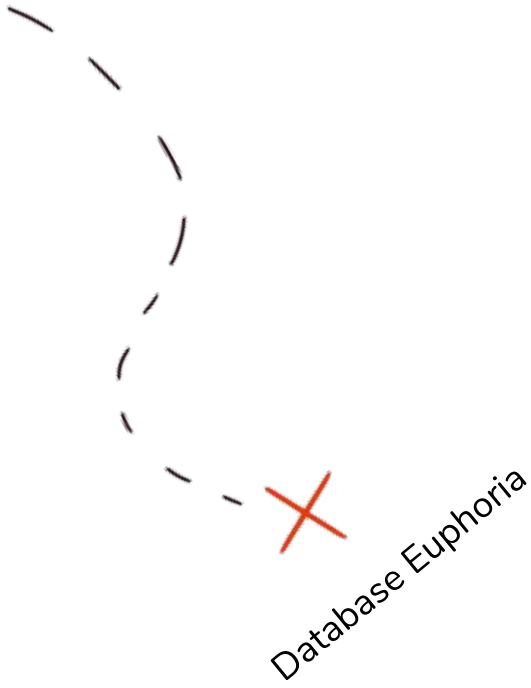
MariaDB

Building Applications using Object-Relational Mapping

Rob Hedgpeth
Developer Evangelist

Agenda

- SQL in Action
- Object-relational mapping
  - What?
  - Why?
- Common Capabilities
- Demo
- What’s the verdict?

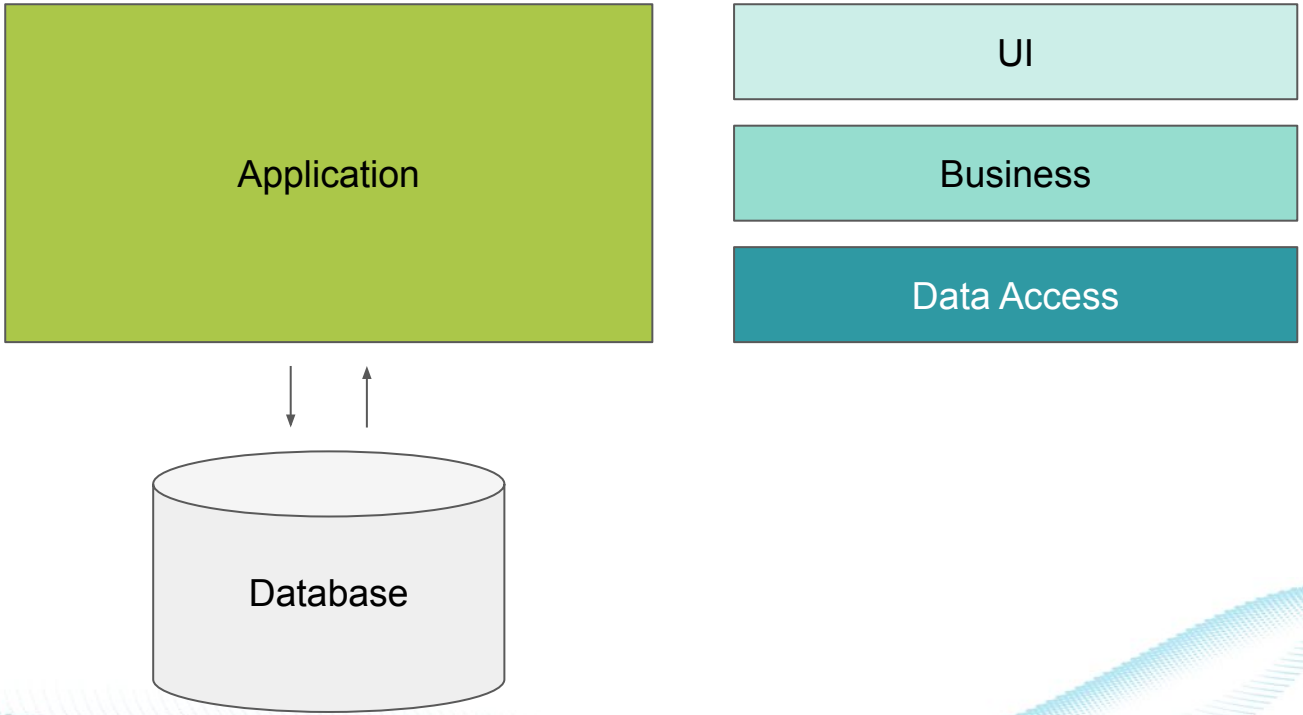


MariaDB

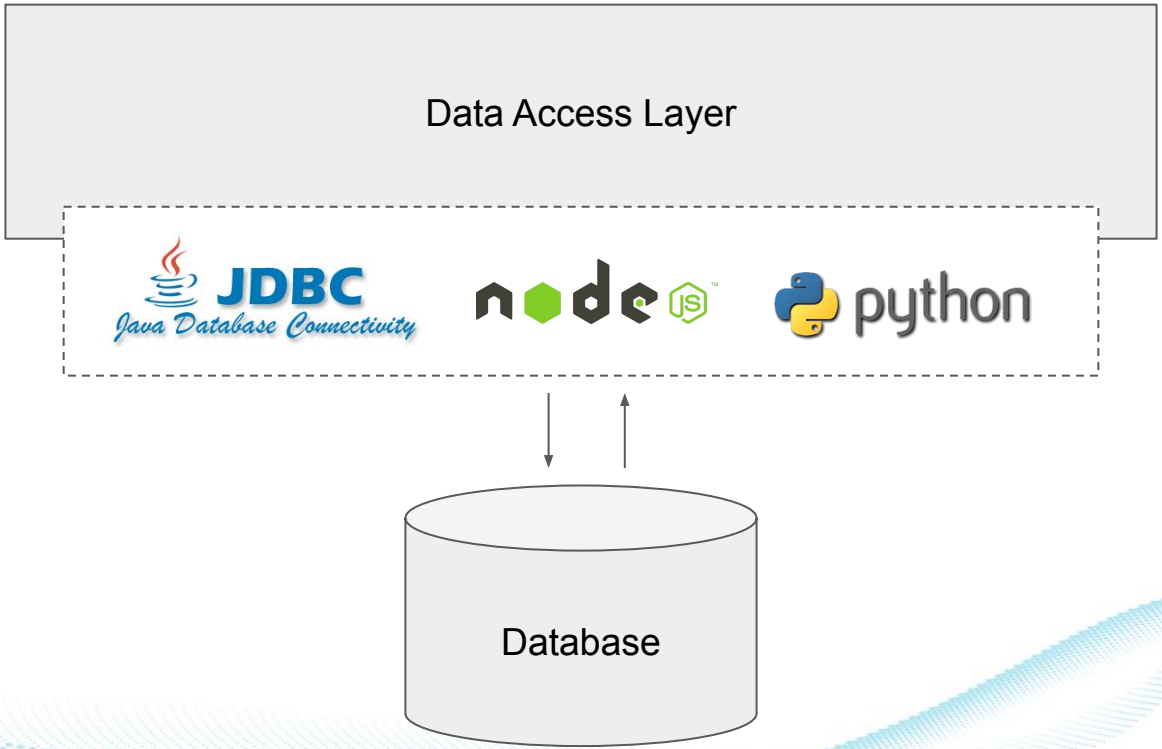
SQL in Action

MariaDB

Traditional Approach



Database Connectors



Connecting to the database

```
String url =
"jdbc:mariadb://localhost:3306/DB?user=root&password=myPassword";

Connection connection = DriverManager.getConnection(url);

Statement stmt = connection.createStatement();

stmt.executeQuery("SELECT * FROM activities");

ResultSet rs = st.executeQuery(query);

stmt.close();

connection.close();
```

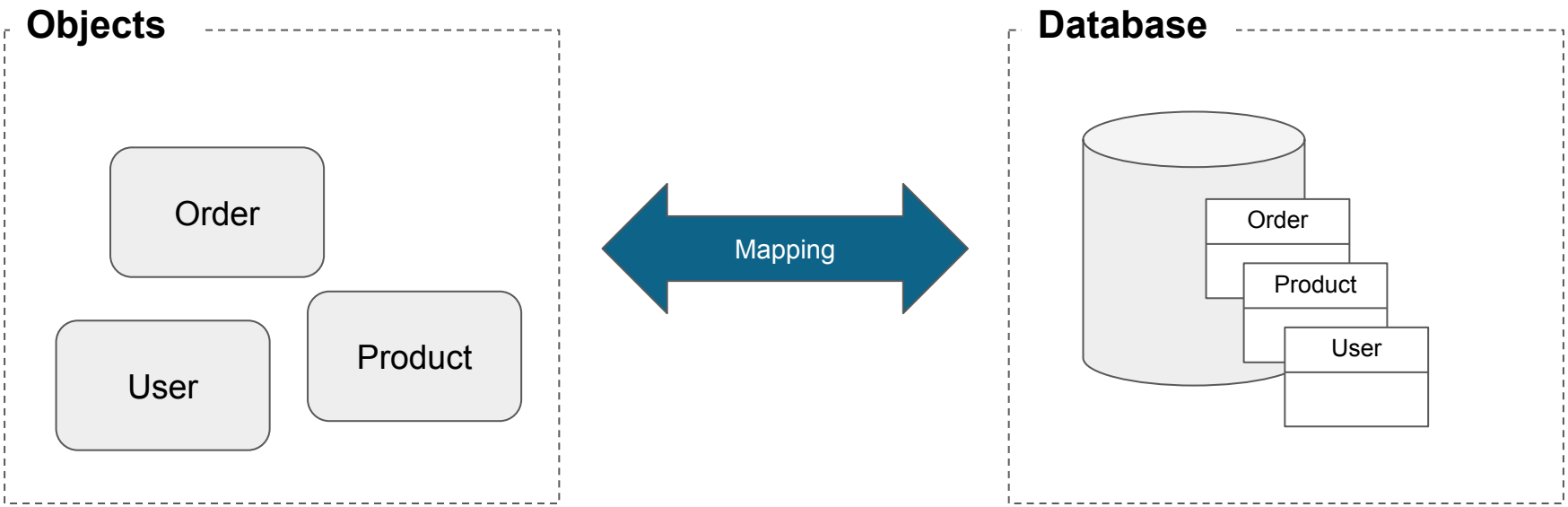
The takeaway

- Most applications require persistence
- Managing persistence directly
  - Error prone
  - Brittle
  - Time consuming
    - Ongoing maintenance / tuning
    - Split skill set
  - Other considerations (e.g. caching)

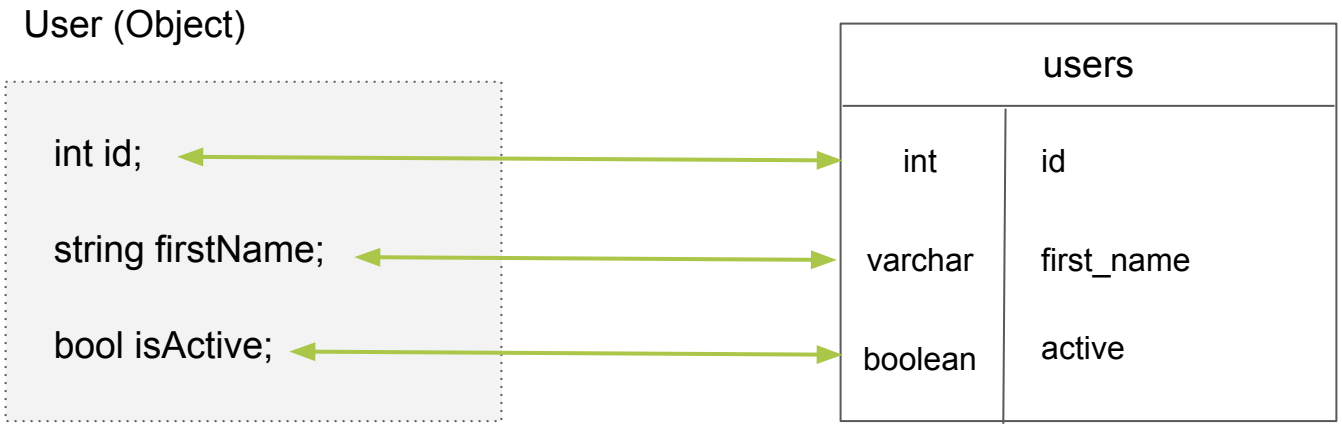
Object-Relational Mapping

MariaDB

What is Object-Relational Mapping?



What is Object-Relational Mapping?



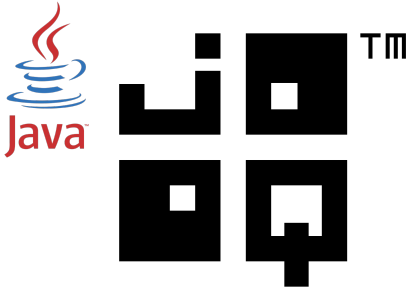
Why Object-Relational Mapping?

- Eliminates the need to create boilerplate code
  - Data access/management
  - Parameterization
  - Basic error handling
- Provides a persistence of objects, often automated
- Maps an object to one or more datasource entities
- Many other out-of-the-box capabilities

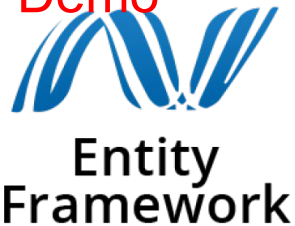
ORM Capabilities

MariaDB

ORM Tools



Demo



Slides

Getting started

MariaDB

Dependencies

```
<dependency>
        <groupId>org.mariadb.jdbc</groupId>
        <artifactId>mariadb-java-client</artifactId>
        <version>2.6.0</version>
</dependency>
```

MariaDB Connector/J

```
<dependency>
       <groupId>org.hibernate</groupId>
       <artifactId>hibernate-core</artifactId>
       <version>5.X.XX.Final</version>
</dependency>
```

Hibernate

```
<dependency>
      <groupId>org.jooq</groupId>
      <artifactId>jooq</artifactId>
      <version>3.13.4</version>
</dependency>
```

jOOQ

Connection Configuration

Config	Value
Dialect	`MariaDB`
Driver	`org.mariadb.Driver`
JDBC	`jdbc:mariadb://localhost:3306/activities`
Username	`some_user`
Password	`some_password`

Connecting

Hibernate

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("p1");
EntityManager entityManager = emf.createEntityManager();
```

Manage persistence

jOOQ

```
Connection connection = DriverManager.getConnection(url, userName, password);
DSLContext context = DSL.using(connection, SQLDialect.MYSQL);
```

Creating Models and Mappings

```
@Entity
@Table(name = "players")
public class Player {
    @Id
    private Long id;
    @Column(name = "first_name")
    private String firstName;
    @Column(name = "last_name")
    private String lastName;
}
```

players

id	first_name	last_name

CRUD

Hibernate

```
Player player = new Player();
player.setFirstName("Rob");
player.setLastName("Hedgpeth");
player.setIsActive(true);

em.persist(player);
```

jOOQ

```
Player player = context.newRecord(PLAYER);
player.setFirstName("Rob");
player.setLastName("Hedgpeth");
player.setIsActive(true);

player.store();
```

```
INSERT INTO activities (first_name, last_name, active) VALUES ('Rob', 'Hedgpeth', true);
```

CRUD

```
SELECT first_name, last_name, active FROM
players;
```

Hibernate

```
Player player =
     em.find(Player.class, 1);

player.setIsActive(true);
em.remove(player);
em.merge(player);
```

jOOQ

```
Player player =
     context.fetchSingle(PLAYER,PLAYER.ID.eq(1));

player.setIsActive(true);
player.delete();
player.store();
```

```
UPDATE players SET active = 1 WHERE id = 1;
DELETE FROM players WHERE id = 1;
```

Querying / Filtering

MariaDB

Filtering

```
SELECT *
FROM players
WHERE first_name = ? AND active = ?
```

Hibernate

```
List<Player> players = entityManager
     .createQuery("SELECT p FROM Player p WHERE p.firstName = :firstName AND p.isActive =
true",       Player.class)
     .setParameter("firstName", firstName)
     .getResultList();
```

Java Persistence Query Language
(JPQL)

jOOQ

```
List<Player> players = context
                              .select()
                              .from(PLAYER)
                              .where(PLAYER.FIRST_NAME.eq(firstName).and( PLAYER.ACTIVE.eq(true))
                              .fetch();
```

Joining

```
SELECT p.*
FROM players p INNER JOIN
     teams t ON p.team_id = t.id
```

Hibernate

```
List<Player> players = entityManager
    .createQuery("SELECT p FROM Player p JOIN Team t ON p.teamId = t.id",  Player.class)
    .getResultList();
```

jOOQ

```
List<Player> players = context.select()
                              .from(PLAYER)
                              .join(TEAM)
                              .on(PLAYER.TEAM_ID.eq(TEAM.ID))
                              .fetch();
```

Aggregates

```
SELECT COUNT(*)FROM players
```

Hibernate

```
int playerCount = entityManager
    .createQuery("SELECT COUNT(p) FROM Player p",  Player.class)
    .getSingleResult();
```

jOOQ

```
int playerCount = context.select(DSL.count())
                         .from(PLAYER)
                         .fetchOne(0, int.class);
```

Advanced Concepts

MariaDB

Associations - One to One


```
public class Team {
    public int id;
    public Player player;
}
```

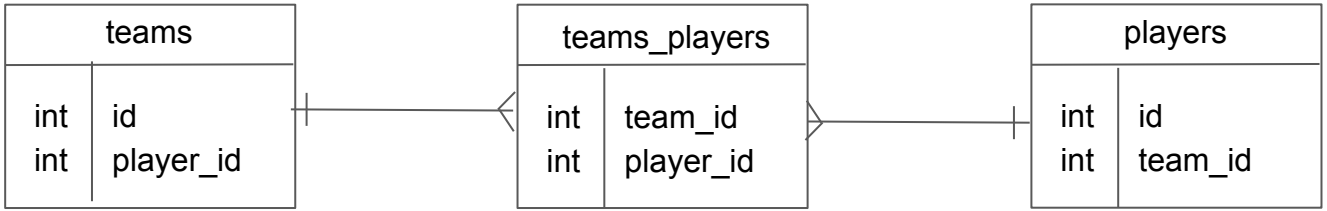
```
public class Player {
    public int id;
    public Team team;
}
```

Associations - One to Many


```
public class Team {
    public int id;
    public Player player;
}
```

```
public class Player {
    public int id;
    public List<Team> teams;
}
```

Associations - Many to Many

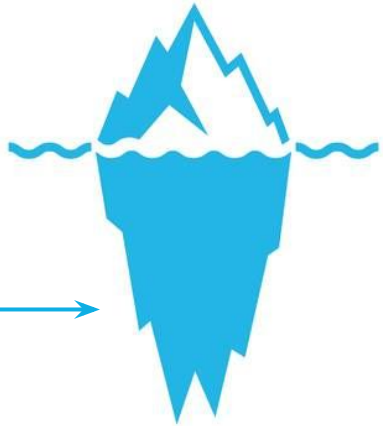



```
public class Team {
    public int id;
    public List<Player> player;
}
```

```
public class Player {
    public int id;
    public List<Team> teams;
}
```

So much more...

- ✓ Transactions
- ✓ Connection pooling
- ✓ Migration
- ✓ Schema generation
- ✓ Code first database schema generation
- ✓ Lazy/eager loading
- ✓ Caching
- ✓ Custom queries



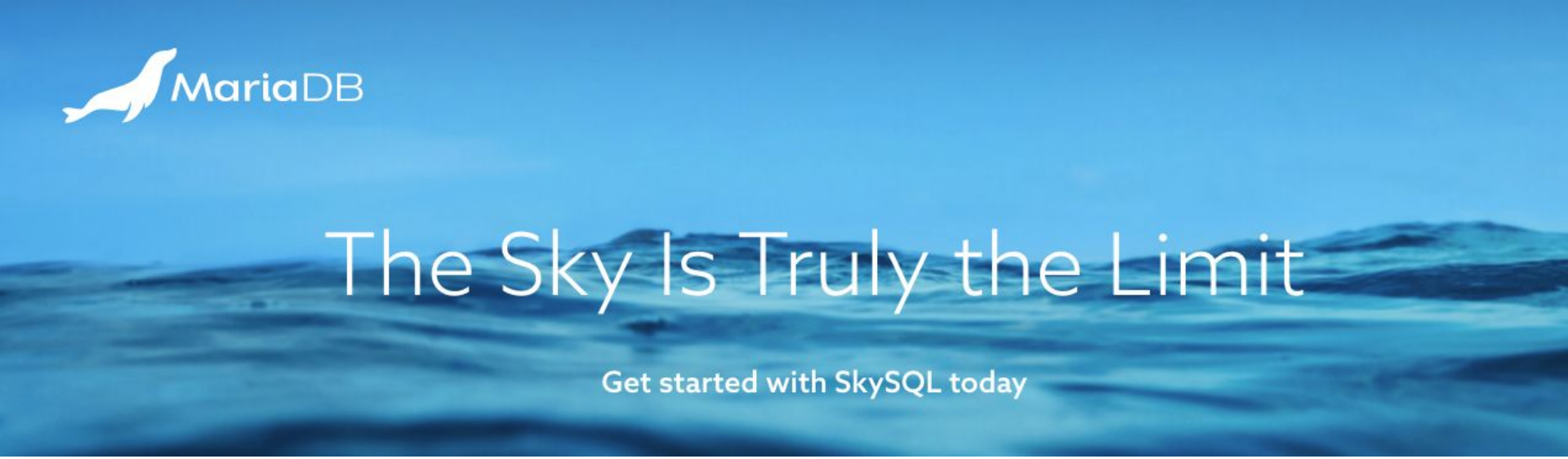
DEMO

MariaDB

So...should you use an ORM?

Advantages	Disadvantages

● Easier to write and maintain code
● Optimized SQL queries
● Multiple features out of the box
  ○ Transactions
  ○ Connection pooling
  ○ Migrations
● Built in SQL injection protection
● Database agnostic
	● Learning curve
	● SQL masters beware!
	● Negative performance impacts
	● Possible vendor lock
	● Too much abstraction
	● Support resources*
	● Cost*



$500 credit to get started

https://mariadb.com/products/skysql/get-started/

Thank you!

Open Source Developer Examples

https://github.com/mariadb-corporation/developer-examples


developers@mariadb.com


@mariadb


mariadb-corporation

MariaDB