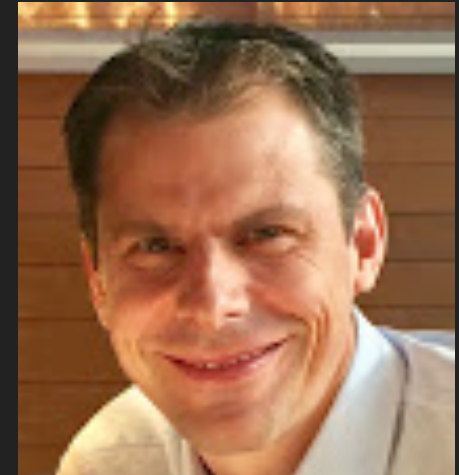


DIMITRI GIELIS

**REST WEB SERVICE? NO,
GRAPHQL PLEASE!**

ABOUT ME

- ▶ Founder & CEO of APEX R&D
- ▶ 20+ years of Oracle Experience (OCP & APEX Certified)
- ▶ Oracle ACE Director
- ▶ "APEX Developer of the year 2009" by Oracle Magazine
- ▶ "Oracle Developer Choice award (ORDS)" in 2015
- ▶ Author Expert Oracle APEX
- ▶ Presenter at Conferences





www.apexRnD.be



www.apexofficeprint.com

http://dgielis.blogspot.com

@dgielis

Dimitri Gielis Blog (Oracle Application Express - APEX)

I created this Blog to share my knowledge especially in Oracle Application Express (APEX) and my feelings ...

FRIDAY, SEPTEMBER 26, 2014

APEX 4.2.6 patch available

Oracle released [APEX 4.2.6](#) yesterday, it's also available from My Oracle Support as a patch set 19595896.

Here's the table of bug fixes.

Table 6 Bugs Fixed in the Oracle Application Express 4.2.6.00.03 Patch Set

Bug Number	Description
18393429	Importing pre-4.2 application with data loading results in runtime error
18609856	APEX_WEB_SERVICE.CLOBBASE642BLOB converts incorrectly
18658967	Corrupted HTML mail messages are created from APEX_MAIL.SEND
18719750	ORA-01858 during application import
18733989	Errors picking up session with cookie when TOP CALL is not F, SHOW, or ACCEPT
19021052	Oracle Application Express 4.2.4 throwing error while generating check boxes
19313700	Check box fails with ORA-06502 after upgrade from release 4.0 to 4.2.4
19482775	ORA-06502 when importing application containing translations

Applying the patch took less than 7 minutes.

```
Session altered.
...Validating Application Express
...Database user "SYS", database schema "APEX_040200", user# "106" 20:48:40
...Compiled 0 out of 3014 objects considered, 0 failed compilation 20:48:40
...271 packages
...263 package bodies
...452 tables
...11 functions
...16 procedures
...3 sequences
...457 triggers
...1320 indexes
...211 views
...0 libraries
...6 types
...0 type bodies
...0 operators
...0 index types
```

My next public appearances:

- Oracle Open World: 27-SEP/2-OCT-2014
- ORCLAPEX-BE Meetup: 19-NOV-2014
- ODTUG KScope: 21/25-JUN-2015

About Me



Dimitri Gielis

Passionate about Oracle and especially Oracle Application Express (APEX). Building applications with love @ APEX R&D.

[View my complete profile](#)



ORACLE
ACE Director

Need APEX consulting, training, development, then visit:



Twitter

[Follow @dgielis](#)





WAYS TO QUERY THE DATABASE?



SQL

STRUCTUREDQL

GRAPHQL

A query language for your API

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

CREATE API?

REST

REST API

GET

/hr/employees/:id

RESPONSE

```
{  
  "empno": 7900,  
  "ename": "JAMES",  
  "job": "CLERK",  
  "mgr": 7698,  
  "hiredate": "1981-12-03T00:00:00Z",  
  "sal": 950,  
  "comm": null,  
  "deptno": 30  
}
```

CHARACTERISTICS

- ▶ Server controls the data you get
- ▶ May need multiple requests to obtain data
- ▶ Static versions of API

CREATE API?

GRAPHQL

GET

/hr

Query

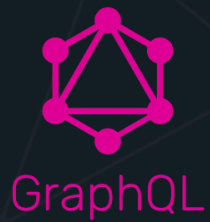
```
{  
  employees(empno: 7900) {  
    ename  
    job  
  }  
}
```

RESPONSE

```
{  
  "data" : {  
    "employees": [{  
      "ename": "JAMES",  
      "job": "CLERK"  
    }]  
  }  
}
```

CHARACTERISTICS

- ▶ Server defines what is available, but Client controls the data it get
- ▶ Single request to get all
- ▶ Evolve API over time
(even make fields deprecated)
- ▶ Auto documented



Describe your data

```
type Project {  
  name: String  
  tagline: String  
  contributors: [User]  
}
```

Ask for what you want

```
{  
  project(name: "GraphQL") {  
    tagline  
  }  
}
```

Get predictable results

```
{  
  "project": {  
    "tagline": "A query language for APIs"  
  }  
}
```

HISTORY

- ▶ Facebook's mobile apps have been powered by GraphQL since 2012.
- ▶ A GraphQL spec was open sourced in 2015
- ▶ Many implementations in different languages
- ▶ Used by many big companies e.g.



DEMO (CONSUMING)



```
1 # Welcome to GraphiQL
2 #
3 # GraphiQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intel
7 # typeaheads aware of the current GraphQL type schema and live synt
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 #   {
16 #     field(arg: "value") {
17 #       subField
18 #     }
19 #   }
20 #
21 # Keyboard shortcuts:
22 #
23 #   Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
24 #
25 #   Run Query:      Ctrl-Enter (or press the play button above)
26 #
27 #   Auto Complete:  Ctrl-Space (or just start typing)
28 #
```

```
30 query {
31   allFilms {
32     edges {
33       node {
34         id
35         title
36         producers
37       }
38     }
39   }
40 }
```

<https://graphql.org/swapi-graphql/>

QUERY VARIABLES

```
{
  "data": {
    "allFilms": {
      "edges": [
        {
          "node": {
            "id": "ZmlsbXM6MQ==",
            "title": "A New Hope",
            "producers": [
              "Gary Kurtz",
              "Rick McCallum"
            ]
          }
        },
        {
          "node": {
            "id": "ZmlsbXM6Mg==",
            "title": "The Empire Strikes Back",
            "producers": [
              "Gary Kurtz",
              "Rick McCallum"
            ]
          }
        },
        {
          "node": {
            "id": "ZmlsbXM6Mw==",
            "title": "Return of the Jedi",
            "producers": [
              "Howard G. Kazanjian",
              "George Lucas",
              "Rick McCallum"
            ]
          }
        },
        {
          "node": {
            "id": "ZmlsbXM6NA==",
            "title": "The Phantom Menace",
            "producers": [
              "Rick McCallum"
            ]
          }
        }
      ]
    }
  }
}
```



```
1 # Hit the Play button above!
2 # Hit "Docs" on the right to explore the API
3
4 {
5   twitter {
6     user (identifier: name, identity: "dgielis") {
7       created_at
8       description
9       id
10      screen_name
11      name
12      profile_image_url
13      url
14      tweets_count
15      followers_count
16      tweets(limit: 1) {
17        text
18      }
19    }
20    search(q: "orclapex", count: 10, result_type: mixed) {
21      user {
22        screen_name
23      }
24      id
25      text
26    }
27  }
28 }
```

QUERY VARIABLES

```
1 null
```

<https://www.graphqlhub.com>

```
{
  "data": {
    "twitter": {
      "user": {
        "created_at": "Wed Jun 18 03:46:33 +0000 2008",
        "description": "Passionate about Oracle Application Express (A",
        "id": "15153717",
        "screen_name": "dgielis",
        "name": "Dimitri Gielis",
        "profile_image_url": "http://pbs.twimg.com/profile_images/9360",
        "url": "https://t.co/L9ZRSRVK7Q",
        "tweets_count": 2048,
        "followers_count": 3212,
        "tweets": [
          {
            "text": "RT @apexofficeprint: It will be even easier to us"
          }
        ]
      },
      "search": [
        {
          "user": {
            "screen_name": "fsoldera"
          },
          "id": "1142876067377668100",
          "text": "RT @swesley_perth: Well this all looks very interes"
        },
        {
          "user": {
            "screen_name": "rimblas"
          },
          "id": "1142874547558858800",
          "text": "RT @LaurestonAPEX: I think all of us NOT attending"
        },
        {
          "user": {
            "screen_name": "mariuszskora"
          }
        }
      ]
    }
  }
}
```



```

1 {
2   user(id: 2) {
3     fullName
4     email
5     posts {
6       id
7       body
8       createdAt
9       comments {
10        id
11        body
12        author {
13          id
14          fullName
15        }
16      }
17    }
18  }
19 }

```

```

{
  "data": {
    "user": {
      "fullName": "Yasmine Rolfson",
      "email": "Earl.Koss41@yahoo.com",
      "posts": [
        {
          "id": 2,
          "body": "Harum unde maiores est quasi totam consequuntur.
Necessitatibus doloribus ut totam dolore omnis quos error eos. Rem
nostrum assumenda eius veniam fugit dicta in consequuntur. Ut porro
dolorem aliquid qui magnam a.",
          "createdAt": "1468236023128",
          "comments": [
            {
              "id": 6,
              "body": "The AI driver is down, program the multi-byte
sensor so we can parse the SAS bandwidth!",
              "author": {
                "id": 2,
                "fullName": "Yasmine Rolfson"
              }
            },
            {
              "id": 8,
              "body": "Try to program the SMS transmitter. maybe it

```

```

SELECT
  "user"."id" AS "id",
  "user"."email_address" AS "email",
  "posts"."id" AS "posts__id",
  "posts"."body" AS "posts__body",
  "posts"."created_at" AS "posts__createdAt",
  "user"."first_name" AS "first_name",
  "user"."last_name" AS "last_name"
FROM accounts "user"
LEFT JOIN posts "posts" ON "user".id = "posts".author_id
WHERE "user".id = 2
ORDER BY "posts"."id" ASC

```

```

SELECT
  "comments"."id" AS "id"

```

<http://join-monster.herokuapp.com>

GraphQL APIS

- ▶ <https://github.com/APIs-guru/graphql-apis>
- ▶ <https://help.shopify.com/en/api/graphql-admin-api/graphql-explorer>
- ▶ <https://developer.github.com/v4/>
- ▶ <https://www.yelp.com/developers/graphql/guides/intro>



GRAPHQL AND THE ORACLE DATABASE

GETTING STARTED

BUILDING BLOCKS

- ▶ Oracle Database
- ▶ node.js
 - ▶ oracledb
 - ▶ graphql
 - ▶ apollo-server

ORACLE®



MY DEVELOPMENT ENVIRONMENT

- ▶ Visual Studio Code
- ▶ Sqlcl
- ▶ Git
- ▶ node.js & nvm & npm
- ▶ Instant Oracle client / Oracle Database

[HOME](#)[ABOUT](#)[DOWNLOADS](#)[DOCS](#)[GET INVOLVED](#)[SECURITY](#)[NEWS](#)[FOUNDATION](#)

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

Download for macOS (x64)

10.16.0 LTS

Recommended For Most Users

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

12.4.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [Long Term Support \(LTS\) schedule](#).

Sign up for [Node.js Everywhere](#), the official Node.js Monthly Newsletter.

CREATE A FIRST NODE PROJECT

- ▶ `npm init`
- ▶ `npm install apollo-server graphql oracledb`

CREATE A FIRST NODE PROJECT

- ▶ `npm init`
- ▶ `npm install apollo-server graphql oracledb`

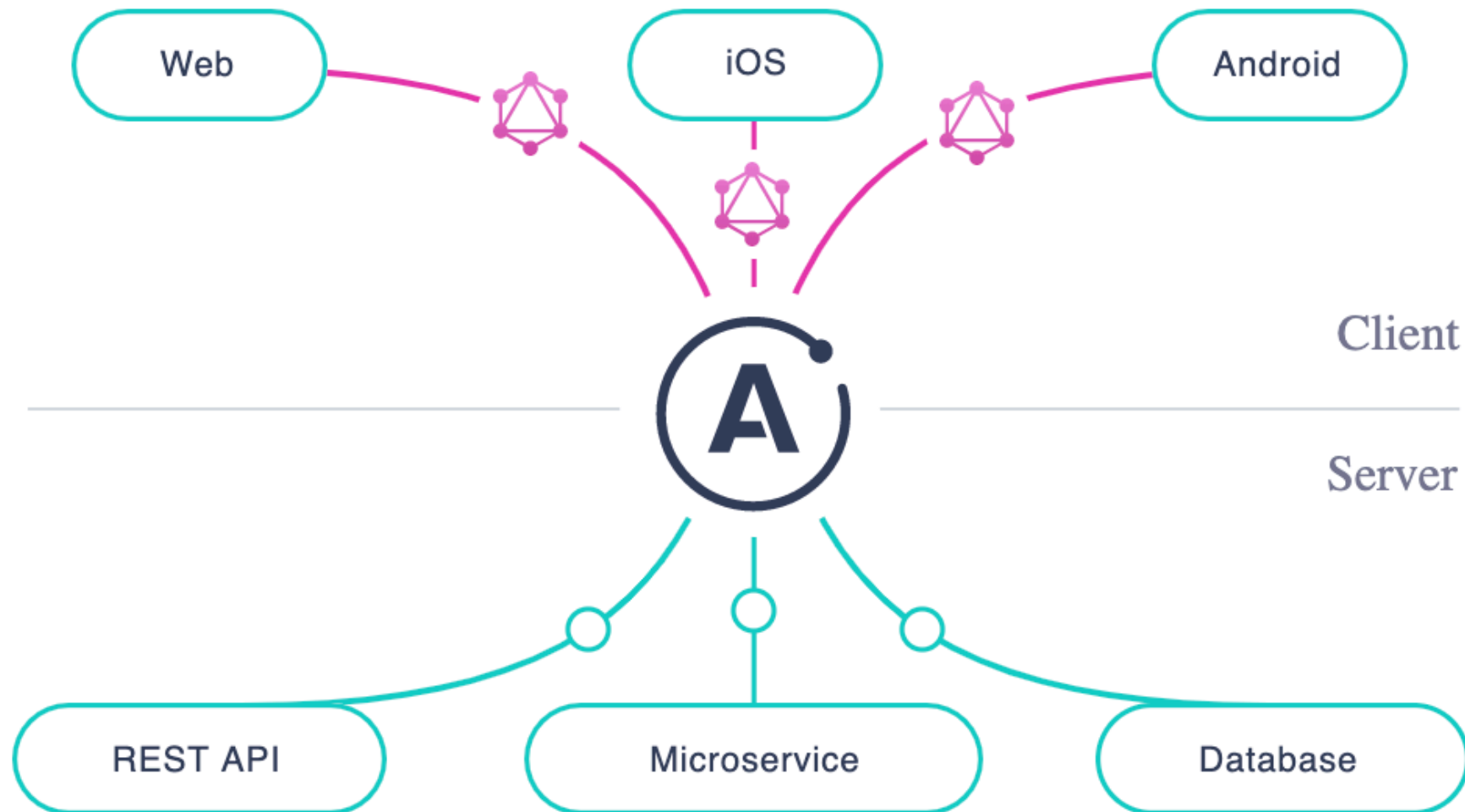
CREATE A FIRST NODE PROJECT

- ▶ `npm init`
- ▶ `npm install apollo-server graphql oracledb`

CREATE A FIRST NODE PROJECT

- ▶ `npm init`
- ▶ `npm install apollo-server graphql oracledb`

ABOUT APOLLO



Apollo Server is the best way to quickly build a production-ready, self-documenting API for GraphQL clients, using data from any source - <https://www.apollographql.com/docs/apollo-server/>

CREATE AN APOLLO SERVER

```
const { ApolloServer, gql } = require('apollo-server');

// This is a (sample) collection of books
const books = [{title: 'Book1'}, {title: 'Book2'}];

// Type definitions define the "shape" of your data
const typeDefs = gql`
  type Book {title: String}
  type Query {books: [Book]}
`;

// Resolvers define the technique for fetching the types
const resolvers = { Query: {books: () => books } };

// The Apollo Server can be started by passing
// type definitions and the resolvers
const server = new ApolloServer({ typeDefs, resolvers });

// This `listen` method launches a web-server
server.listen().then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});
```

CREATE AN APOLLO SERVER

```
const { ApolloServer, gql } = require('apollo-server');
```

```
// This is a (sample) collection of books
const books = [{title: 'Book1'}, {title: 'Book2'}];
```

```
// Type definitions define the "shape" of your data
const typeDefs = gql`
  type Book {title: String}
  type Query {books: [Book]}
`;
```

```
// Resolvers define the technique for fetching the types
const resolvers = { Query: {books: () => books } };
```

```
// The Apollo Server can be started by passing
// type definitions and the resolvers
const server = new ApolloServer({ typeDefs, resolvers });
```

```
// This `listen` method launches a web-server
server.listen().then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});
```

CREATE AN APOLLO SERVER

```
const { ApolloServer, gql } = require('apollo-server');
```

```
// This is a (sample) collection of books
const books = [{title: 'Book1'}, {title: 'Book2'}];
```

```
// Type definitions define the "shape" of your data
const typeDefs = gql`
  type Book {title: String}
  type Query {books: [Book]}
`;
```

```
// Resolvers define the technique for fetching the types
const resolvers = { Query: {books: () => books } };
```

```
// The Apollo Server can be started by passing
// type definitions and the resolvers
const server = new ApolloServer({ typeDefs, resolvers });
```

```
// This `listen` method launches a web-server
server.listen().then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});
```

CREATE AN APOLLO SERVER

```
const { ApolloServer, gql } = require('apollo-server');
```

```
// This is a (sample) collection of books
const books = [{title: 'Book1'}, {title: 'Book2'}];
```

```
// Type definitions define the "shape" of your data
const typeDefs = gql`
  type Book {title: String}
  type Query {books: [Book]}
`;
```

```
// Resolvers define the technique for fetching the types
const resolvers = { Query: {books: () => books } };
```

```
// The Apollo Server can be started by passing
// type definitions and the resolvers
const server = new ApolloServer({ typeDefs, resolvers });
```

```
// This `listen` method launches a web-server
server.listen().then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});
```

CREATE AN APOLLO SERVER

```
const { ApolloServer, gql } = require('apollo-server');

// This is a (sample) collection of books
const books = [{title: 'Book1'}, {title: 'Book2'}];

// Type definitions define the "shape" of your data
const typeDefs = gql`
  type Book {title: String}
  type Query {books: [Book]}
`;

// Resolvers define the technique for fetching the types
const resolvers = { Query: {books: () => books } };

// The Apollo Server can be started by passing
// type definitions and the resolvers
const server = new ApolloServer({ typeDefs, resolvers });

// This `listen` method launches a web-server
server.listen().then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});
```

CREATE AN APOLLO SERVER

```
const { ApolloServer, gql } = require('apollo-server');
```

```
// This is a (sample) collection of books
const books = [{title: 'Book1'}, {title: 'Book2'}];
```

```
// Type definitions define the "shape" of your data
const typeDefs = gql`
  type Book {title: String}
  type Query {books: [Book]}
`;
```

```
// Resolvers define the technique for fetching the types
const resolvers = { Query: {books: () => books } };
```

```
// The Apollo Server can be started by passing
// type definitions and the resolvers
const server = new ApolloServer({ typeDefs, resolvers });
```

```
// This `listen` method launches a web-server
server.listen().then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});
```

CREATE AN APOLLO SERVER

```
const { ApolloServer, gql } = require('apollo-server');

// This is a (sample) collection of books
const books = [{title: 'Book1'}, {title: 'Book2'}];

// Type definitions define the "shape" of your data
const typeDefs = gql`
  type Book {title: String}
  type Query {books: [Book]}
`;

// Resolvers define the technique for fetching the types
const resolvers = { Query: {books: () => books } };

// The Apollo Server can be started by passing
// type definitions and the resolvers
const server = new ApolloServer({ typeDefs, resolvers });

// This `listen` method launches a web-server
server.listen().then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});
```

TEST IN BROWSER: [HTTP://LOCALHOST:4000](http://localhost:4000)

Q books ×

+

⚙

PRETTIFY

HISTORY

● http://localhost:4000/

COPY CURL

1 # Write your query or mutation here

2 query{

3 books {

4 title

5 }

6 }

7

▶

{

"data": {

"books": [

{

"title": "Book1"

},

{

"title": "Book2"

}

]

}

}

DOCS

SCHEMA

QUERY VARIABLES

HTTP HEADERS

TRACING

QUERY PLAN

HOOKING UP ORACLE DB WITH NODE-ORACLEDDB

oracle / node-oracledb

Watch

186

Star

1,577

Fork

791

<> Code

Issues 35

Pull requests 0

Projects 0

Security

Insights

Join GitHub today

Dismiss

GitHub is home to over 36 million developers working together to host and review code, manage projects, and build software together.

Sign up

Oracle Database driver for Node.js maintained by Oracle Corp. <http://oracle.github.io/node-oracledb/>

1,353 commits

7 branches

43 releases

5 contributors

View license

Branch: master

New pull request

Find File

Clone or download



cjbj Allow compilation with Node.js 12

Latest commit d18b024 on 16 May

.github	Link to non-dev versions of the doc. Add specific commands to find ve...	7 months ago
doc	Work on the connection and node-oracledb configuration section	4 months ago
examples	Review sample password usage	5 months ago
lib	Fix error message for invalid use of SODA metaData	5 months ago
odpi @ f7d3fb6	Update to ODPI-C 3.1.2	4 months ago

QUERY ORACLE DB

```
var oracledb = require('oracledb');
async function run() {
  let connection;
  try {
    connection = await oracledb.getConnection( {
      user      : "hr",
      password  : "hr",
      connectString : "localhost/XEPDB1"
    });
    let result = await connection.execute(
      `SELECT manager_id, department_id, department_name
      FROM departments
      WHERE manager_id = :id`,
      [103], // bind value for :id
    );
    console.log(result.rows);
  } catch (err) {
    console.error(err);
  } finally {
    if (connection) {
      try {
        await connection.close();
      } catch (err) {
        console.error(err);
      }
    }
  }
}
run();
```

QUERY ORACLE DB

```
var oracledb = require('oracledb');
async function run() {
  let connection;
  try {
    connection = await oracledb.getConnection( {
      user      : "hr",
      password  : "hr",
      connectString : "localhost/XEPDB1"
    });
    let result = await connection.execute(
      `SELECT manager_id, department_id, department_name
      FROM departments
      WHERE manager_id = :id`,
      [103], // bind value for :id
    );
    console.log(result.rows);
  } catch (err) {
    console.error(err);
  } finally {
    if (connection) {
      try {
        await connection.close();
      } catch (err) {
        console.error(err);
      }
    }
  }
}
run();
```

QUERY ORACLE DB

```
var oracledb = require('oracledb');
async function run() {
  let connection;
  try {
    connection = await oracledb.getConnection( {
      user      : "hr",
      password   : "hr",
      connectString : "localhost/XEPDB1"
    });
    let result = await connection.execute(
      `SELECT manager_id, department_id, department_name
      FROM departments
      WHERE manager_id = :id`,
      [103], // bind value for :id
    );
    console.log(result.rows);
  } catch (err) {
    console.error(err);
  } finally {
    if (connection) {
      try {
        await connection.close();
      } catch (err) {
        console.error(err);
      }
    }
  }
}
run();
```

QUERY ORACLE DB

```
var oracledb = require('oracledb');
async function run() {
  let connection;
  try {
    connection = await oracledb.getConnection( {
      user      : "hr",
      password   : "hr",
      connectString : "localhost/XEPDB1"
    });
    let result = await connection.execute(
      `SELECT manager_id, department_id, department_name
      FROM departments
      WHERE manager_id = :id`,
      [103], // bind value for :id
    );
    console.log(result.rows);
  } catch (err) {
    console.error(err);
  } finally {
    if (connection) {
      try {
        await connection.close();
      } catch (err) {
        console.error(err);
      }
    }
  }
}
run();
```

COMBINING APOLLO AND ORACLE

- ▶ Map SQL to Types

- ▶ TIP:

sqlcl to output SQL statement to JSON

- ▶ convert JSON to GraphQL Types automatically
<https://walmartlabs.github.io/json-to-simple-graphql-schema/>

Paste and edit JSON text

```
{
  "id": "some-id-0",
  "name": "A fun object",
  "color": {
    "id": "color-id-1",
    "name": "Test color"
  },
  "subType": {
    "id": "some-id-1",
    "name": "A fun sub-type",
    "color": {
      "id": "color-id-1",
      "name": "Test color",
      "hex": "#ff0000"
    }
  }
}
```

Generated GraphQL Schema

```
type Color { id: String name: String hex: String }

type SubType { id: String name: String color: Color }

type AutogeneratedMainType { id: String
  name: String
  subType: SubType
  color: Color }
```

DEMO

(PROVIDING)

source: <https://www.apexofficeprint.com/graphql>

JOIN MONSTER

- ▶ A GraphQL to SQL query execution layer for query planning and batch data fetching.

Join Monster



DEMO

(PROVIDING)

source: Dan McGhan presentation: emp/dept example



Core Concepts

GraphQL Fundamentals (3/4)

www.howtographql.com

GRAPHQL CONCEPTS

- ▶ Schema
 - ▶ Object type (character)
 - ▶ Field
 - ▶ Arguments
 - ▶ Scalar type
 - ▶ Interface

HOW TO QUERY A GRAPHQL SERVER

- ▶ Queries
 - ▶ Fields, Aliases, Fragments
 - ▶ Arguments
 - ▶ Variables
 - ▶ Directives
 - ▶ Operation name
- ▶ Mutations
- ▶ Subscriptions

INTERESTING RESOURCES & PROJECTS

- ▶ <https://graphql.org/learn/best-practices/>
- ▶ <https://blogs.oracle.com/opal/demo:-graphql-with-node-oracledb>
- ▶ <https://github.com/sblack4/oracledb-graphql-demo>
- ▶ <https://www.prisma.io> (no Oracle support yet)
- ▶ <https://github.com/rexxars/sql-to-graphql> (unmaintained)