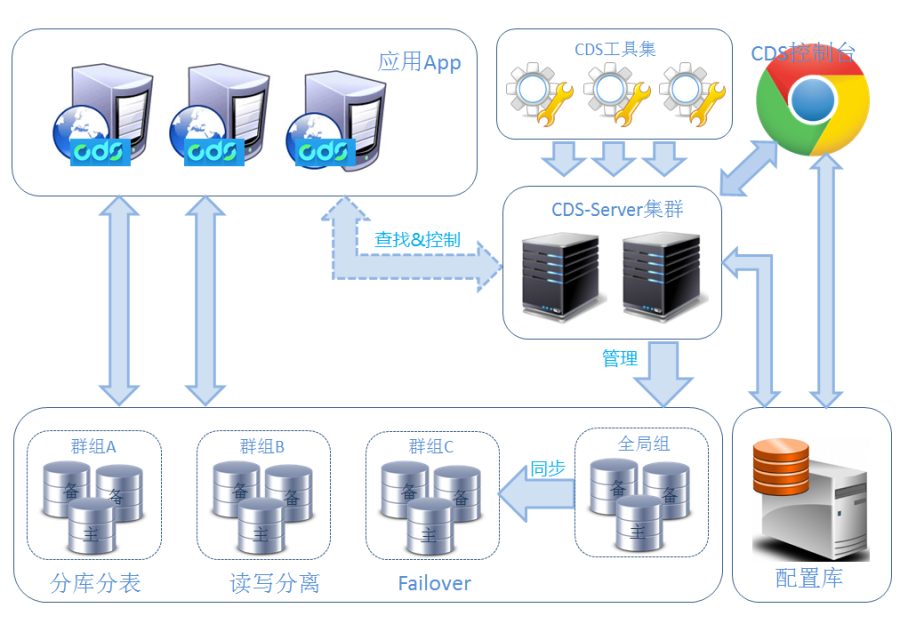
一、 CDS介绍

1.1 CDS框架

CDS(Completed Database Splitter)是一个数据库中间件,业务零成本接入,注重分库分表、读写分离、Failover，提供高性能、高可靠的海量数据存储服务，目前支持Oracle和Mysql数据库。  


整个CDS系统架构如图1-1所示，主要分为几个模块：

1. CDS Driver：实现JDBC标准API，轻量级分库分表服务，以Jar包方式集成在应用程序中；
2. CDS Server：高可用服务集群，相当于配置服务中心，对外提供一致性服务。CDS Driver驱动通过Http从CDS Server获取规则，CDS Console产生的应用事件也通过CDS Server推送到应用端；
3. CDS Console：CDS集群控制台，主要用于：集群管理、应用管理、数据迁移、数据同步、系统配置等功能；
4. CDS运维工具：Mydb、集群检查工具、全量数据迁移平台、增量数据同步作业。

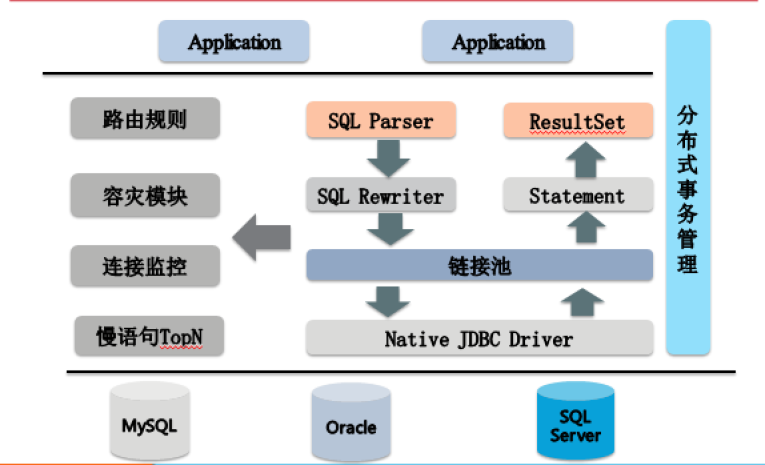
### 1.1.1 CDS集群



CDS集群主要包括几种业务表类型：

1. **全局业务表：**每个工作组都存在相同表结构和业务数据(比如：城市、类别）
2. **切分表：**数据量非常大，单表无法支撑，根据切分列进行水平拆分（业务流水表）
3. **孤立表：**数据量不大，不需要进行水平拆分，表结构可以存放在任意分库中，（比如.配置表）

### 1.1.2 CDS Driver



1.2 CDS功能

1. CDS分库分表路由规则，四种方式（hash、mod、range、suffix）;
2. 切换策略：failover、读写分离
3. 写库切换，主库失败，当前事务失败，然后failover到写备库；
4. 读写分离，读库自管理，不可用的读库自动排除掉，当前读事务自动failover到其它读库中；当读库恢复后，自动加入工作列表；按权重分配读事务。
5. 自定义路由算法,应用可根据实际的需求自定义路由规则，使用groovy实现对应算法，通过配置项动态载入；
6. 多分表结果集聚合：支持sum、max、min、count四个函数
7. 复合切分键：一个切分键由多列组成
8. 级联切分键：切分键包含多级切分键，每级切分键可由多列组成，切分规则可指向具体分表或者下一级切分键；
9. 查询切分键：只用于查询的切分键cl
10. Sequence统一管理：
11. Mysql:支持全局Sequence统一接口和分表Sequence接口
12. Oralce：支持全局Sequence统一接口和分表Sequence接口（支持Oracle原生Sequence接口）
13. 模拟执行，只输出重写后的sql，不会对数据库的操作产生任何变化
14. 分页实现，瀑布型式分页，支持缓存前一页的位置信息，提高分页查询效率
15. 重读/双写映射配置及变更推送
16. 驱动配置、增量规则在线推送
17. 自定义宏定义SQL，优化查询方式

## 2.3 CDS配置项说明

|  |  |  |
| --- | --- | --- |
| 配置名称 | 默认值 | 配置说明 |
| wycds.route2all | FALSE | 允许定位多个分表还是直接抛出异常 |
| wycds.sql | FALSE | 是否打开sql输出 |
| wycds.splittingtable | TRUE | 应用是否考虑分库分表，cds-driver是否解析sql |
| wycds.multitable.update | FALSE | 是否允许更新多个分表 |
| wycds.ddl.support | FALSE | 是否允许执行ddl |
| wycds.sql.rollback | FALSE | 模拟执行，回退所有事务 |
| wycds.parted.timeout | 60000 | 单个分表查询最长等待时间,单位：毫秒 |
| wycds.read2master | FALSE | 是否允许在全部读库失联后，路由到写库去读 |
| wycds.readwrite | FALSE | 写库可以作为读库，分流读事务 |
| wycds.seq | TRUE | insert语句是否使用cds的sequence获取方式 |
| wycds.seq.zero.replace | TRUE | insert语句sequence为0或空值时是否自动获取 |
| wycds.seq.fromworkgroup | FALSE | Sequence是否从各个群组的sequence表获取 |
| wycds.oracle.raw.sequence | FALSE | ORACLE集群是否使用原生的sequence |
| wycds.isolate.sequence | TRUE | 孤立表sequence查询直接到孤立表所在库的cluster\_autoinc\_table查询 |
| wycds.jdbclog | FALSE | 是否打开jdbc日志 |
| wycds.range.compare | FALSE | 允许切分键的范围比较表达式 |
| wycds.sql.appender | std | Sql日志输出方式：file/std |
| wycds.sql.filename |  | Sql输出方式为file时，日志路径 |
| wycds.emit.algorthm |  | 切分键算法前缀，值为关联的算法类包路径 |
| groovy. wycds.emit.algorthm |  | Groovy方式自定义切分键算法前缀，值为自定义的Groovy语法的自定义算法实现 |
| wycds.emit.suffix.algorthm |  | 分表后缀自定义算法前缀 |
| groovy.wycds.emit.suffix.algorthm |  | Groovy方式分表后缀自定义算法前缀 |
| wycds.emit.regex.eval.suffix.algorthm |  | 分表后缀表达式获取算法前缀 |
| groovy.wycds.emit.regex.eval.suffix.algorthm |  | Groovy方式分表后缀表达式获取算法前缀 |
| wycds.emit.all |  | 切分键列枚举值，如果查询条件缺少某个切分列时，枚举出该值的分表后缀，默认为全部分表后缀 |
| groovy.wycds.emit.all |  | Groovy方式实现切分键列枚举值，如果查询条件缺少某个切分列时，枚举出该值的分表后缀，默认为全部分表后缀 |
| wycds.emit.between |  | 切分键范围表达式定位算法 |
| groovy.wycds.emit.between |  | Groovy方式实现切分键范围表达式定位算法 |
| wycds.gather.sql.open | true | 是否开启指标采集 |
| wycds.gather.sql.period | 6000 | 指标采集频率，只有wycds.gather.sql.open设置为true时有效，单位毫秒 |
| **wycds.wycp.decoding** | **false** | **生产数据库密码默认用wangyincp加密，如果用其他连接池，可以把这个值设成true完成解密，access denied for user 'xxx'@'xxx'** |

## 2.4 CDS SQL支持

### 2.4.1 不支持的sql

1. update 不允许修改切分键的值。说明：修改切分键会路由失败，无法查询该行数据
2. where条件中切分键表达式和其他表达式使用or连接。说明：该语句只会根据切分键表达式定位，无法根据另一个表达式定位分表，不能保证结果集聚合的正确性。
3. 不支持distinct函数。
4. update\delete 不支持 limit,如果要批量更新建议参考[3.10](http://wiki.cbpmgt.com/confluence/pages/viewpage.action?pageId=20585602)。

### 2.4.2 需特殊处理的的sql

1. union语句。说明：union的语句必须包含切分键且定位到相同分表中; 需要通过cds提供的api预定义该sql对应的切分键信息

### 2.4.3 SQL书写规约

1. 字符串用单引号
2. 表名，字段名等用反引号
3. Oracle大小写冲突用双引号
4. 只有在Oracle里面才会强制转化大小写，MySQL实际上是区分大小写的
5. MySQL里面双引号引用字符串在某些版本中会报错，为了避免错误，尽量使用单引号来操作字符串
6. 即使表名和字段名不是你所知的数据库保留字，尽可能也养成加上反引号的好习惯，这也可以避免错误，保证开发的顺利进行

## 2.5 应用接入CDS流程

### 2.5.1水平拆分方案规划

#### 2.5.1.1 开发DBA做应用分表规划

业务开发提交应用单表数量、数据增长速率、数据表的关联关系以及数据库访问压力等参数给开发DBA，开发DBA决定拆分的数据库和表的数目。

开发DBA联系，姚文斌或王志民

#### 2.5.1.2 应用SQL审核及改造

开发DBA审核应用的SQLMAP文件，分析应用SQL是否符合CDS规范；分析SQL性能，结合CDS的限制进行SQL改造。

#### 2.5.1.3 确定分库分表方案

开发DBA根据业务需求，确定以下问题：

1. 哪些业务表需要进行拆分，根据什么字段进行拆分；
2. 选择哪种拆分方式；
3. 哪些表之间有关联关系，拆分粒度如何确定；
4. 集群Failover、读写分离方案；
5. 集群群组主备数据库分配；
6. 确定应用数据迁移和数据同步的方案；

确定方案后，在CDS中配置数据库集群、集群数据库机器以及分表规则策略。

#### 2.5.1.4 搭建测试环境

开发DBA准备好测试环境要用的数据库，初始化分表结构。

#### 2.4.1.5 应用接入测试

应用进行代码改造，接入CDS集群，在测试环境进行业务测试。

## 3.2 常用接入案例

### 3.2.1 哈希/取模常用场景

由于一致性哈希路由并不均匀，故默认情况下建议使用取模的方式。字符串取mod时，先hash后mod，如果不需要先进行哈希，可以配置属性：wycds.emit.char.hashaftermod=false。  
取模为常用分库分表方法，可以使数据均匀分配。但对于切分键为时间的场景不建议取模。  
常用的按pin取模切分。

### 3.2.2 范围常用场景

常用场景：

1. 按时间做数据分片，比如有的业务要一个月一个表，查询的时候带上时间。这种情况就可以把时间字段按范围做切分。如果需要使数据均匀分配，可以考虑再加一级路由，第二级用pin或其它id字段做取模即可。
2. 新老库并存的情况：比如有的业务新接入CDS，需要保留对老库的操作，则可以取某个时间点，该时间点之前的操作都路由到老库，之后的数据路由到新库。

