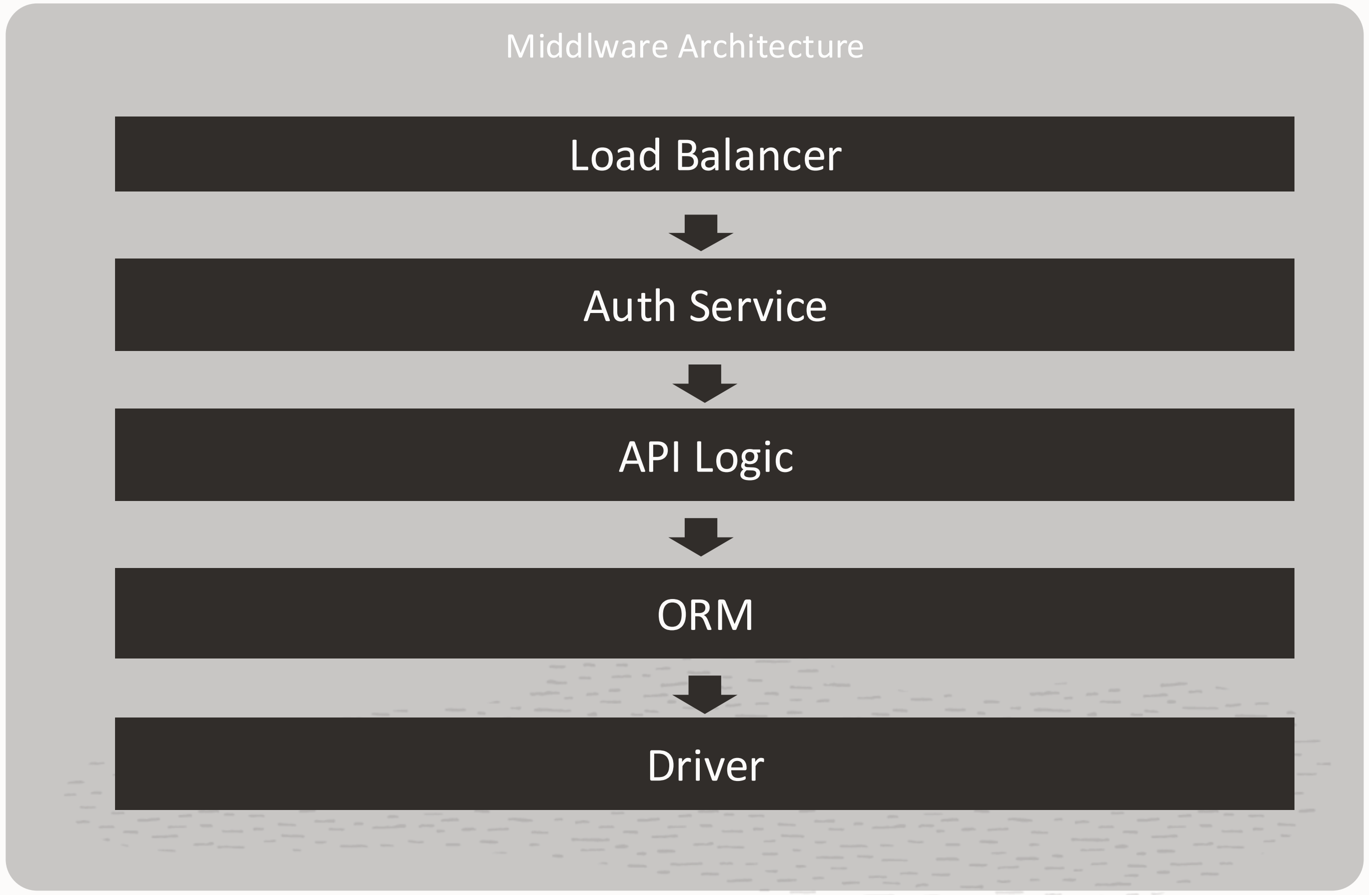


Multi Tier Application Architecture: REST APIs as digital transformation enabler



The middleware tax (or middleware dependency)

- **Development:** Writing the same CRUD for every new table.
- **Latency:** The "Double Serialization" penalty—converting DB rows to Objects, then Objects to JSON.
- **Synchronization:** Keeping your Backend Models, Frontend Types, and Database Schema in sync
- **Maintenance:** Patching the runtime, updating libraries, and managing container orchestration for a layer that mostly just “passes data through.”
- **Dependency:** Adjust again all of the above every time a middleware is changed



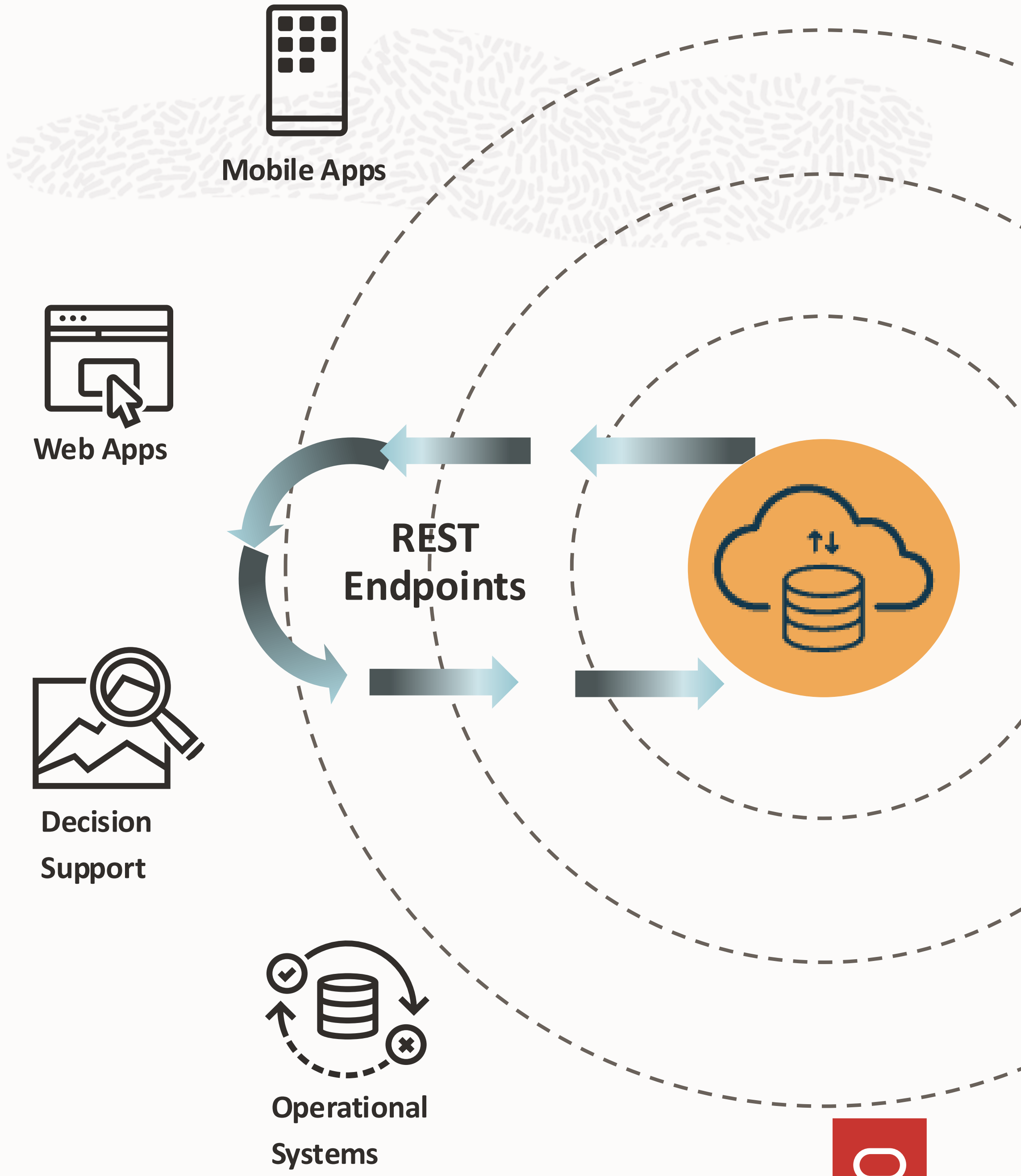
MySQL REST Service joins the party

What is the MySQL REST Service?

"The MySQL REST Service is a next-generation JSON Document Store solution, enabling fast and secure HTTPS access for data stored in MySQL, InnoDB ClusterSet, InnoDB ReplicaSet."

Common use cases:

- Application Backends
- Data-Centric Microservices
- Progressive Web Apps (PWA)
- Data Management
- Data APIs
- Dynamic Content Serving



MySQL REST Service - Building Blocks

MySQL REST Service

Fast, Secure HTTPS Access for MySQL Data & Apps

1. RESTful Web Services

- Auto REST for tables, views and routines
- {JSON} responses with paged results
- Developer support (GUI, CLI, API)
- Support for popular OAuth2 services

2. REST SQL Extension

```
sql> CONFIGURE REST METADATA;
sql> CREATE REST SERVICE /myService;
sql> CREATE REST SCHEMA /sakila
   ON SERVICE /myService FROM `sakila`;
```

- Fully manageable through REST SQL extension
- Full GUI support for increased ease-of-use

3. Powerful Data Mapping

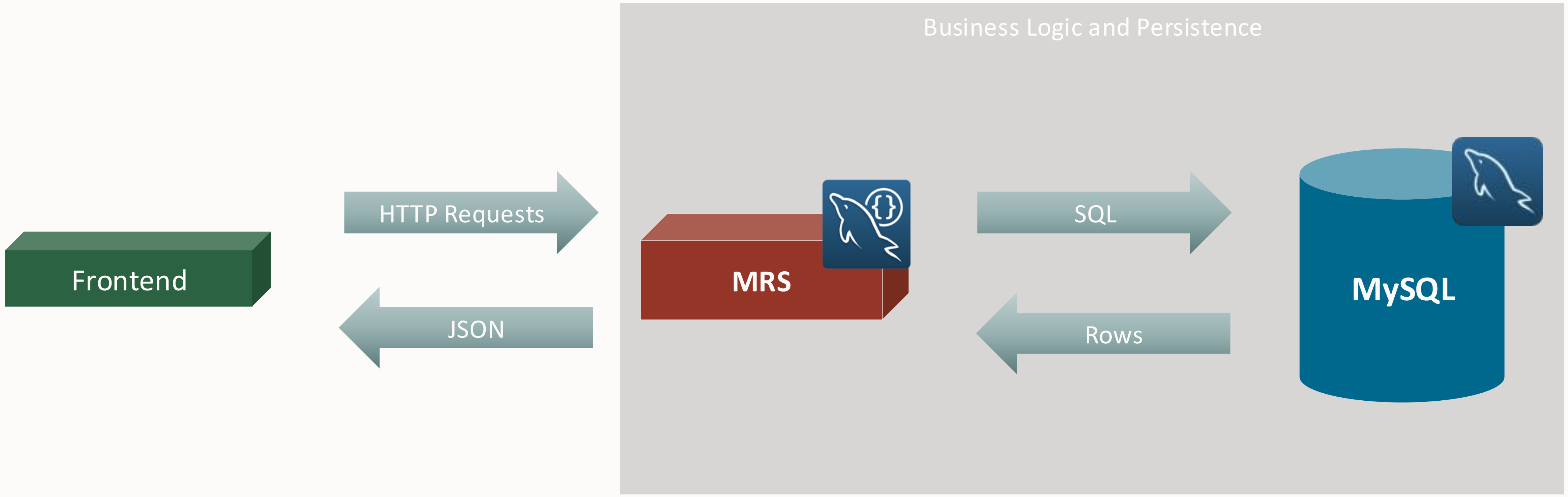
- Nested TABLEs to REST endpoints mapping
- Visual Data Mapping Editor to build complex JSON structures with ease
- SQL & SDK Preview

4. Client SDK Generation

- Tailored SDK for all RESTful Endpoints
- Fully-typed SDK to prevent errors
- Popular, Prisma-like API, live prototyping

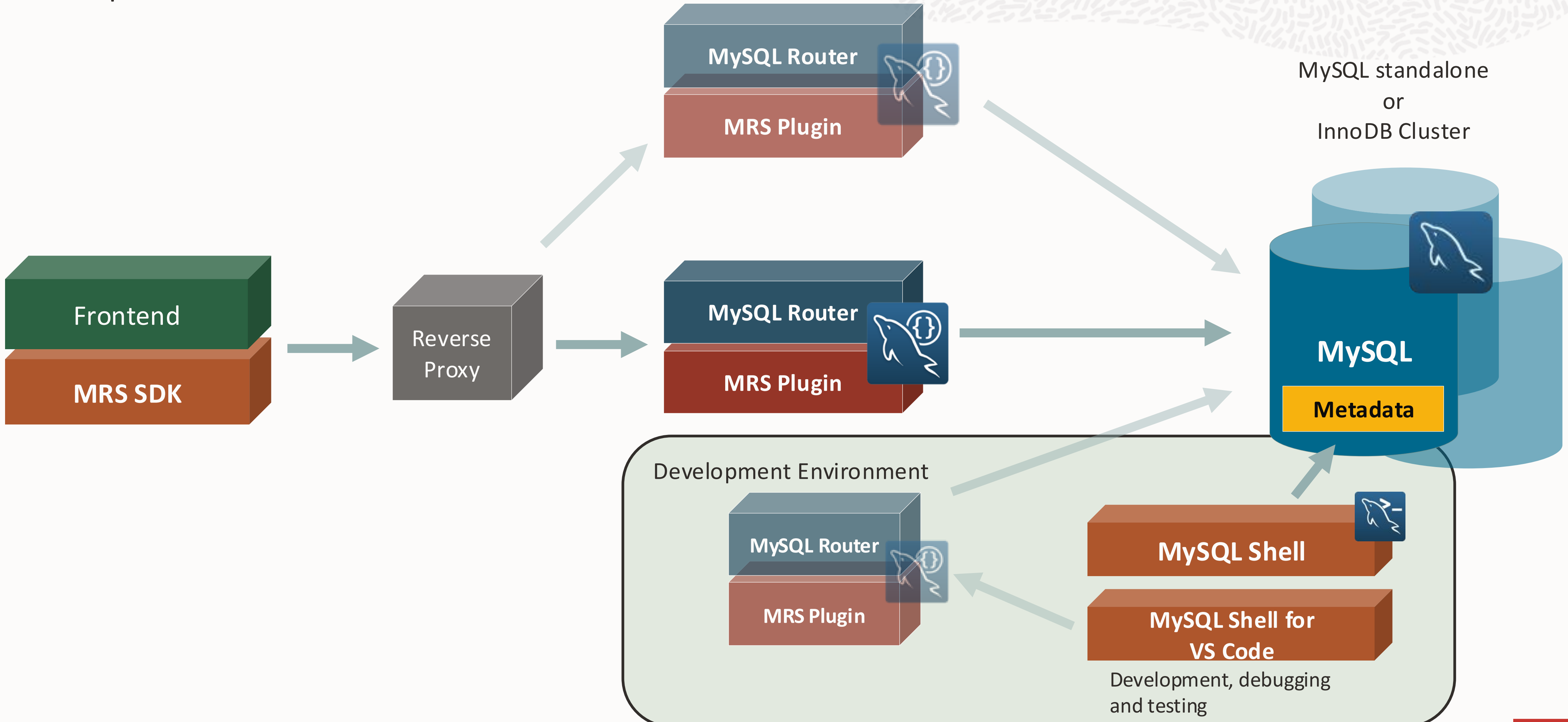
MySQL REST Service

Overview



MySQL REST Service

Component Overview

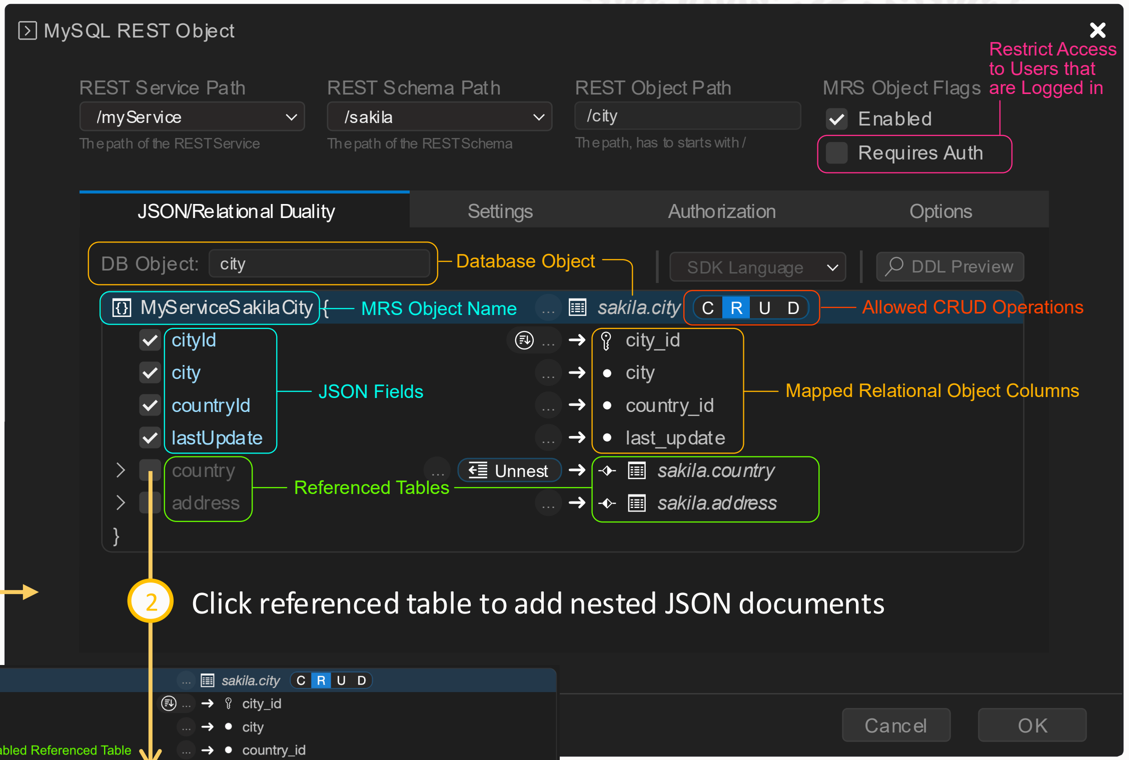


Eliminating the “tax”: key functionalities

Direct Data Mapping

- Powerful WYSIWYG Data-Mapping Editor
- Creation of complex JSON structures with a few clicks
- Automatic database schema analysis
- REST SQL Preview

1 Add single relational table as REST object



MySQL REST Object

REST Service Path	REST Schema Path	REST Object Path	MRS Object Flags
/myService	/sakila	/city	✓ Enabled
The path of the REST Service	The path of the REST Schema	The path, has to starts with /	Requires Auth

Restrict Access to Users that are Logged in

JSON/Relational Duality | Settings | Authorization | Options

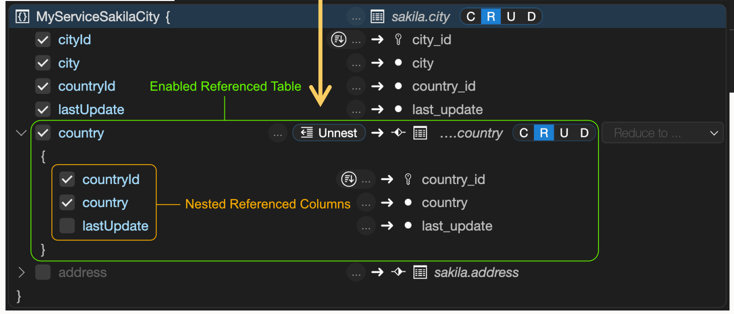
DB Object: city — Database Object

SDK Language | DDL Preview

```
MyServiceSakilaCity {          — MRS Object Name      sakila.city  C R U D   — Allowed CRUD Operations
  ✓ cityId                                        →  city_id
  ✓ city            — JSON Fields                 →  city          — Mapped Relational Object Columns
  ✓ countryId                                     →  country_id
  ✓ lastUpdate                                    →  last_update
  > country                             Unnest    →  sakila.country
  > address         — Referenced Tables           →  sakila.address
}
```

Cancel | OK

2 Click referenced table to add nested JSON documents

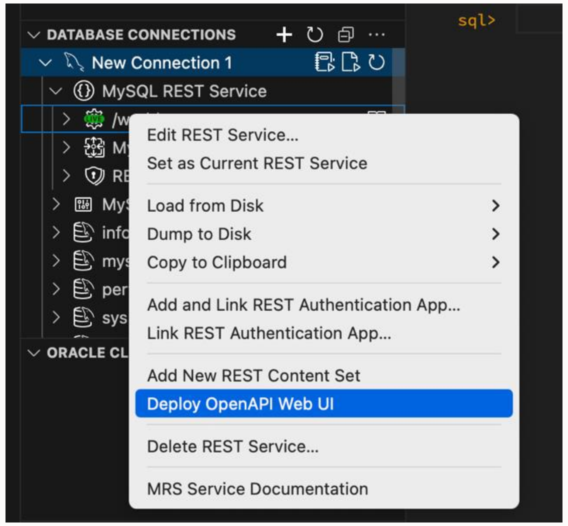


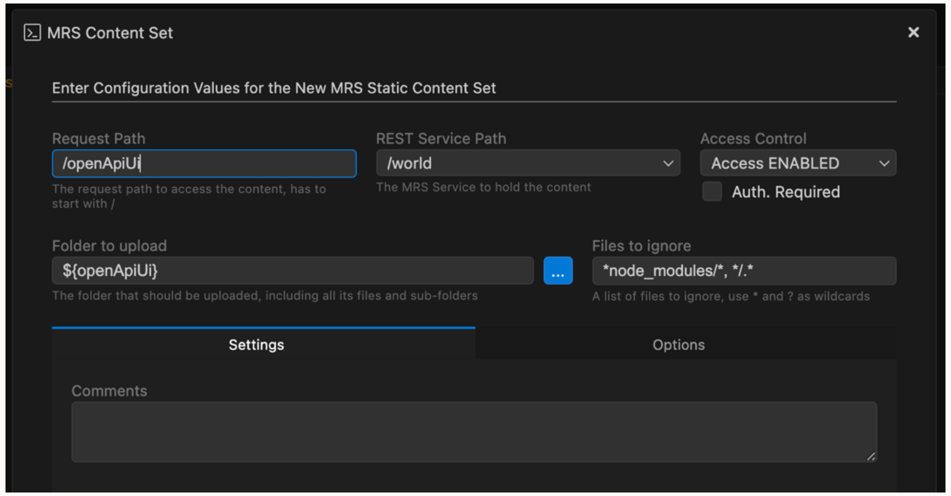
```
MyServiceSakilaCity {                              sakila.city  C R U D
  ✓ cityId                                    →  city_id
  ✓ city                                      →  city
  ✓ countryId   Enabled Referenced Table      →  country_id
  ✓ lastUpdate                                →  last_update
  ✓ country                         Unnest    →  ....country  C R U D   Reduce to ...
  {
    ✓ countryId                               →  country_id
    ✓ country    Nested Referenced Columns    →  country
      lastUpdate                              →  last_update
  }
  > address                                   →  sakila.address
}
```

3 Store REST Mapping

One click OpenAPI Documentation

Deploy instantly online documentation according to OpenAPI specification in one click and start testing APIs right away.




MRS Content Set

Enter Configuration Values for the New MRS Static Content Set

Request Path: /openApiUi
The request path to access the content, has to start with /

REST Service Path: /world
The MRS Service to hold the content

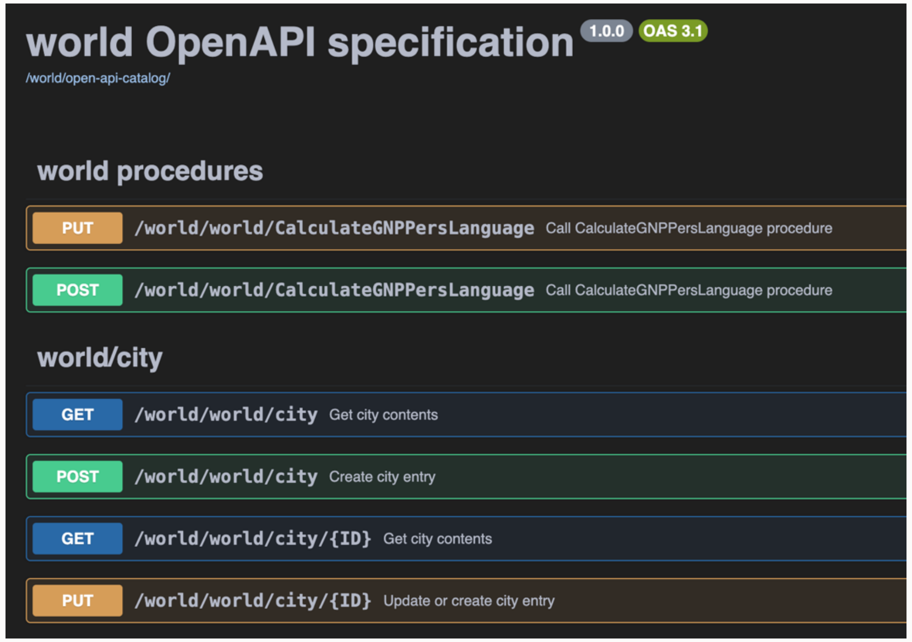
Access Control: Access ENABLED
Auth. Required

Folder to upload: ${openApiUi}
The folder that should be uploaded, including all its files and sub-folders

Files to ignore: *node_modules/*, */.*
A list of files to ignore, use * and ? as wildcards

Settings | Options

Comments


world OpenAPI specification 1.0.0 OAS 3.1
/world/open-api-catalog/

world procedures

PUT /world/world/CalculateGNPPersLanguage Call CalculateGNPPersLanguage procedure

POST /world/world/CalculateGNPPersLanguage Call CalculateGNPPersLanguage procedure

world/city

GET /world/world/city Get city contents

POST /world/world/city Create city entry

GET /world/world/city/{ID} Get city contents

PUT /world/world/city/{ID} Update or create city entry

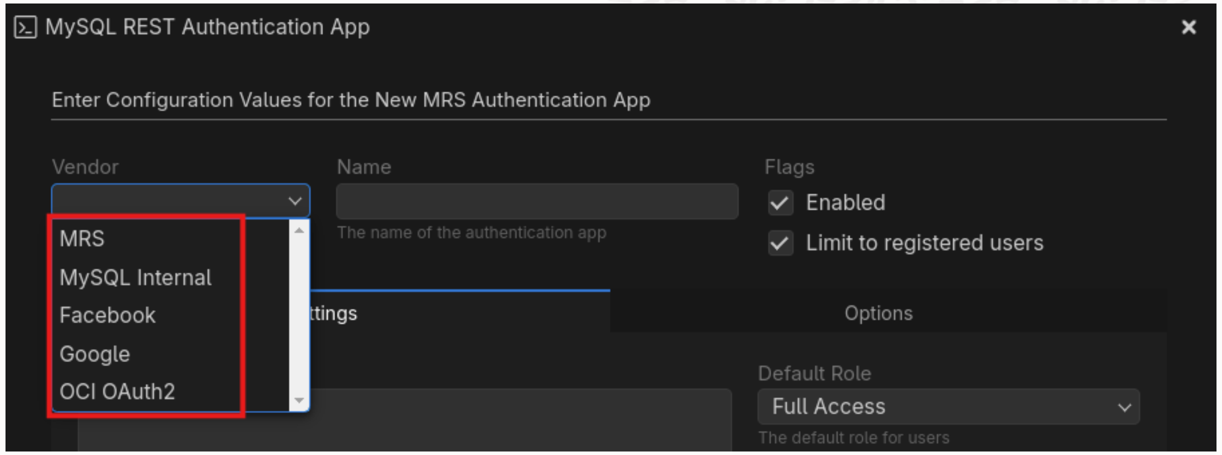
Built-in Security

Integrated authentication:

- OAuth2 (Google, Facebook, OCI)
- Local MRS accounts
- Native MySQL accounts.

Row-level security:

- Ensure users only see what they are supposed to see



MySQL REST Authentication App

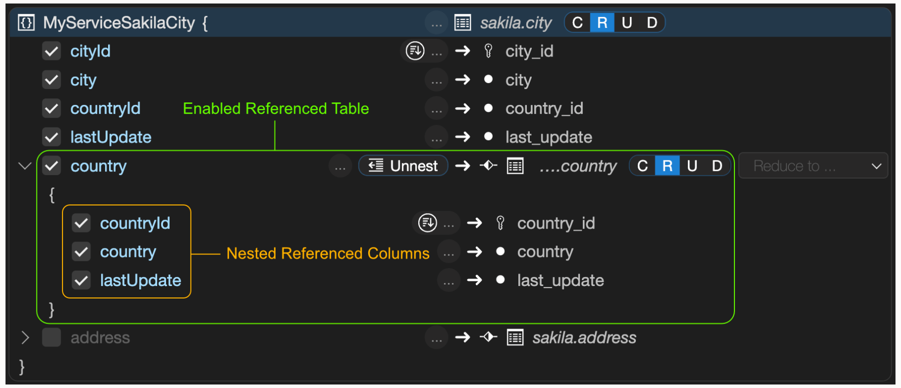
Enter Configuration Values for the New MRS Authentication App

Vendor	Name	Flags
MRS	The name of the authentication app	✓ Enabled
MySQL Internal		✓ Limit to registered users
Facebook		
Google		
OCI OAuth2		

ttings | Options

Default Role: Full Access

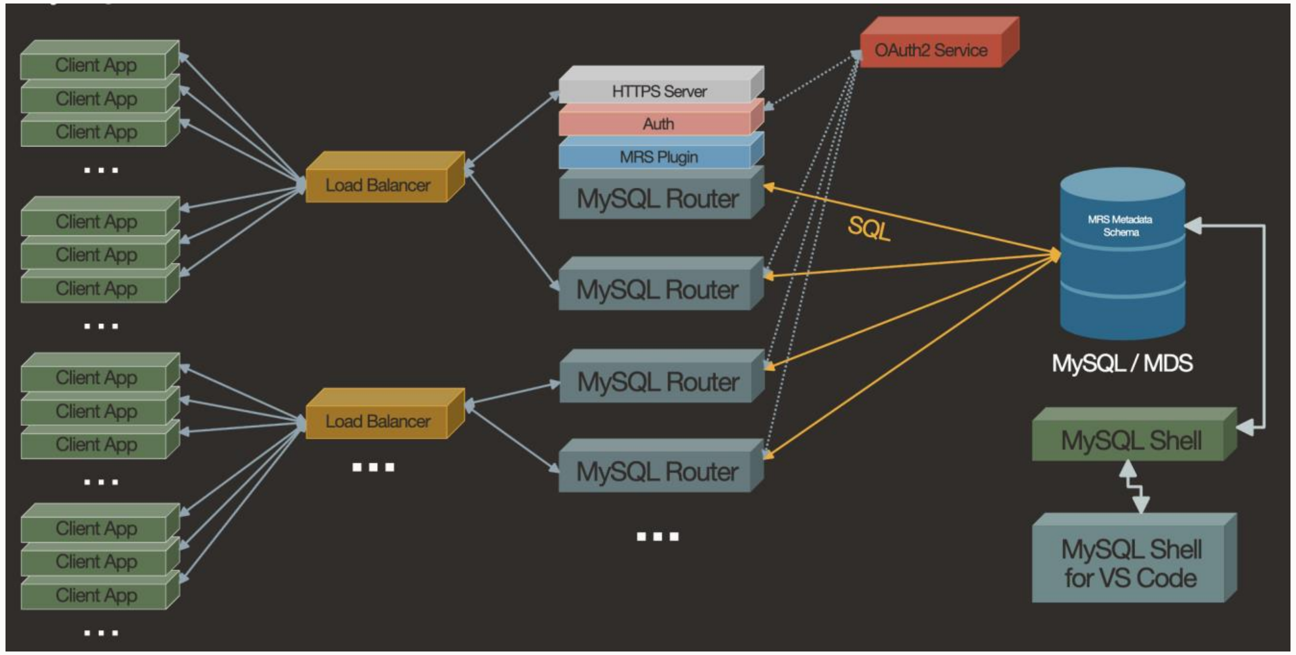
The default role for users



```
MyServiceSakilaCity {                                   sakila.city   C R U D
  ✓ cityId                                        →  city_id
  ✓ city                                          →  city
  ✓ countryId      Enabled Referenced Table       →  country_id
  ✓ lastUpdate                                    →  last_update
  ✓ country                             Unnest    →  ....country   C R U D   Reduce to ...
  {
    ✓ countryId                                   →  country_id
    ✓ country       Nested Referenced Columns     →  country
    ✓ lastUpdate                                  →  last_update
  }
    address                                       →  sakila.address
}
```

Performance and Scaling

- **Integrated In-Memory Caching**: Built-in response and file caching significantly reduce latency and database load for frequent requests.
- **Native JSON**: Eliminates ORM overhead by mapping relational data directly to JSON, enabling high-speed data exchange with modern web apps.
- **Built-in Horizontal Scalability**: Designed for deployment across multiple MySQL Router instances to handle high-traffic web and mobile workloads.
- **Optimized Data Retrieval**: Native support for server-side filtering, sorting, and automatic paging to minimize network overhead and memory usage.
- **Resource Protection**: Integrated request throttling and rate-limiting to maintain system stability during traffic spikes or security threats.



Enrich your APIs:
JavaScript Stored Programs in MySQL

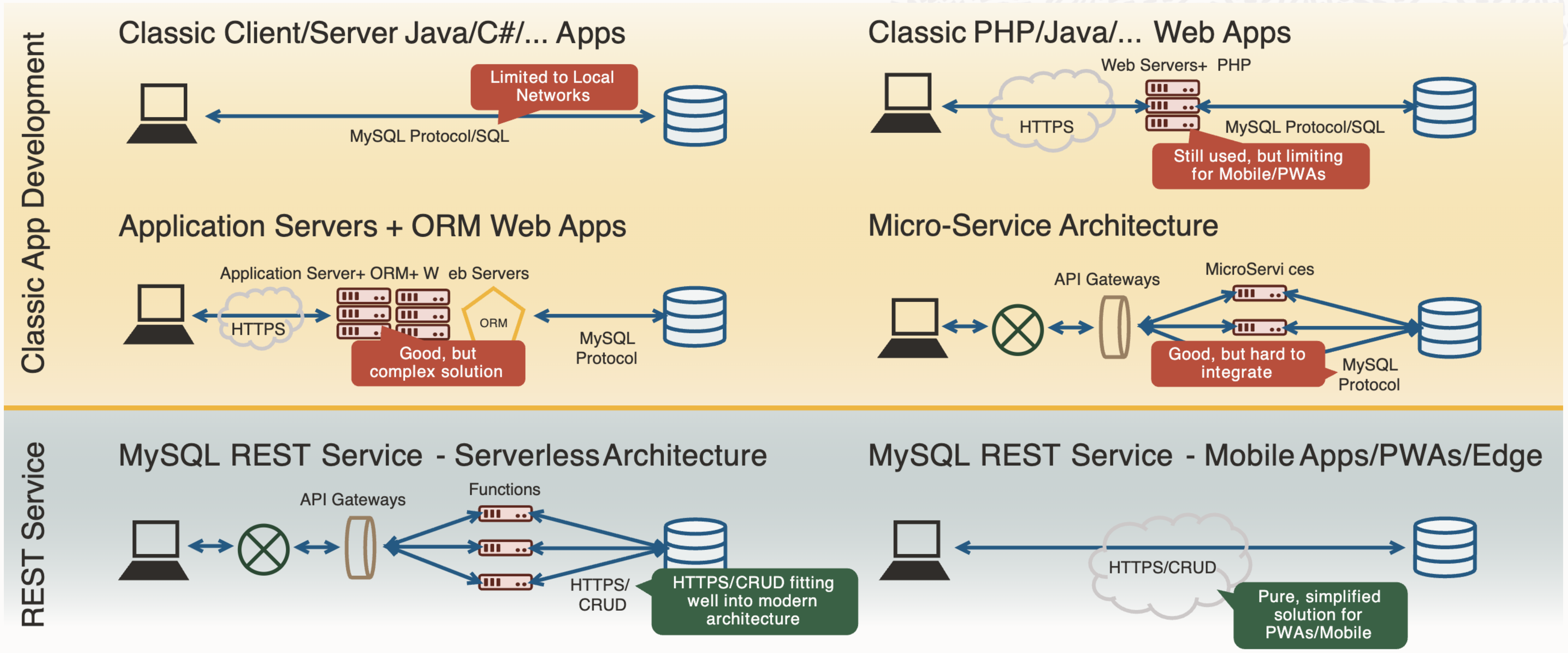
```
CREATE FUNCTION construct_url (path VARCHAR(50),
search VARCHAR(20)) RETURNS VARCHAR(100)
LANGUAGE JAVASCRIPT AS $$
  let url = `${path}${search &&
    !search.startsWith('?') ? '?' : ''}${search ?? ''}`;
  return encodeURI(url);
$$
```

```
SELECT construct_url('/page', 'query=шел  лы');
/page?query=%D1%88%D0%B5%D0%BB%D0%BB%D1%8B
```

```
CREATE PROCEDURE update_item_urls(OUT url_count INT)
LANGUAGE JAVASCRIPT AS $$
  let result = mysql.getSession().runSql(
    `UPDATE my_table
     SET url = construct_url(path, CONCAT('item=',product))
     WHERE product IS NOT NULL`
  );
  url_count = result.getAffectedItemsCount();
$$
```

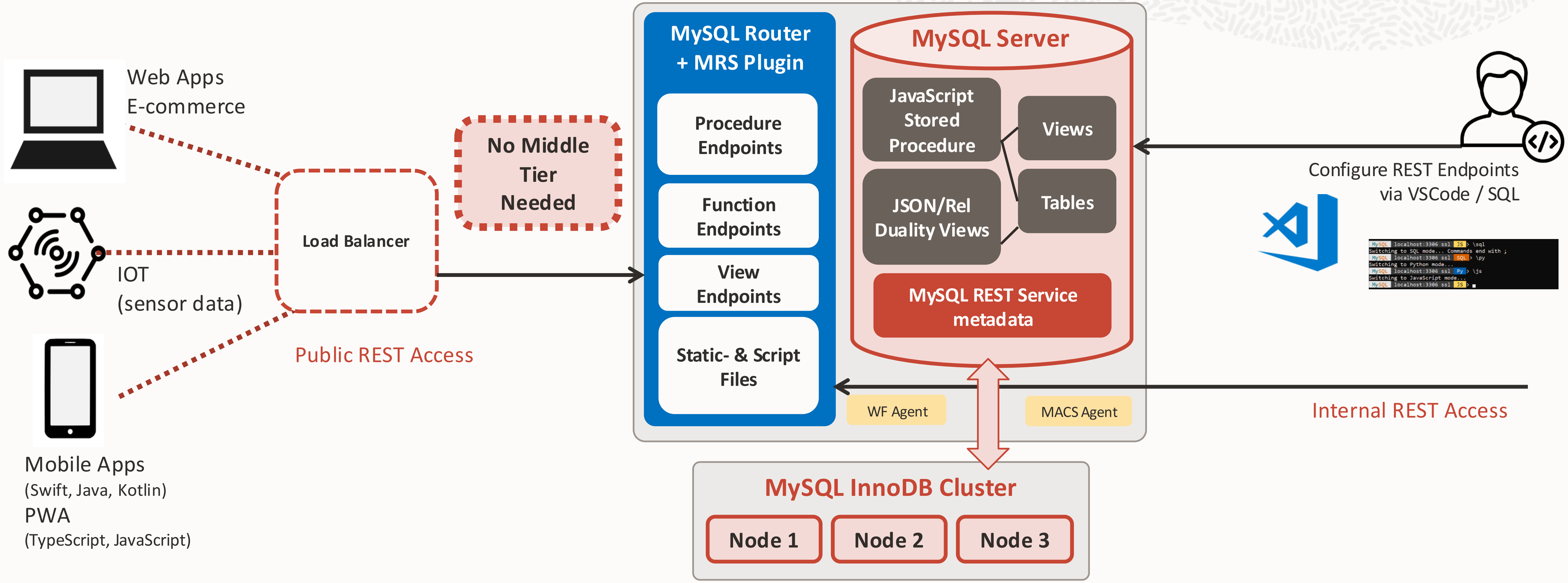
- Seamless MySQL ↔ JavaScript type conversion for input / output arguments
- Can be used anywhere a SQL stored function can be used
    - e.g. SELECT, WHERE, ORDER BY
- Support for DML, DDL, Views
- Existing XDevAPI used to execute SQL inside JavaScript

Sample Use cases


Classic App Development
Classic Client/Server Java/C#/... Apps
Limited to Local Networks
MySQL Protocol/SQL
Classic PHP/Java/... Web Apps
Web Servers+ PHP
HTTPS
MySQL Protocol/SQL
Still used, but limiting for Mobile/PWAs
Application Servers + ORM Web Apps
Application Server+ ORM+ W eb Servers
HTTPS
ORM
Good, but complex solution
MySQL Protocol
Micro-Service Architecture
API Gateways
MicroServi ces
Good, but hard to integrate
MySQL Protocol
REST Service
MySQL REST Service - Serverless Architecture
API Gateways
Functions
HTTPS/ CRUD
HTTPS/CRUD fitting well into modern architecture
MySQL REST Service - Mobile Apps/PWAs/Edge
HTTPS/CRUD
Pure, simplified solution for PWAs/Mobile

Summary and Q&A

Modern Application Architecture
End-to-end Use Case



```
MySQL  localhost:3306 ssl  JS > \sql
Switching to SQL mode... Commands end with ;
MySQL  localhost:3306 ssl  SQL > \py
Switching to Python mode...
MySQL  localhost:3306 ssl  Py > \js
Switching to JavaScript mode...
MySQL  localhost:3306 ssl  JS >
```

Putting It All Together

Make MySQL independent from specific middleware
Moving the REST interface directly into the database allows you the choice to bypass the middleware or to easily plug to many different middle-tier tools.

Developer velocity via direct access
By allowing the database to speak to applications in their native language, teams can bypass the time-consuming process of writing, testing and deploying API code.

Data Security without complexity
Keep the security closer to the data and streamlined, leveraging MySQL integrated security for data and APIs, rather than outsourcing it to the middle tier.

Reduced TCO
Fewer moving parts will reduce the technical debt on the long run. Ending middleware dependency will result in a stack easier to maintain.

Q&A

Thank you!

vittorio.cioe@oracle.com

ORACLE