## OLTP与OLAP



联机事务处理OLTP（on-line transaction processing） 主要是执行基本日常的事务处理，比如数据库记录的增删查改。比如在银行的一笔交易记录，就是一个典型的事务。   
OLTP的特点一般有：   
1.实时性要求高。我记得之前上大学的时候，银行异地汇款，要隔天才能到账，而现在是分分钟到账的节奏，说明现在银行的实时处理能力大大增强。   
2.数据量不是很大，生产库上的数据量一般不会太大，而且会及时做相应的数据处理与转移。   
3.交易一般是确定的，比如银行存取款的金额肯定是确定的，所以OLTP是对确定性的数据进行存取   
4.高并发，并且要求满足ACID原则。比如两人同时操作一个银行卡账户，比如大型的购物网站秒杀活动时上万的QPS请求。

联机分析处理OLAP（On-Line Analytical Processing） 是数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。典型的应用就是复杂的动态的报表系统。

OLAP的特点一般有：   
1.实时性要求不是很高，比如最常见的应用就是天级更新数据，然后出对应的数据报表。   
2.数据量大，因为OLAP支持的是动态查询，所以用户也许要通过将很多数据的统计后才能得到想要知道的信息，例如时间序列分析等等，所以处理的数据量很大;   
3.OLAP系统的重点是通过数据提供决策支持，所以查询一般都是动态，自定义的。所以在OLAP中，维度的概念特别重要。一般会将用户所有关心的维度数据，存入对应数据平台。

总结：   
OLTP即联机事务处理，就是我们经常说的关系数据库，增删查改就是我们经常应用的东西，这是数据库的基础；TPCC(Transaction Processing Performance Council)属于此类。   
OLAP即联机分析处理，是数据仓库的核心部心，所谓数据仓库是对于大量已经由OLTP形成的数据的一种分析型的数据库，用于处理商业智能、决策支持等重要的决策信息；数据仓库是在数据库应用到一定程序之后而对历史数据的加工与分析，读取较多，更新较少，TPCH属于此类。   
随着大数据时代的到来，对于OLAP，列存储模式或者说nosql模式比传统意义的行存储模式可能更具优势。

## [DDL,DML,DCL,TCL](http://www.cnblogs.com/henryhappier/archive/2010/07/05/1771295.html)

1.**DDL**（[**Data**](javascript:;)**Definition Language**）[**数据库**](javascript:;)**定义语言**statements are used to define the database structure or schema.

DDL是[**SQL**](javascript:;)语言的四大功能之一。  
用于定义数据库的三级结构，包括外模式、概念模式、内模式及其相互之间的映像，定义数据的完整性、安全控制等约束  
DDL不需要commit.  
CREATE  
ALTER  
DROP  
TRUNCATE  
COMMENT  
RENAME  
  
2.**DML**（**Data Manipulation Language**）**数据操纵语言**statements are used for managing data within schema objects.

由DBMS提供，用于让用户或程序员使用，实现对数据库中数据的操作。  
DML分成交互型DML和嵌入型DML两类。  
依据语言的级别，DML又可分成过程性DML和非过程性DML两种。  
需要commit.  
SELECT  
INSERT  
UPDATE  
DELETE  
MERGE  
CALL  
EXPLAIN PLAN  
LOCK TABLE  
  
3.**DCL**（**Data Control Language**）**数据库控制语言**  授权，角色控制等  
GRANT 授权  
REVOKE 取消授权  
  
4.**TCL**（**Transaction Control Language**）**事务控制语言**  
SAVEPOINT 设置保存点  
ROLLBACK  回滚  
SET TRANSACTION

## ACID

**ACID**，是指[数据库管理系统](https://zh.wikipedia.org/wiki/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F)（[DBMS](https://zh.wikipedia.org/wiki/DBMS)）在写入或更新资料的过程中，为保证[事务](https://zh.wikipedia.org/wiki/%E6%95%B0%E6%8D%AE%E5%BA%93%E4%BA%8B%E5%8A%A1)（transaction）是正确可靠的，所必须具备的四个特性：[原子性](https://zh.wikipedia.org/w/index.php?title=%E5%8E%9F%E5%AD%90%E6%80%A7&action=edit&redlink=1)（atomicity，或称不可分割性）、[一致性](https://zh.wikipedia.org/wiki/%E4%B8%80%E8%87%B4%E6%80%A7)（consistency）、[隔离性](https://zh.wikipedia.org/wiki/%E9%9A%94%E9%9B%A2%E6%80%A7)（isolation，又称独立性）、[持久性](https://zh.wikipedia.org/w/index.php?title=%E6%8C%81%E4%B9%85%E6%80%A7&action=edit&redlink=1)（durability）。

* 原子性：一个事务（transaction）中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被恢复（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。
* 一致性：在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设规则，这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作。
* 隔离性：数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括读未提交（Read uncommitted）、读提交（read committed）、可重复读（repeatable read）和串行化（Serializable）。
* 持久性：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。

## sql 语句 replace

保证幂等性，即多次重复操作和一次操作效果一样。

如：

1. **replace into**replace into table (id,name) values('1','aa'),('2','bb')   
   此语句的作用是向表table中插入两条记录。如果主键id为1或2不存在   
   就相当于   
   insert into table (id,name) values('1','aa'),('2','bb')   
   如果存在相同的值则不会插入数据
2. **replace(object,search,replace)**  
   把object中出现search的全部替换为replace

例：把表table中的name字段中的aa替换为bb   
  
update table set name=replace(name,'aa','bb')

## Hamal迁移工具

对增量迁移，DDB实现了一套独立的迁移工具Hamal来订阅各个数据节点的增量更新，Hamal内部又依赖DBI模块将增量更新Apply到新库新表，如图8所示。

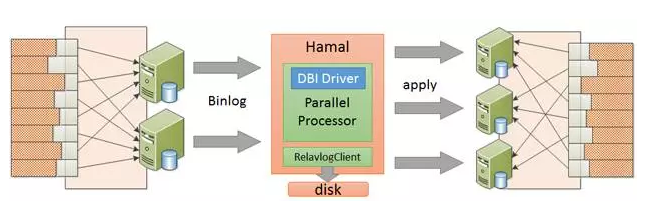


图8

Hamal作为独立服务，与Proxy一样由DDB统一配置和管理，每个Hamal进程负责一个数据节点的增量迁移，启动时模拟Slave向原库拉取Binlog存储本地，之后实时通过DBI模块Apply到新库新表，除了基本的迁移功能外，Hamal具备以下两个特性：

1. 并行复制：Hamal的并行复制组件，通过在增量事件之间建立有向无环图，实时判断哪些事件可以并行执行，Hamal的并行复制与MySQL的并行复制相比快10倍以上；
2. 断点续传：Hamal的增量Apply具有幂等性，在网络中断或进程重启之后可以断点续传。

## NDC本地部署

### 1.center本地部署

1.1建立元数据库（MySQL数据库）

/dts-center/src/main/resources/script/create\_tables.sql



1.2修改 数据库配置

**diff --git a/center/src/main/resources/dbPool.xml b/center/src/main/resources/dbPool.xml**

**index d610b99a..ab959a71 100644**

**--- a/center/src/main/resources/dbPool.xml**

**+++ b/center/src/main/resources/dbPool.xml**

@@ -1,8 +1,8 @@

<?xml version="1.0" encoding="UTF-8" ?>

<dbPool>

- <url>jdbc:mysql://10.164.96.168:3306/dts\_zjs</url>

- <user>readwrite</user>

- <passwd>3221631</passwd>

+ <url>jdbc:mysql://127.0.0.1:3306/testdb</url>

+ <user>ndc</user>

+ <passwd>ndc</passwd>

<!-- no setting use default -->

<connectTimeout>5000</connectTimeout>

<socketTimeout>10000</socketTimeout>

1.3 运行as java application

程序入口为/dts-center/src/main/java/com/netease/dts/center/DTSContainer.java



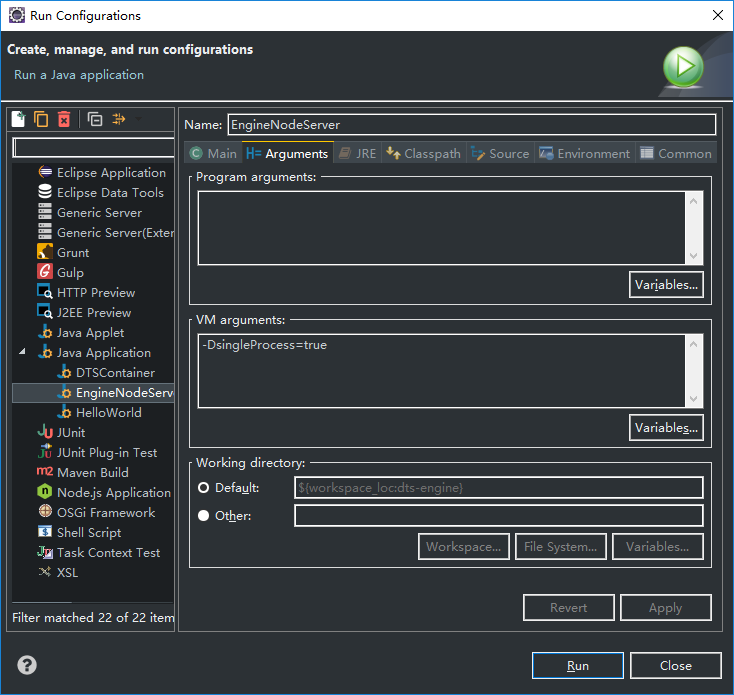
### 2.engine本地部署

2.1程序入口

/dts-engine/src/main/java/com/netease/dts/engine/server/EngineNodeServer.java



* 1. 运行



### dashboard本地部署

D:\workspace\dts-dashboard\dashboard目录下执行

mvn -DskipTests jetty:run

or

mvn jetty:run

修改用户信息数据库

**--- a/dashboard/src/main/resources/dbPool.xml**

**+++ b/dashboard/src/main/resources/dbPool.xml**

@@ -1,9 +1,9 @@

<?xml version="1.0" encoding="UTF-8" ?>

<dbPool>

<!--<url>jdbc:mysql://127.0.0.1:3306/ndc\_dashboard</url>-->

- <url>jdbc:mysql://127.0.0.1:3306/ndc\_dashboard</url>

- <user>dts</user>

- <passwd>dtsdts</passwd>

+ <url>jdbc:mysql://127.0.0.1:3306/test\_dashboard</url>

+ <user>ndc</user>

+ <passwd>ndc</passwd>

<!-- no setting use default -->

<connectTimeout>5000</connectTimeout>

<socketTimeout>10000</socketTimeout>

## 4.测试