# 图数据库

## 定义

Graph db:

* 1. 每个节点保存邻居的指针，无需额外的adjacency index
  2. 存在三种元素 nodes edges properties

数据库分类：

[http://markorodriguez.com/2011/04/30/putting-and-getting-data-from-a-database/]

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Primitive store |  | // put data  db.put('marko');  db.put(31);  db.put(true);  // get data  Iterator results = db.get();  Iterator filteredResults = db.get{it.startsWith('ma')}; |
| Key value store |  | // put data  db.put('name', 'marko');  db.put('age', 31);  // get data  Object value = db.get('name'); |
| Document store | Json/xml | db.put(document);  // get data  Document result = db.findOne({      type : "person",      skills.languages : "R"  }); |
| Graph sotre | 两种查询方式：pattern match query / traversing | Vertex v = db.putVertex({'name' : 'marko', 'age' : 31});  Vertex u = db.putVertex({'language' : 'gremlin'});  db.putEdge(v, 'hasSkill', u); |

The property graph model

|  |  |  |
| --- | --- | --- |
| Vertex | * each vertex has a collection of properties defined by a map from key to value. |  |
| Edge |  |  |
|  |  |  |

RDF

[http://www.linkeddatatools.com/introducing-rdf]

## Orientdb 介绍

Orientdb 是开源的graph & document 数据库。符合APACHE2协议。分为社区版和企业版。区别如下：

|  |  |  |
| --- | --- | --- |
|  | Community Edition | Enterprise Edition |
| License | Open Source [Apache2 License](http://www.apache.org/licenses/LICENSE-2.0.html) | Commercial License |
| Embedded, Remote and In-Memory Storage | √ | √ |
| [Console Tool](https://github.com/nuvolabase/orientdb/wiki/Console-Commands) | √ | √ |
| [Studio Visual Tool](https://github.com/orientechnologies/orientdb-studio/wiki) | √ | √ |
| [High Availability](https://github.com/nuvolabase/orientdb/wiki/Distributed-Architecture) Replication, Sharing and Fail-over | √ | √ |
| [TinkerPop Stack](https://github.com/nuvolabase/orientdb/wiki/Graph-Database-Tinkerpop) (Blueprints, Gremlin, Pipes, SparQL) | √ | √ |
| Metric Recording for Historical Analysis (Monitor and tune your servers for maximum performance) |  | √ |
| Query Profiler (Locate 80% of the bottlenecks in your application) |  | √ |
| Configurable Alerts (Email alerts or executing HTTP requests) |  | √ |
| Hot-fix and New Release Notifications |  | √ |
| Centralised Online Backups |  | √ |
| [Production Support](http://www.orientechnologies.com/support/) |  | √ |
|  |  |  |

参考

【<http://www.orientechnologies.com/orientdb-enterprise/】>

【<http://www.orientechnologies.com/docs/last/orientdb.wiki/Tutorial-Installation.html>】

## PropertyGraph 介绍

# ORIENT DB

## 安装

必须：

JDK1.6+

### 从原代码安装

//clone remote master 分支到本地

git clone https://github.com/nuvolabase/orientdb.git

cd orientdb

//创建本地develop分支

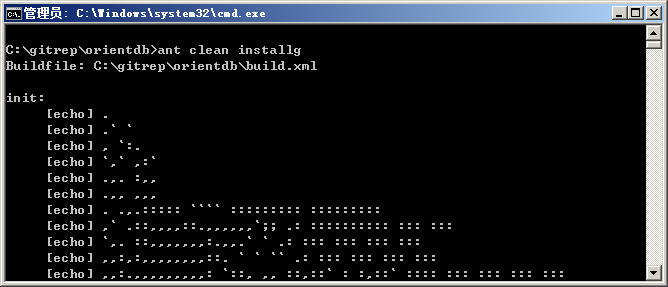
git checkout -b develop

//将远程develop分支 merge 到本地develop分支

git pull origin develop

//执行构建

ant



项目列表

|  |  |
| --- | --- |
| 项目 | 依赖于 |
| commons |  |
| core | nativeos commons |
| nativeos | commons |
| object | core ，数据库对象 |
| enterprise | core |
| client | enterprise |
| server | client |
| distributed | server |
| tools | common core client object |
| graphdb | server enterprise tools commons core |
| tests |  |

Ant 构建目标(base-build.xml)

Install -> jar -> compile ->init

|  |  |
| --- | --- |
| init | make build directory |
| compile | make lib dir, 调用javac compile源码，copy resources到build 目录 |
| jar | make dist directory，调用jar命令将build目录编译结果打包的分发目录，将依赖于project lib的jar也copy到分发目录 |
| install | 将分发目录内容copy到global lib |

### 构建产物

Build完后产生如下目录，

|  |  |  |
| --- | --- | --- |
| benchmarks | 性能测试 |  |
| bin | Scripts to run Orient server and console |  |
| config | Configuration files |  |
| Databases | Default place to store databases |  |
| lib | Compiled Java archives |  |
| log | Server log files |  |
| plugins | Additional plugins like [Studio web tool](https://github.com/orientechnologies/orientdb-studio/wiki) and [Lucene Indexes](https://github.com/orientechnologies/orientdb-lucene/wiki) |  |
| www | Web static contents |  |

参考

【<https://github.com/orientechnologies/orientdb/wiki/Installation】>

## 运行

### 启动服务

启动位于bin目录下的 Server.bat

### 控制台

运行位于bin目录下的 console.bat 管理控制台

链接orientdb 默认数据库

orientdb> connect plocal:../databases/GratefulDeadConcerts admin admin

查看数据信息

orientdb> info

## 结构

### 文件系统

OFile

OFileClassic

OFile

DataSegment

由一个或多个file 组成，

File接口用于底层存储字节流，对应有经典file实现和MemoryMap实现。

### Database

Create db流程

ODatabaseDocumentTx

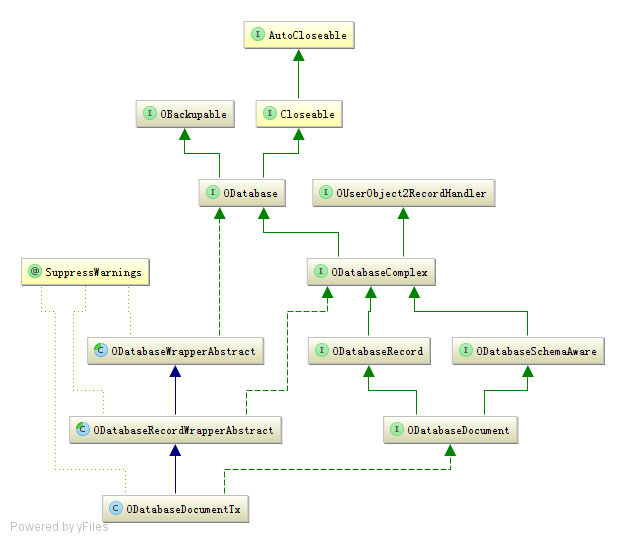
ODatabaseWrapperAbstract

ODatabaseRecordAbstract

ODatabaseWrapperAbstract

ODatabaseRaw

OStorageLocal



### ODatabaseWrapperAbstract

用了对接口ODatabase进行包装

ODatabaseDocumentTx

underlying-> ODatabaseRecordTx

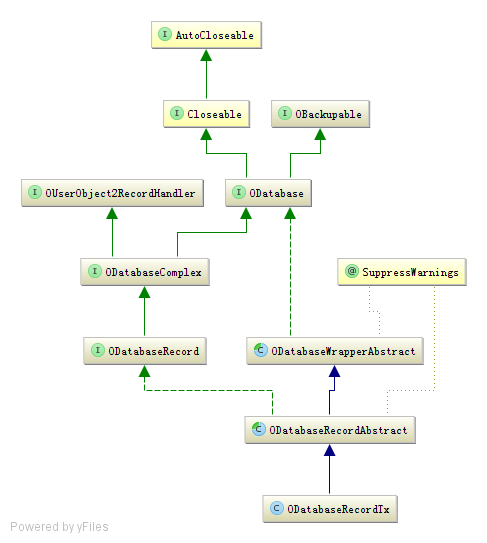
ODatabaseRecordWrapperAbstract

对接口ODatabaseRecord进行包装

ODatabaseRecordAbstract

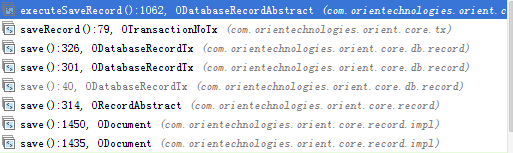
Underlying-> ODatabaseRaw

ODatabaseRaw 为最低级别的databse实现，被其它extend 或 wrap



#### Save调用栈

ODocument.save 调用如下



### Record

记录，记录由Header和内容组成，记录存储在segment中。

|  |  |  |
| --- | --- | --- |
| 2 Bytes cluster id | Cluster Position serialized size | content |

### Cluster

用于保存记录逻辑地址到物理隐射表，保存记录的physical position

# 使用ORIENTDB

前置条件，启动服务，启动控制台。

## 用户密码

默认超级用户位于config/orientdb-server-config.xml，使用明文存储。

## 基础概念

|  |  |  |
| --- | --- | --- |
| class | 相当于table |  |
| property | 等同于field |  |
| records/documents/vertices |  |  |
| cluster（分为physical和memory） | 记录存储的地方 like partition | class: cluster 1:n,缺省1:1 |
| RecordID | 每个记录都有的unique id | #<cluster-id>:<cluster-position> |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| browse class | Browse class ouser |  |
| browse cluster | Browse cluster OUser | 等同上条命令 |
| load record | load record #12:4 |  |
| create class | create class Person extends V  create class Eat extends E |  |
| create vertex | create vertex Person set name = 'Luca' |  |
| create edge | create edge Eat from (select from Person where name = 'Luca') to (select from Restaurant where name = 'Dante')  create edge Eat from #11:1 to #12:0 |  |
| create database | create database plocal:../databases/cars admin admin plocal |  |
|  |  |  |

## API 操作

【<http://www.orientechnologies.com/docs/last/orientdb.wiki/Tutorial-Java.html】>

# BIGP项目

## 涉及产品

|  |  |  |
| --- | --- | --- |
| Orientdb | orientdb-community-1.7.9 |  |
| rexster-server | 2.6.0 |  |
| nutch | Nutch 1.9 | 12/08/2014 |
|  |  |  |
|  |  |  |

## 结构

1. ***Connect localhost root root***

[connect remote:localhost/GratefulDeadConcerts admin admin]

[create database plocal:../databases/bigp admin admin plocal]

1. ***create database plocal:../databases/bigp***
2. ***create class Person extends V***

CREATE PROPERTY <class>.<property> <type>

create property Person.name string

create property Person.url string

create property Person.img string

create property Person.w\_address string

create property Person.h\_address string

create property Person.tel string

create property Person.birthdate string

create property Person.occupation string

|  |  |  |
| --- | --- | --- |
| name | string |  |
| sex | string |  |
| url | string |  |
| img | string |  |
| w\_address | string |  |
| h\_address | string |  |
| tel | string |  |
| birthdate | string | yyyymmdd |
| occupation | string |  |

# TinkerPop-Graph-Stack

## [gremlin](https://github.com/tinkerpop/gremlin)

Gremlin is a domain specific language for traversing property graph

## rexster

Rexster  is a graph server that exposes any [Blueprints](http://blueprints.tinkerpop.com/) graph through [REST](http://en.wikipedia.org/wiki/Representational_State_Transfer) and a binary protocol called [RexPro](https://github.com/tinkerpop/rexster/wiki/RexPro).

### Rexster Server 启动

./bin/rexster.sh --start

Bin\rexster.bat –s

### 问题？

[WARN] GraphConfigurationContainer - GraphConfiguration [com.tinkerpop.rexster.OrientGraphConfiguration] could not instantiate a class [com/orientechnologies/orient/core/config/OGlobalConfiguration]. Ensure that it is in Rexster's path.com.tinkerpop.rexster.config.GraphConfigurationException: GraphConfiguration [com.tinkerpop.rexster.OrientGraphConfiguration] could not instantiate a class [com/orientechnologies/orient/core/config/OGlobalConfiguration]. Ensure that it is in Rexster's path.

原因 缺少jar包。

## Rexster graph conf

1. copy orient-client.jar and orient-enterprise.jar in$REXSTER\_HOME/ext

拷贝如下jar文件

|  |  |
| --- | --- |
| orientdb-tools |  |
| orientdb-enterprise |  |
| orientdb-server |  |
| orient-commons |  |
| orientdb-core |  |
| orientdb-graphdb | 其依赖上面项 |
|  |  |

在rexster conf目录文件rexster.xml中增加如下配置

<graph>

<graph-enabled>true</graph-enabled>

<graph-name>orientGraph</graph-name>

<graph-type>com.tinkerpop.rexster.OrientGraphConfiguration</graph-type>

<graph-location>remote:localhost/bigp</graph-location>

<properties>

<username>admin</username>

<password>admin</password>

</properties>

</graph>

参考如下：

【http://www.orientechnologies.com/docs/last/orientdb.wiki/Rexster.html】

【<https://github.com/tinkerpop/rexster/wiki/Specific-Graph-Configurations>】

## Blueprint

Blueprints is a collection of interfaces, implementations, ouplementations, and test suites for the property graph data model.

## Graph Visualisation

The default [Neo4j Server](http://neo4j.com/download) has a powerful, customizable data visualization tool based on the built-in D3.js

Graph visualisations are a powerful tool to convey the content of a graph. They can highlight patterns, show clusters and connections. There are many excellent options for graph visualization, such as [the D3.js](http://d3js.org/), [three.js](http://threejs.org/) and [sigma.js](http://sigmajs.org/).

[http://neo4j.com/developer/guide-data-visualization]

## Rexster Web Server

提供rest 接口，提供图的访问能力。

## Dog House

Rexster’s Dog House is a browser-based interface to Rexster

Basic graph visualization with [JavaScript InfoVis Toolkit](http://thejit.org/).

## Rexster console

Rexster Console is a [REPL](http://en.wikipedia.org/wiki/Read-eval-print_loop) similar to the REPL available from [Gremlin](https://github.com/tinkerpop/gremlin/wiki/Getting-Started)

# ORIENTDB 结构

## Server

OServer

Startup

activate

## Instance

可以注册多个Engine

## Engine

仅对storage进行包装，提供创建storage的能力，同时提供engine名称。关系为1对1.

Engine

Create

Storage

支持4种storage

|  |  |
| --- | --- |
| plocal | page storage ，数据访问在同一个jvm |
| remote | by using the network to access a remote storage |
| memory | 数据在内存 |
| local | 第一版存储，已被plocal取代 |

## 事务、锁与MVCC

## 索引

## 缓存

## DB

### ODatabase接口

最基础、原始的数据库访问接口。

### ODatabaseRaw

实施ODatabase接口，提供最基础的数据操作功能。所有的wrapper等都基于其之上。

其仅提供单条记录操作包装（增加、修改、删除）

### ODatabaseComplex接口

Extends ODatabase 提供高级别的操作接口

### ODatabaseRecord

Extends ODatabaseComplex 提供记录级操作接口

### ODatabaseDocument

Extends ODatabaseRecord 提供文档级操作接口

## 存储

提供基于key、value的键值对操作API(增、删、改)

### OStorageLocal 本地存储

本地存储通过文件接口来实现。

记录ID 为ORecordId其由clusterId、clusterPosition组成，记录内容为字节流。

clusterID为int型，clusterPosition可转换为long型。ORecordId为逻辑地址。

#### 锁

其在进行增加、修改、删除记录操作时，获取存储级别排斥所。

#### ？

为何需要数据记录逻辑地

逻辑物理地址对照保存在cluster中，可以通过cluster迭代所有记录。

本地实现时读取磁盘没有Page缓存，

page缓存在paginatedCluster中实现

#### 版本

对记录进行操作时，均需指定记录版本。

数据的版本存放在OStorageLocal（逻辑地址到物理地址映射文件）中。针对每个ORecordId数据仅容许一个版本。

***Create***时可以指定数据的版本号，存储时会使用指定的版本号。

***Update***时会检查version，如果version不匹配将throw OConcurrentModificationException

***Delete***时会检查version，如果version不匹配将throw OConcurrentModificationException

版本接口为ORecordVersion，其包含OSimpleVersion(counter) 和 ODistributedVersion( 用于分布式包含信息为mac地址和timestamp及counter)

### DataSegment

ODataLocal,用来保存record持久化数据，每条记录以如下格式进行持久化：

|  |  |  |
| --- | --- | --- |
| 名称 | 大小 | 说明 |
| contentSize | 4bytes | 记录持久化长度 |
| clusterId | 2bytes |  |
| clusterPos | 8bytes |  |
| content | <record size> bytes |  |

ODataLocal基于OMultiFileSegment实现。

在记录保存时需要指定记录的dataSegmentId，

ODatabaseRecordAbstract.executeSaveRecord中方法

int dataSegmentId = dataSegmentStrategy.assignDataSegmentId(this, record);

可以获取记录的segmentId，默认为0

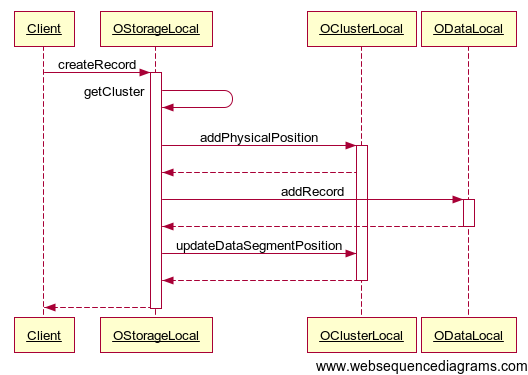
#### ？Hole管理

对于数据段文件由于增加、修改、删除记录导致，记录与记录之间会存在hole，ODataLocal有自己的hole管理器。

#### createRecord

需要参数：记录所在segment、记录所在cluster、记录content、记录id

1. 构造记录的物理位置 physical position
2. 产生记录的cluster position
3. 将记录的physical position加入到cluster中
4. 在data segment中添加记录，同时更新 physical position
5. 更新cluster中此记录的physical position



#### UpdateRecord

#### deleteRecord

需要参数：记录所在cluster、记录id、

1. 在cluster中找出数据记录的physical position记录
2. 在记录的segment中删除记录
3. 将physical position 中version 置为-1，更新holeSegment

### 事务OStorageLocalTxExecuter

OStorageLocal使用OStorageLocalTxExecuter类进行事务事务管理。

OStorageLocalTxExecuter提供操作

createRecord:调用OstorageLocalTxExcecutor的createRecord,同时记录日志，日志记录在OTxSegment中

***事务存储段*** 当对record进行操作时，会用变更前的数据和事务id构造一个TxRecord保存在txSegment中。

***回滚 rollback*** Undo,使用TxSegment中的TxEntry

### OlocalPaginatedStorage

页面存储,DataSegmentId 将被忽略，没有数据段概念，数据存储于Cluster中。

#### 页面大小

页面缺省大小为64kb

#### CreateRecord

直接调用cluster的createRecord。

### OPaginatedCluster

***职责***：负责记录的存储。

***产生：***

#### PositionMap

OClusterPositionMap 用于将记录的物理地址转换为逻辑地址。在分页存储模式下，记录的物理地址由(pageIndex,pagePosition)组成。

对照数据存储在.cpm文件中。数据存储到此文件时也采用page技术

##### Add

添加一个位置，首先查看当前OClusterPositionMapBucket 页面是否已满，如果full，则生成新的OClusterPositionMapBucket。

在OClusterPositionMapBucket中保存记录的物理地址(pageIndex, recordPosition) 同时返回逻辑地址position。

***寻址空间*** PageIndex为long， pagePosition为int，即entry在page中偏移量

其使用storage的diskCache进行持久化操作。

记录的逻辑地址：

#### 压缩

保存在OPaginatedCluster的数据将被压缩，默认使用snappy压缩，支持的压缩列表如下：

|  |  |
| --- | --- |
| snappy |  |
| gzip |  |
| snappy |  |
| snappy-native |  |
| zip |  |

#### Record

|  |  |  |
| --- | --- | --- |
| 名称 | 大小 | 值 |
| recordType | 1byte |  |
| content.length | 2bytes |  |
| content | n (content.length) bytes |  |
| …保留 | 0.2\*n byte |  |
| crc32 | 2bytes | 校验 |
|  | 1byte | 创建时值为1 |
|  | 4byte | 创建时值为-1 |
|  |  |  |
|  |  |  |

有时候record需要修改，修改后的record大小如果在 Record的保留字节，

### 原子操作

一个或多个操作组成一个整体，要么全部成功，要么全部失败，不可出现中间态。由OAtomicOperationsManager进行管理。

***startAtomicOperation*** 开始一个原子操作，并赋予当前线程上下文。原子操作由一个128bits的操作ID（uuid）,和OLogSequenceNumber（segment，position）组成。当前线程引用计数增1，WALlog登记操作开始。

***endAtomicOperation*** 结束原子操作，当前线程引用计数减1，当计数为0时释放操作资源引用到的lock，WALlog登记操作结束。当前线程上下文关联的操作对象清除。

### WALlog

OLocalPaginatedStorage使用其记录事务日志

***文件地址*** 位于storage所在目录，文件名称为\*.number.wal

***LSN*** *Log Sequence Number*:

### Version of data

# 代码分析

## 存储层

## OCluster

用来处理logical address 到 physical address 的转换

Record structure: 15 bytes

|  |  |  |
| --- | --- | --- |
| DATA SEGMENT | 2 bytes = max 2^15-1 |  |
| DATA OFFSET | 8 bytes = max 2^63-1 |  |
| RECORD TYPE | 1 byte | 记录类型 |
| VERSION | 4 bytes = max 2^31-1 | 记录的版本 |

每条记录15bytes

## OClusterLocal

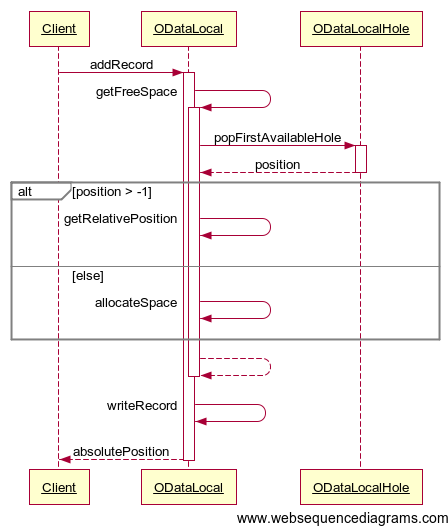
实现了OCluster接口，其中OMultiFileSegment为其底层存储。

## OSegment

### OMultiFileSegment

Implements OSegment接口，将多个文件抽象为Segment，提供面向文件级的存储功能。

### ODataLocal



## OsingleFileSegment

提供单文件操作能力

## OSBTree

仅用来存储索引

### 幕后存储

整棵树保存在一个文件中。

为了实现高效的磁盘访问，介入DiskCache，DiskCache针对文件的page进行缓存OCacheEntry。

整个树保存在一个文件中，树被分为多个Page，每个page关联一个OCacheEntry。

有三类page， 叶子Page， 节点Page， ValuePage，

### 如何放入数据

数据以(key,value)对放入

如果value为超大字段，使用ValuePage保存此value先。

1. Tree已经含有此key，找到key对应的OSBTreeBucket，找到key对应itemIndex，更新相应value。

Hole 问题？

由于新旧值大小不一样直接更新会造成hole或溢出，会先删除、再插入。

### 查找Bucket

根据key查找相应的bucket，一直找到leaf节点才返回

Bucket

Bucket

K1

K2

K3

K4

K1

K2

K3

K1

K2

K3

K4

Bucket

K1

K2

K3

K4

Bucket

K1

K2

K3

K4

在treebucket 中查找key对应位置 (如果存在返回key位置，不存在返回key应插入的位置)

### Split Bucket

Bucket拆分，

1. 选取entry, bucket.size()/2
2. 如果拆分的是叶子节点，

K1

K2

K3

KS

Bucket

K5

K6

K7

K8

### OSBTreeBucket

Extends ODurablePage，每个bucket为一页，页类型分为叶子和非叶子，叶子页内部的entry都为叶子节点，非叶子页的entry都为内部节点。

***数据结构***

继承ODurablePage内部数据格式

|  |  |  |
| --- | --- | --- |
| 43-46 | 4btes | SIZE\_OFFSET |
| 46-49 | 4bytes | IS\_LEAF\_OFFSET |
| 49-50 | 8bytes | LEFT\_SIBLING\_OFFSET |
| 51-57 | 8bytes | LEFT\_SIBLING\_OFFSET |
| 58-63 | 8bytes | TREE\_SIZE\_OFFSET |
| 64-71 | 8bytes | KEY\_SERIALIZER\_OFFSET |
| 72-73 | 1byte | VALUE\_SERIALIZER\_OFFSET |
| 74-75 | 1byte | FREE\_VALUES\_LIST\_OFFSET |
| 75-82 | 8bytes | POSITIONS\_ARRAY\_OFFSET |
|  |  |  |

其中Key\_serializer\_offset和value\_serializer\_offset为以前版本兼容而保留

全局字段

FREE\_POINTER\_OFFSET指向剩余空间的末端，用于判定bucket是否还有空间加入entry。

FREE\_VALUES\_LIST\_OFFSET 指向OSBTreeValuePage List的头节点。

***查找key***

根据折半查找法，返回key所在entry的index

***根据index获取entry***

***添加entry***

添加时需要指定添加的位置(index) ，entry的属性值（leftChild ，rightChild，key）

计算entry size

校验剩余空间

如果不是在末尾添加entry，需要移动entry position

将entry数据写入到bucket

Entry

Entry

Entry

Entry

Entry

FREE\_POINTER\_OFFSET

New Entry

***删除entry***

根据位置删除entry，只有叶子TreeBucket上的entry才能被删除。

内部bucket上的entry是否永久保存？

Entry

Entry

Entry

Entry

Entry

FREE\_POINTER\_OFFSET

当删除entry时，将出现hole，将position数组左移填补黑洞。Entry黑洞填补算法：从free\_pointer\_of指向的位置到要删除的entry都向右移一个entry size。根据entry移动情况，调整Position\_array\_offset中的指针位置

### SBTreeEntry

***Entry结构***

OSBTreeBucket由Entry组成。如果OSBTreeBucket为leaf，则其内的entry均为叶子节点，仅包含key、value。如果OSBTreeBucket非leaf，则其内的entry由left、right、key构成

Left

Right

Key

Key

Value

***POSITIONS\_ARRAY\_OFFSET***

每个entry 包含key 和value，全部序列化为字节流，entry的地址保存在positions\_array\_offset数组中。

POSITIONS\_ARRAY\_OFFSET

ENTRY\_INDEX

Entry

？position数组的必要性

在不进行反序列化的情况下即可进行key查找。

***针对超长value问题***

对于超长value，其value保存在？entry中的value仅保存真正value的地址

***数组大小问题***

***Size***

Size为key的数量

### ODurablePage

内部数据格式为

|  |  |  |
| --- | --- | --- |
| 0-7 | 8btes | magic Number |
| 8-11 | 4bytes | crc32 of page content |
| 12-19 | 8bytes | WAL\_SEGMENT\_OFFSET (LSN) |
| 20-27 | 8bytes | WAL\_POSITION\_OFFSET (LSN) |
| 28-35 | 8bytes | NEXT\_FREE\_POSITION |
|  |  |  |

***Change log***

Page 提供基本的字节码操作，根据TrackMode的不同会记录相应的change log。Change log可以用于事务回滚。ChangeLog保存为Page的pageChanges（内存），根据需要会被持久化到WALlog。

### OSBTreeValuePage

Extends ODurablePage

内部数据格式，TreeValuePage会形成一个链，用来保存大value值。

|  |  |  |
| --- | --- | --- |
| 36 | 8btes | FREE\_LIST\_NEXT\_PAGE\_OFFSET |
| 37-40 | 4bytes | WHOLE\_VALUE\_SIZE\_OFFSET |
| 41-44 | 4bytes | PAGE\_VALUE\_SIZE\_OFFSET |
|  | 8bytes | NEXT\_VALUE\_PAGE\_OFFSET |

ValuePage 会形成2个链, 未使用list 由rootPage 的FREE\_VALUES\_LIST\_OFFSET指向。已使用list 由上一个linke Value 的link指向。

Entry

Entry

Value.link

NEXT\_VALUE\_PAGE\_OFFSET

Rootbucket. FREE\_VALUES\_LIST\_OFFSET

FREE\_LIST\_NEXT\_PAGE\_OFFSET

## OClusterPage

用来保存数据记录，提供数据修改、追加、删除。

### AppendRecord

添加记录

数据存储格式为：

## ODiskCache

OrientDB 磁盘数据缓存接口，管理所有文件的缓存，基于DirectoryMemory进行内存操作，使用分页模型。

### OReadWriteDiskCache

Implements ODiskCache

对内存中的Page缓存进行管理，装载或移除Page缓存。

算法及结构

|  |  |
| --- | --- |
| maxSize | 最大页面数, Max(外部设置的最大字节数/pageSize, 256)  及其最小为256页 |
| K\_IN | maxSize/4 |
| K\_OUT | maxSize/2 |
| am | LRUList |
| a1out | LRUList |
| alin | LRUList |
|  |  |

***LRUList***

**Least Recently Used 最近最少使用链表，核心思想：最近使用的东西放在前面**

**通用算法**

原理 1. 核心思想：使用age 位，回收age最小的

原理2. 核心思想：最近使用的东西放在最前面，最容易读取。Head 和 tail 指针。每个元素有before和after指针。（LinkedHashMap使用）

每当访问一个元素被便将其移动到head之前。

每次都从tail的after移除数据

**OrientDb 采用原理2**

**其大致与LinkedHashMap相同，区别在于其使用tail表示最近被访问的元素。其使用hash算法进行快速查找使用head和tail进行迭代。**

**参加putToMRU** **(OCacheEntry cacheEntry)**

## OWOWCache

存储文件ID和名称的缓存基于文件(name\_id\_map.cm) 文件记录格式如下：

|  |  |
| --- | --- |
| 4bytes | nameSize |
| n bytes | n 有上面的值决定 |
| long | file id |
|  |  |

Why 其将文件名称映射成ID？

# 版本发布

0.9.2 @2010/04/20

# 通用数据库总体结构

Database 层

Storage 层

FileSystem

Global Data Area

G

Local Data Area

G

Local Data Area

G

Local Data Area

G

wal

G

Transaction

G

每个Transaction 都存在一个LocalDataArea 与其关联

Transaction commit后，LocalDataArea中被提交的数据会进入Global Data Area。同时记录wal

Global Data Area 中数据最终会被同步到 FileSystem, 同步成功的数据的wal会被清除

FileSystem以key 、value形式存储数据

## What is record?

|  |  |  |  |
| --- | --- | --- | --- |
| Record id 组成12 Bytes | 20Bytes | 20Bytes | 12Bytes |

12 Bytes cluster id

20 Bytes File id

20 Bytes Block id

12 Bytes Block offset

总共支持4K cluster ，每个cluster支持1M个文件，每个文件支持1M个Block，每个Block为4K bytes大小。

因此每个文件最大大小为 4G

## What is File

第一个block为reserved，为meta block。

文件用来保存Block，由header和content组成。Hear结构

used block

## What is block

数据存取的最小单位，

基于block的数据存取系统，数据仅支持block单位大小的读取和写入。

Block中前32 bytes为保留

metaBlock

unusedBlock

usedBlock

unusedBlock

usedBlock

### Block allocate

首先从unused blocks 链中获取block, 否则新开辟一个block。放入used Blocks链头部

### Block release

将block 放入unused blocks 链头部

## Block 缓存策略

record分为两部分header 和 content。

## What is segment?

Segment 由一个或多个文件组成。Segment的模式参见record模式。

## Data segment

保存数据value

## Key segment

保存数据key和物理偏移对应

## Segment Compress

段压缩，会将段中记录没有使用的空间挤出。压缩后record的模式为origin，记录以原始尺寸占用磁盘空间。一般在数据库闲置时进行此操作，需要支持online compress。

## Cluster

Key segment 和 data segment组成cluster

## Meta cluster

数据库元信息

## 图物理存储层

## Disk block Size

http://pclt.sites.yale.edu/blog/2010/03/10/disk-block-size

Disk 以512bytes 作为基础单位进行数据操作，当写入、读取操作小于512bytes时，操作系统将会补足后再进行操作。

This tiny size is inefficient, and the operating systems long ago abandoned it for (in almost all cases) a 4096 unit of storage called a "page" or "allocation unit". Physically, each page is simply a collection of 8 sectors on the disk.

SSD 单位块大小为512K

8ms (seek data) + 4ms(rotate

# 事务

一个或多个数据库操作组成一个事务，数据库支持多个事务并发。

事务几个特性ACID(Atomic、Consistency、Isolation、Durability)

|  |  |  |
| --- | --- | --- |
| 特性 | 描述 | 例子 |
| Atomic | 事务为一个整体要么全部成功，要么全部失败 | 1. db.put(k1,v1) 2. db.put(k2,v2)   如果第二步失败，第一步更新也会失败 |
| Consistency | 应用于数据库中对象 | 表的主外键，列的值校验 |
| Isolation | 数据库事务并发时的相互影响级别 |  |
| Durability | 数据库重启后，数据不能丢失 | 事务一旦提交，数据就不能丢失 |

Isolation分析

WAL （write aHead log）确保数据库Durability

事务的产生主要为了保证能数据库数据处于一个正常的业务状态（可以理解的状态）。

当多个事务同时运行时，事务之间产生了相互影响，出现了3种现象。

T1

T2

DB

|  |  |  |
| --- | --- | --- |
| 现象 | 描述 | 问题 |
| dirty read | T1 修改数据库，T2 可以读到T1没有commit的任何数据 | 事务之间完全相互影响， |
| Non repeatable read | 针对单个数据，重复读取时发现值不一致  T1 修改数据库，T2能看到T1 commit后数据，T3修改后，T1再次读取，发现T3 commit后值 | 看到的数据可以随时被其它事务修改提交  其会造成update lost。事务读取的数据可以被任意事务更新 |
| Phantom Reads | 针对range 数据，重复读取时看见 insert 或 delete |  |
|  |  |  |

3种现象，4个级别

Read uncommited

Rad committed

DIRTY READ

Non repeatable read

Phantom Reads

Repeatable read

Serializable

# 测试

|  |  |
| --- | --- |
| StorageTest | 存储测试 |
|  |  |
|  |  |

# Red black tree

定义

Insert node ( always insert red node)

Case 1 , tree is empty

Before

After

Case 2, Parent is black

Before

After

Case 3 Parent is red, uncle is black or null

Before

After

Delete node

Before

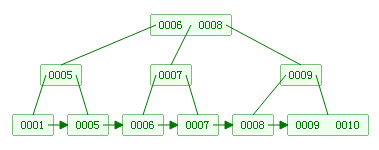
After

# B+ tree

演示地址

<https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html>

Max. Degree = 3 的B+tree



所有节点值均要在叶子节点出现。 叶子节点均链接起来。完整b+ tree 结构如下图。

008

006

Bucket

005

005

001

006

009

007

007

008

009

010