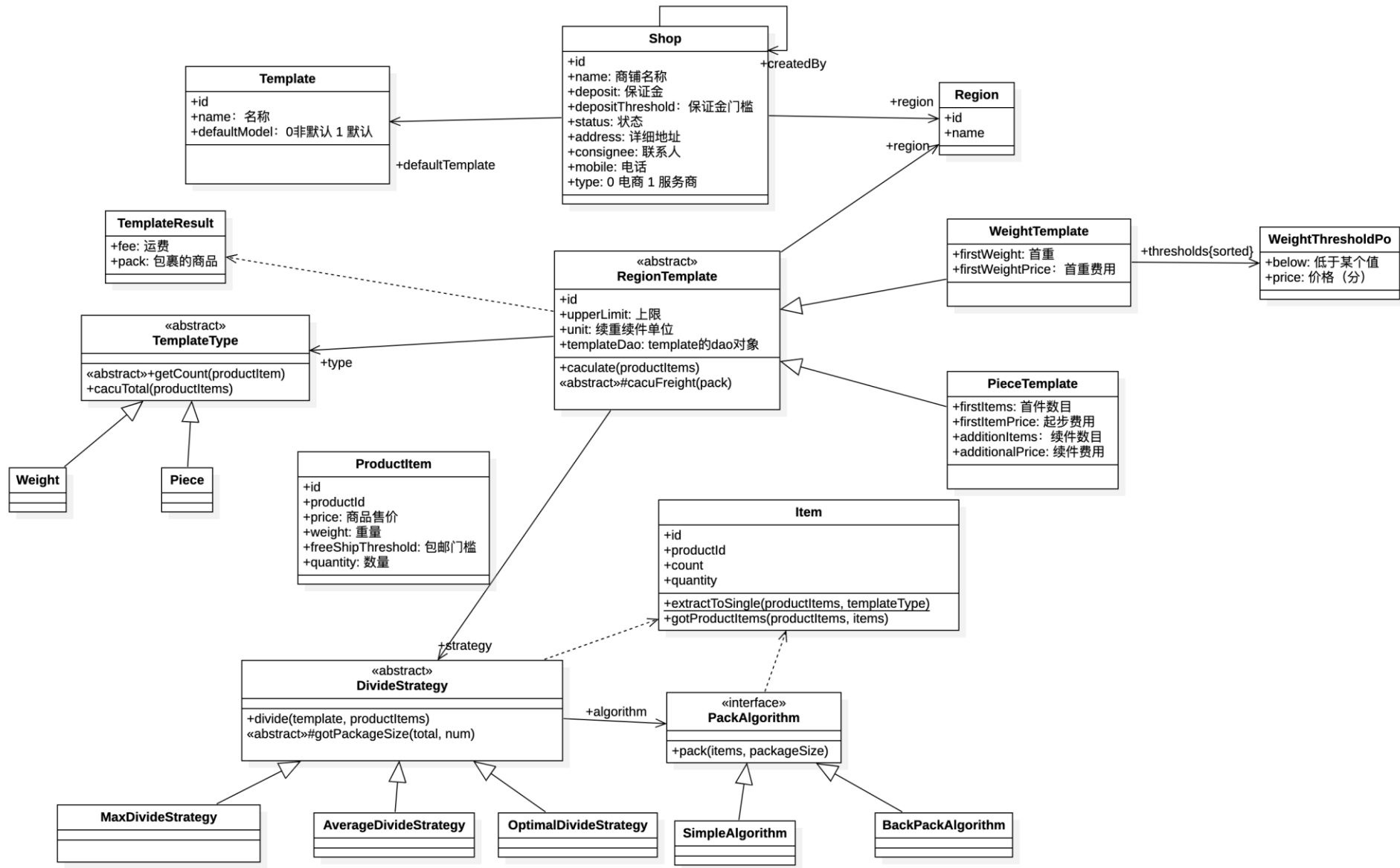


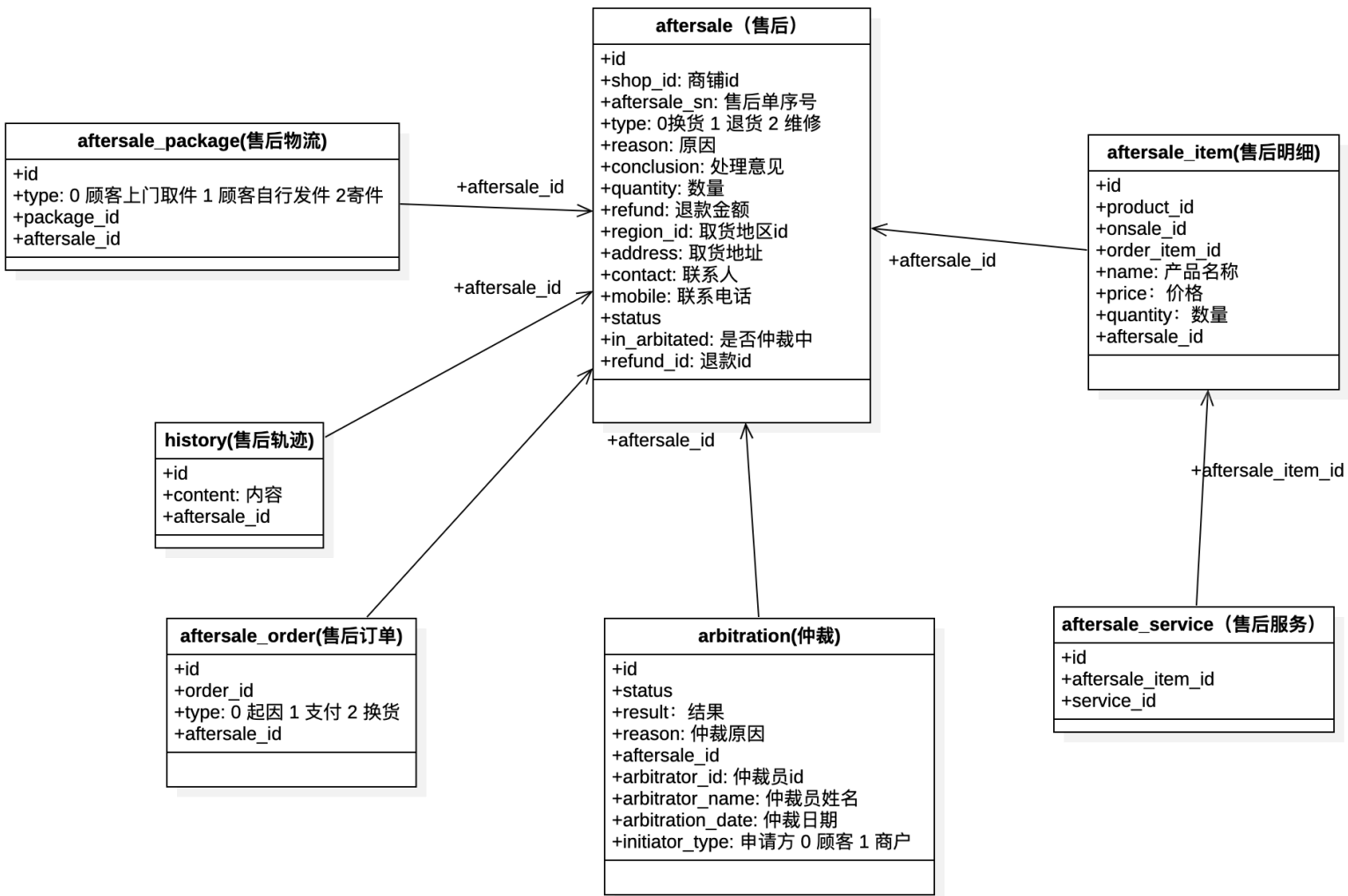
JavaEE平台技术 Spring Data 和MongoDB

邱明 博士

厦门大学信息学院

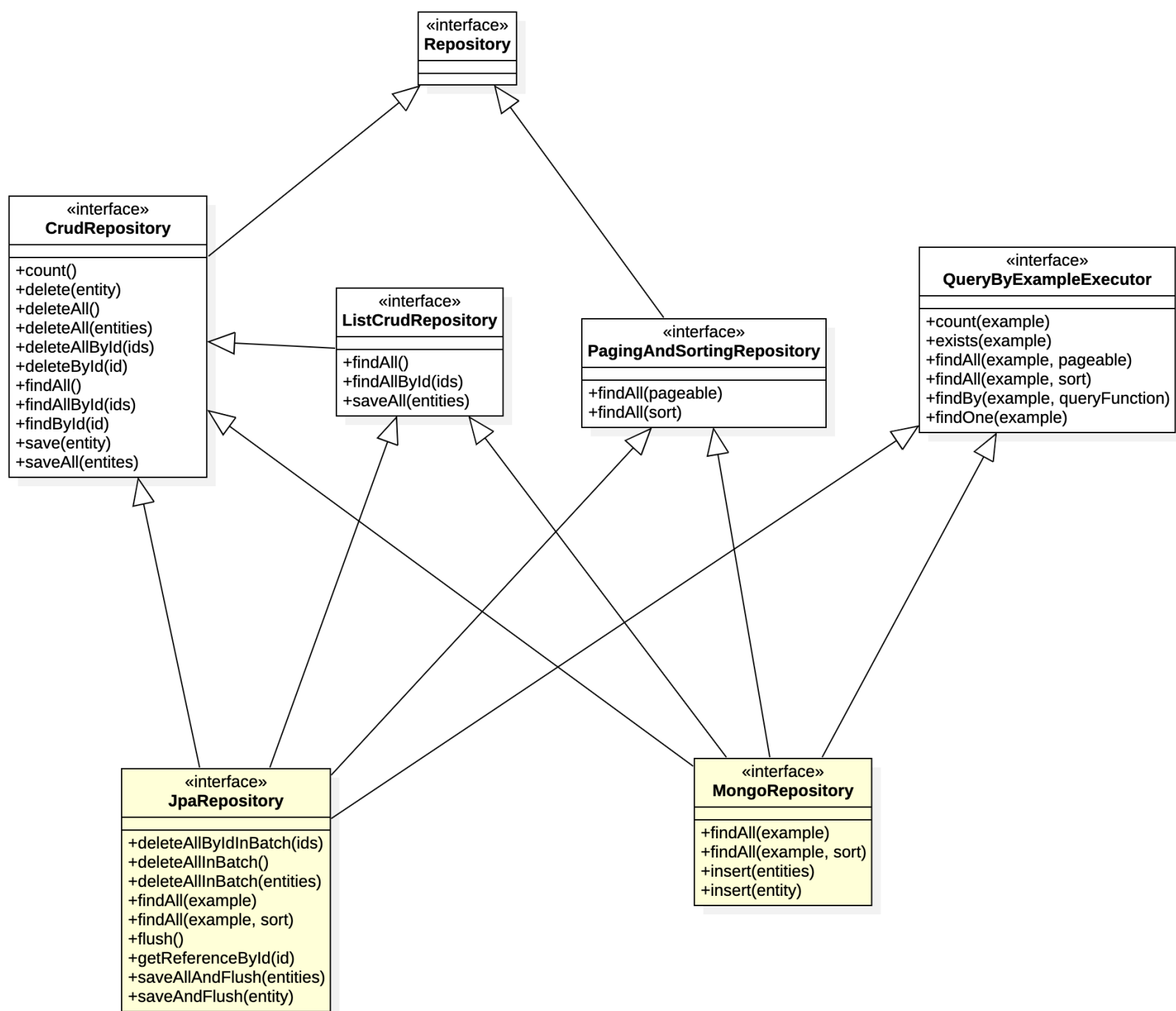
mingqiu@xmu.edu.cn





1. Spring Data Repositories

- 通过定义一个Repository接口
- 大幅度减少数据操作的格式代码 (boilerplate code)
 - MyBatis需要写mapper
 - Hibernate需要使用Session
 - JPA需要使用EntityManager
 - Spring Data只需要定义接口



2. Spring Data JPA

- 定义Entity

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "shop_service_product")
public class ShopServiceProductPo {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    /**
     * 创建者id
     */
    private Long creatorId;

    /**
     * 创建者
     */
    private String creatorName;
```

2. Spring Data JPA

- 定义和使用Repository

```
@Repository
public interface ShopServiceProductPoMapper extends JpaRepository<ShopServiceProductPo, Long> {
    Page<ShopServiceProductPo> findByShopId(Long shopId, Pageable pageable);
    Page<ShopServiceProductPo> findByShopIdEqualsAndInvalidEqualsAndBeginTimeBeforeAndEndTimeAfter(Long shopId, Byte invalid, LocalDateTime beginTime, LocalDateTime endTime, Pageable pageable);
}
```

```
public List<Product> retrieveByShopId(Long shopId, Integer page, Integer pageSize) throws RuntimeException{
    Pageable pageable = PageRequest.of(page, pageSize);
    Page<ShopServiceProductPo> ret = shopServiceProductPoMapper.findByShopId(shopId, pageable);
    return retrieveProduct(ret);
}
```

2. Spring Data JPA

- 定义和使用Repository

@Repository

```
public interface OnsalePoMapper extends JpaRepository<OnsalePo, Long> {  
    List<OnsalePo> findByProductIdEqualsAndEndTimeAfter(Long productId, LocalDateTime time, Pageable pageable);
```

```
    List<OnsalePo> findByProductIds(Long productId, Pageable pageable);
```

```
    List<OnsalePo> findByIdAndProductId(Long id, Long productId, Pageable pageable);
```

```
    @Query(value = "select o from OnsalePo o where o.productId = ?1 and ((o.beginTime >= ?2 and o.beginTime < ?3) or (o.endTime > ?2 and o.endTime <= ?3) or (o.beginTime <= ?2 and o.endTime >= ?3))")
```

```
    List<OnsalePo> findOverlap(Long productId, LocalDateTime beginTime, LocalDateTime endTime, Pageable pageable);
```

```
    List<OnsalePo> findByShopId(Long shopId, Pageable pageable);
```

```
    List<OnsalePo> findByShopIdAndProductId(Long shopId, Long productId, Pageable pageable);
```

```
    List<OnsalePo> findByProductIdAndInvalidEquals(Long productId, Byte invalid, Pageable pageable);
```

```
    List<OnsalePo> findByShopIdAndInvalidEquals(Long shopId, Byte invalid, Pageable pageable);
```

```
    List<OnsalePo> findByShopIdAndProductIdAndInvalidEquals(Long shopId, Long productId, Byte invalid, Pageable pageable);
```

```
    List<OnsalePo> findByInvalidEquals(Byte invalid, Pageable pageable);
```

```
    @Query(value = "select onsale from OnsalePo onsale JOIN ActivityOnsalePo activityOnsale on onsale.id = activityOnsale.onsaleId where activityOnsale.actId = :activityId")
```

```
    List<OnsalePo> findByActIdEquals(Long activityId, Pageable pageable);
```

```
}
```


2. Spring Data JPA

- 定义和使用Repository

```
@Repository
public interface ActivityPoMapper extends JpaRepository<ActivityPo, Long> {

    @Query(value = "select DISTINCT a from ActivityPo a join ActivityOnsalePo b on a.id = b.actId where b.onsaleId = :onsaleId")
    List<ActivityPo> findByOnsaleIdEquals(Long onsaleId, Pageable pageable);

    @Query(value = "select DISTINCT act from ActivityPo act JOIN ActivityOnsalePo actOnsale ON act.id = actOnsale.actId JOIN OnsalePo onsale ON actOnsale.onsaleId = onsale.id WHERE onsale.productId=:productId and act.actClass=:actClass")
    List<ActivityPo> findByActClassEqualsAndProductIdEquals(String actClass, Long productId, Pageable pageable);

    @Query(value = "select DISTINCT a from ActivityPo a join ActivityOnsalePo b on a.id = b.actId join OnsalePo c on c.id = b.onsaleId where c.productId = :productId and a.actClass = :actClass and c.endTime >=:beginTime and c.endTime <= :endTime and a.status = 1")
    List<ActivityPo> findValidByActClassEqualsAndProductIdEquals(String actClass, Long productId, LocalDateTime beginTime, LocalDateTime endTime, Pageable pageable);

    List<ActivityPo> findByActClassEqualsAndShopIdEqualsAndStatusEquals(String actClass, Long shopId, Integer status, Pageable pageable);

    @Query(value = "select DISTINCT a from ActivityPo a join ActivityOnsalePo b on a.id = b.actId join OnsalePo c on c.id = b.onsaleId where a.shopId = :shopId and a.actClass = :actClass and c.endTime >=:beginTime and c.endTime <= :endTime and a.status = 1")
    List<ActivityPo> findValidByActClassEqualsAndShopIdEquals(String actClass, Long shopId, LocalDateTime beginTime, LocalDateTime endTime, Pageable pageable);

    @Query(value = "select DISTINCT a from ActivityPo a join ActivityOnsalePo b on a.id = b.actId join OnsalePo c on c.id = b.onsaleId where a.actClass = :actClass and c.endTime >=:beginTime and c.endTime <= :endTime and a.status = 1")
    List<ActivityPo> findValidByActClassEquals(String actClass, LocalDateTime beginTime, LocalDateTime endTime, Pageable pageable);

    List<ActivityPo> findNEWByByActClassEqualsAndStatusEquals(String actClass, Integer status, Pageable pageable);

}
```

3 MongoDB

- 是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统。
- MongoDB以Collections管理数据，数据结构由键值(key=>value)对组成，将数据格式为Binary JSON。
- MongoDB采用B-tree实现数据的索引，支持辅助索引
- 支持分布式部署

3 MongoDB

- MongoDB发展里程碑

- 2009年2月，MongoDB1.0发布，实现了面向集合、模式自由、自由扩展的文档数据库。
- 2012年6月，MongoDB2.0.6发布，支持分布式文档数据库。
- 2015年3月，MongoDB 3.0.1 发布，包含WireTiger存储引擎，大幅度提升了MongoDB的写入性能。
- 2018年8月，MongoDB 4.0.2 发布，支持多文档的事务，成为第一个支持ACID的NoSQL数据库。
- 2019年10月，MongoDB 4.2.0 发布，支持分布式事务。

3 MongoDB

类型	部分代表	特点	用途
关系数据库	MySQL, Oracle, Microsoft SQL Server	关系数据库, 支持复杂的查询	用于存储业务数据, 方便查询
内存Key-Value数据库	Memcached Redis	采用内存存储数据, 可以通过key快速查询到其value。	用于做缓存和高并发应用
文档数据库	MongoDB CouchDB	一般用类似json的格式存储, 存储的内容是文档型的。这样也就有机会对某些字段建立索引, 实现关系数据库的某些功能。	用于存储非结构化的数据
大数据数据库	Hbase Cassandra Hypertable	支持大数据量和分布式存储, 是方便存储结构化和半结构化数据, 方便做数据压缩, 采用列存储的方式, 对针对某一系列或者某几列的查询有非常大的IO优势。	大数据应用

2 Mongo

关系数据库



Table

name	age
John	11
Tom	12

Mongo

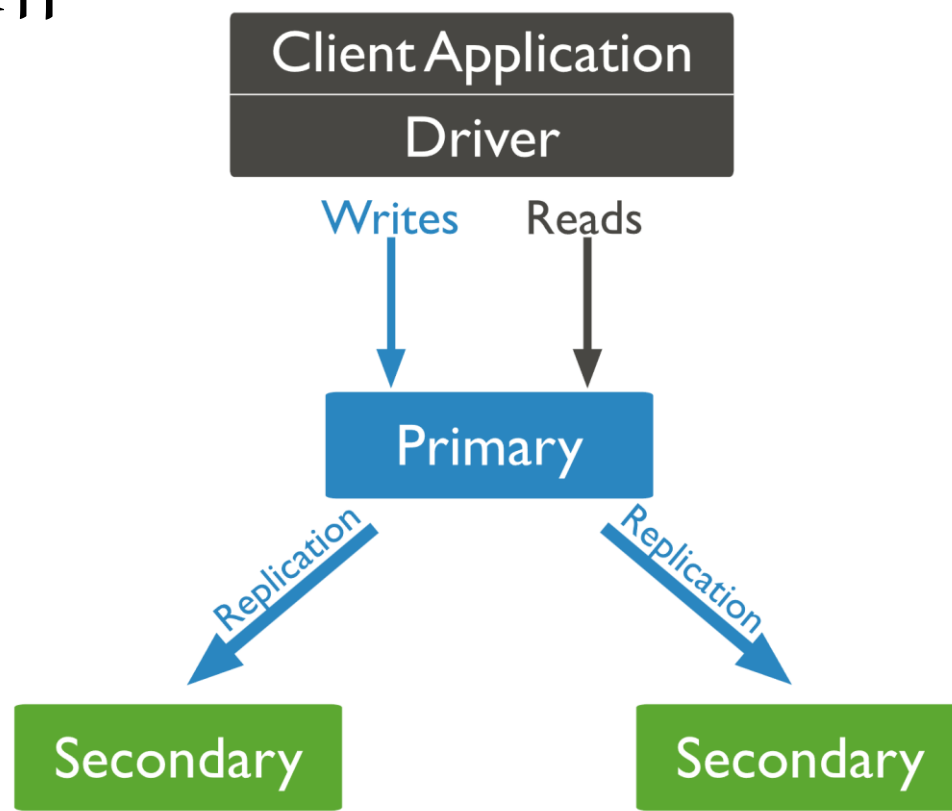


Collection

```
[
  {
    _id: ObjectId("637d81a0dbf61f104409fd64"),
    firstItem: 2,
    firstPrice: 10,
    additionalItems: 2,
    additionalPrice: 10
  },
  {
    _id: ObjectId("637d81addbf61f104409fd65"),
    firstItem: 5,
    firstPrice: 10,
    additionalItems: 10,
    additionalPrice: 10
  },
  {
    _id: ObjectId("637e0ed72a4e513f4a78dc68"),
    firstItem: 3,
    firstPrice: 11,
    additionalItems: 2,
    additionalPrice: 13
  }
]
```

2 Mongo

- 主节点 - 负责写操作
- 所有节点都可以读



3 Spring Data Mongo

- 定义Entity

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Document("pieceTemplate")
public class PieceTemplatePo {

    @MongoId
    private String objectId;

    private Integer firstItems;

    private Long firstPrice;

    private Integer additionalItems;

    private Long additionalPrice;
}
```

3 Spring Data Mongo

- 定义Repository

```
@Repository  
public interface PieceTemplatePoMapper extends MongoRepository<PieceTemplatePo, String> {  
}
```


3 Spring Data Mongo

- 使用Repository

```
public String save(RegionTemplate bo){  
    PieceTemplatePo po = cloneObj(bo, PieceTemplatePo.class);  
    PieceTemplatePo newPo = this.mapper.insert(po);  
    return newPo.getObjectId();  
}
```

```
public RegionTemplate getRegionTemplate(RegionTemplatePo po) {  
    PieceTemplate bo = cloneObj(po, PieceTemplate.class);  
    Optional<PieceTemplatePo> wPo = this.mapper.findById(po.getObjectId());  
    wPo.ifPresent(templatePo -> copyObj(templatePo, bo));  
    return bo;  
}
```

3 Spring Data Mongo

