

The Java API quick reference

You can create both local (that is, without using a remote server) and remote database with the Java API. Each kind of database has a specific related class, but they expose the same interface:

- To create a local document database, use the `ODatabaseDocumentTx` class:

```
ODatabaseDocumentTx db = new ODatabaseDocumentTx  
("local:<path>/<db-name>").create();
```

- To create a local graph database, use the `OGraphDatabase` class:

```
OGraphDatabase db = new GraphDatabase("local:<path>/<db-name>").  
create();
```

- To create a local object database, use the `OObjectDatabaseTx` class:

```
OGraphDatabase db = new GraphDatabase("local:<path>/<db-name>").  
create();
```

- To create a remote database:

```
new OServerAdmin("remote:<db-host>").connect(<root-  
username>,<root-password>).createDatabase(<db-name>,<db-  
type>,<storage-type>).close();
```

- To drop a remote database:

```
new OServerAdmin("remote:<db-host>/<db-name>").connect(<root-  
username>,<root-password>).dropDatabase();
```

Where:

- `path`: This specifies the path where you wish to create the new database.
- `db-host`: This is the remote host. It could be an IP address or the name of the host.
- `root-user`: This is the root username as defined in the server config file.
- `root-password`: This is the root password as defined in the server config file.
- `db-name`: This is the name of the database.
- `db-type`: This specifies the type of database. It can be "document" or "graph".
- `storage-type`: This specifies the storage type. It can be `local` or `memory`. `local` means that the database will be persistent, while `memory` means that the database will be volatile.

Open and close connections

To open a connection, the open API is available:

```
<database> (<db>) .open (<username>, <password>);
```

Where:

- database: This is a database class. It can be:
 - ODatabaseDocumentTx for a document database
 - OGraphDatabase for a graph database
 - OObjectDatabaseTx for an object database (remember to register your model classes)

```
database.getEntityManager().registerEntityClass("<your-model-package>");
```

- db: This is the path to the database. Its syntax is `local:<path>/<db-name>` for local databases or `remote:<db-host>:<db-port>/<db-name>` for remote databases.
- username: This specifies the username to be used.
- password: This specifies the password.

In a multithreaded environment, you may use the connection pool. In this case, the connections are established only once and then assigned to the threads. Remember to close the connection when the thread no longer uses it. This makes the connection available to other threads:

- Document databases:

```
ODatabaseDocumentTx database = ODatabaseDocumentPool.global().  
acquire("<db>", "<username>", "<password>");
```
- Graph databases:

```
OGraphDatabase database = OGraphDatabasePool.global().  
acquire("<db>", "<username>", "<password>");
```
- Object databases:

```
OObjectDatabaseTx database = OObjectDatabasePool.global().  
acquire("<db>", "<username>", "<password>");
```

To close a connection, use the database's `close()` method.

Schema manipulation

To get access to the database schema, you must use the `getSchema()` API:

```
OSchema schema=database.getMetadata().getSchema();
```

Where `database` is an instance of an opened database.

Classes

To manage classes you have to use the schema object:

- To create a new class, use the following statement:

```
OClass newClass = schema.createClass("<class-name>");
```
- To get an already created class, use the following statement:

```
OClass aClass = schema.getClass("<class-name>");
```
- To drop a class, use the following statement:

```
database.getMetadata().getSchema().dropClass("<class-name>");
```

Where `class-name` is the name of the new class.



When a class is dropped, its records are not deleted. You must delete them before dropping the class. You may use the following command:

```
database.command( new OCommandSQL("TRUNCATE  
CLASS <class-name>") ).execute();
```

Class fields

To create, modify, or remove class fields (also known as properties), you have to act using the `OClass`'s methods:

- To create a property, use the following statement:

```
aClass.createProperty("<new-property>", OType.INTEGER);
```

The `OType` enumerator contains all the supported property types. For example, to create a property of type `link` for another class, use the following statement:

```
aClass.createProperty("<property-name>", OType.LINK, otherClass);
```

- To define constraints against properties, there are several methods exposed by the `OProperty` class. These methods are as follows:
 - `setNotNull()`
 - `setMandatory()`
 - `setReadOnly()`
 - `setMin()`
 - `setMax()`
 - `setRegexp()`
 - `createIndex()`

Example:

```
aClass.getProperty("<property-name>").setMandatory(true).setNotNull(true);
```

- To drop a property:

```
aClass.dropProperty("<property-name>");
```



When you drop a property from a class, the records belonging to that class do not lose their properties. You only change the class's schema definition. To remove a property from records, execute the following command:

```
database.command(new OCommandSQL("UPDATE <class-name> REMOVE <property-name>")).execute();
```

Indices management

Indices are managed by the `IndexManager` class. To obtain an instance of the manager, use the following statement:

```
OIndexManager idxManager = database.getMetadata().getIndexManager();
```

Here, `database` is an open instance of a database connection. The available APIs are as follows:

- `getIndex("<index-name>")`: This returns an instance of the `OIndex` class representing the requested index
- `existsIndex("<index-name>")`: This returns `true/false` according to the existence of the requested index
- `dropIndex("<index-name>")`: This drops the specified index

To create an index there are several options according to the type of index you need:

- To create an index against a class property, the most convenient way is to use the `createIndex` method of `OClass`:

```
aClass.createIndex("<index-name>", OClass.INDEX_TYPE.<index-type>,
"<property-names>");
```

Where:

- `index-type`: This is a value among the `OClass.INDEX_TYPE` enumerator
 - `property-name`: This is an already existing class property or a comma-separated list of properties' names for composite indices
- To create a dictionary in the database, you could use the `createIndex()` method of the `OIndexManager` class:

```
idxManager.createIndex(<dictionary-name>, OClass.INDEX_TYPE.
DICTIONARY.toString(), new OSimpleKeyIndexDefinition(OType.
STRING), null, null);
```

- Once you obtain an index instance, you can get and/or put a document in it:

```
OIdentifiable record = index.get("<key>");
if(record!= null ) ODocument doc=( ODocument) record.
getRecord();
```

Data management

Once you have opened a connection, you can manipulate data in the same thread without making any explicit reference to the connection itself.

Documents management

The class used to manipulate a document is the `ODocument` class.

- A document can be created as follows:

```
ODocument doc = new ODocument("<class-name>");
doc.field( "<property>", <value> );
doc.save();
```

Where `value` is the value to be assigned to `property`. It can be of any supported data type, even another `ODocument` instance (to embed documents).

- A document can be loaded as follows:

```
ODocument doc=database.load(<rid>);
```

Where `database` is the opened database connection and `rid` is a valid document's RID.

- To update a document, you have to load it, update it, and then save it:

```
ODocument doc = database.load(<rid>);  
doc.field("property-name", <new-value>);  
doc.save();
```

- You can merge two documents using the `merge()` method. It takes the properties of a given `ODocument` instance and merges them to the document as follows:

```
theDocument.merge(otherDocument, iAddOnlyMode,  
iMergeSingleItemsOfMultiValueFields);
```

Where:

- `theDocument`: This is an `ODocument` instance. It is the document to which the merge method will be applied.
 - `otherDocument`: This is another `ODocument` instance. It is the other document to be merged.
 - `iAddOnlyMode`: This is a Boolean value. If set to true, the method only adds or replaces the `otherDocument` fields. Otherwise, all the properties that are not present in `otherDocument` will be removed from the `theDocument` instance.
- To delete a document:

```
doc.delete();
```

Vertices and edges management

Vertices and edges are particular kinds of documents, so to manipulate them, you have to use the `ODocument` class. However, in the following statements, `database` is an instance of the `OGraphDatabase` class:

- A vertex can be created as follows:

```
ODocument node = database.createVertex();
```

- An edge between two vertices can be created as follows:

```
ODocument edge=database.createEdge(vertex1,vertex2);
```

- A vertex and an edge can be deleted as follows:

```
database.removeVertex(vertex);
database.removeEdge(edge);
```



To delete edges and vertices, you must use a specific API rather than the ODocument ones.

- All the outgoing or incoming edges of a vertex can be retrieved as follows:


```
database.getOutEdges(node);
database.getInEdges(node);
```
- All the outgoing or incoming vertices of an edge can be retrieved as follows:


```
database.getInVertex(edge);
database.getOutVertex(edge);
```

Objects management

If `db` is an instance of the `ObjectDatabaseTx` class, and `PojoClass` is your POJO class, you have:

- `db.getEntityManager().registerEntityClass(PojoClass.class)`: This registers `PojoClass` to the database engine.
- `PojoClass pojo = db.newInstance(PojoClass.class)`: This obtains a new instance of `PojoClass`.
- `db.save(pojo)`: This persists the `pojo` parameter. It creates the relative document if this is a new object or updates it if it already exists.
- `db.delete(pojo)`: This deletes the `pojo` parameter and the related document from the database.
- `db.attach(pojo)`: This attaches the `pojo` parameter and binds it to its related document representation in the database.
- `db.attachAndSave(pojo)`: This attaches the `pojo` parameter and persists it.
- `db.detach(pojo)`: This detaches the `pojo` parameter from its document representation. It returns a proxied object.
- `db.detach(pojo, true)`: This is similar to the previous one, but it returns a non-proxied instance.
- `db.detachAll(pojo)`: This detaches all the `pojo`'s tree recursively. It can cause an `StackOverflowException` exception on big graphs.

Queries

OrientDB can execute SQL queries via the Java API. So you can execute all the SQL queries within your Java code. There is support for native queries as well. However, although these queries may seem faster than the SQL ones due to the absence of the SQL parser overload, they are generally slower because they don't use indices and they are not optimized internally by the database engine. Furthermore, they offer fewer operators than the SQL language:

- To execute an SQL query:

```
List<ODocument> result = db.query(new
OSQLSynchQuery<ODocument>("<sql-statement>")).execute();
```

- To use a prepared statement you can use `?` for positional parameters or the `:var` notation for named parameters:

```
OSQLSynchQuery<ODocument> query = new
OSQLSynchQuery<ODocument>("<sql-statement>");
Map<String, Object> params = new HashMap<String, Object>();
params.put("<parameter>", "<value>");
List<ODocument> result = database.command(query).execute(params);
```

- To execute an SQL command (update or delete) rather than a query, you must use the `OCommandSQL` class instead of the `OSQLSynchQuery` class.
- To execute native queries, you must use the `OQueryContextNative` class:

```
List<ODocument> johnsPosts = database.query(new
ONativeSynchQuery<ODocument, OQueryContextNative<ODocument>>
(database, "Posts", new OQueryContextNativeSchema<ODocume
nt>()) {
    @Override
    public boolean filter(OQueryContextNative<ODocument>
iRecord) {
        return iRecord.field("author").field("name").eq("John").
go();
    };
});
```

- A List of records belonging to a class can be returned as follows:

```
database.browseClass("<class-name>");
```

- The number of records belonging to a class can be returned as follows:

```
database.countClass("<class-name>");
```




Don't forget that the field names are case sensitive, while class names are case insensitive. And also don't forget to invoke the `go()` method at the end of the operators chain.

Traverse

To traverse a graph or a document database, you can use the `OTraverse` class, which exposes several methods to build traverse commands. The complete documentation is available at the official OrientDB wiki <https://github.com/nuvolabase/orientdb/wiki/Java-Traverse>.

`OTraverse` exposes `Iterator` and implements the `Iterable` interface, so you can iterate through its result. This is very useful because an `Iterator` uses less system resources than loading an entire result set in a single shot.

Until now, we have seen an overview of the available Java API. You can find the complete reference at the official OrientDB wiki. Furthermore, since the `OQueryContextNative` class is not well documented, you can find its source code located at <https://github.com/nuvolabase/orientdb/blob/ef7d7c9d0a54a1d6bf9b9cac444152f753e39a2a/core/src/main/java/com/orientechnologies/orient/core/query/nativ/OQueryContextNative.java>, where you can see an entire set of available fluent APIs. In the next section, we will take a look at some of the most useful configuration parameters available to tune an OrientDB server.

Configuration parameters reference

OrientDB has several configuration parameters that control almost every aspect of the server behavior.

To dump the current server configuration

You can inspect the current configuration in different ways:

- Via the console tool using the `config` command
- At server startup via the JVM parameter:

```
java -Denvironment.dumpCfgAtStartup=true ...
```

- At runtime via the Java API:
`OGlobalConfiguration.dumpConfiguration(printStream);`

Where `printStream` is an instance of the `PrintStream` class, for example, `System.out`.

To set a configuration parameter

All parameters affect the current JVM. So, for example, modifying the cache size in a client JVM does not affect the cache size on the server JVM. The following are the options to set a parameter value:

- Via the console tool:
`config set <key> <value>`
- At server startup via the JVM parameter:
`java -D<key>=<value>`
- At server startup via the configuration file:

```
<properties>
  ...
  <entry name="<key>" value="<value>" />
  ...
</properties>
```
- At runtime via the Java API using the `OGlobalConfiguration` enumerator. Each configuration parameter has a corresponding entry in `OGlobalConfiguration`:
`OGlobalConfiguration.<parameter>.setValue(<value>);`

Configuration parameters

Here you will find the configuration parameters grouped by scope. Each table has four columns:

- **Parameter:** This is the parameter key to be used in the XML configuration files in the console and also as JVM startup parameters.
- **OGlobalConfiguration:** This is the enumerator of all the configuration properties. You can use it to alter parameter values using Java.
- **Type:** This is the type of the value accepted by a parameter.
- **Description:** This is a brief description about the parameter.

All the parameters are valid for both client and server, except for some specific ones.

Environment

These parameters allow you to inspect the current configuration and to set specific multithreaded options:

Parameter	OGlobalConfiguration	Type	Description
environment. dumpCfgAtStartup	ENVIRONMENT_DUMP_ CFG_AT_STARTUP	Boolean	This dumps the configuration settings at startup.
environment. concurrent	ENVIRONMENT_ CONCURRENT	Boolean	This specifies if OrientDB is running in a multithreaded environment. Set this to <code>false</code> to turn off the internal lock management.

Memory

Garbage collector configurations:

Parameter	OGlobalConfiguration	Type	Description
jvm. gc.delayForOptimize	JVM_GC_DELAY_FOR_ OPTIMIZE	Long	It specifies the minimum number of seconds since last <code>System.gc()</code> after tree optimization. Default is 600.

Storage

The following are very low level storage parameters, generally, you need not modify them:

Parameter	OGlobalConfiguration	Type	Description
storage.cluster. useNodeIdAsCluster Position	USE_NODE_ID_ CLUSTER_POSITION	Boolean	
storage.keepOpen	STORAGE_KEEP_OPEN	Boolean	If true, the storage will be kept open even when a database is closed. The storage will be closed at server shutdown. Default is <code>false</code> .

Parameter	OGlobalConfiguration	Type	Description
storage.lockTimeout	STORAGE_LOCK_TIMEOUT	Integer	This specifies the maximum timeout (in milliseconds) to lock the storage.
storage.record.lockTimeout	STORAGE_RECORD_LOCK_TIMEOUT	Integer	This specifies the maximum timeout (in milliseconds) to lock a single shared record.
storage.useTombstones	STORAGE_USE_TOMBSTONES	Boolean	If true, that is, when a record is deleted, its cluster space will not be reallocated.

Cache

The following are the cache parameters, and you can enable/disable them and modify some of their behaviors:

Parameter	OGlobalConfiguration	Type	Description
cache.level1.enabled	CACHE_LEVEL1_ENABLED	Boolean	This enables/disables the level 1 cache.
cache.level1.size	CACHE_LEVEL1_SIZE	Integer	This expresses the size of the level 1 cache in terms of the number of records to be kept in memory. -1 means no limit, but if the system will run out of heap space, the cache will be freed.
cache.level2.enabled	CACHE_LEVEL2_ENABLED	Boolean	This enables/disables the level 2 cache.
cache.level2.size	CACHE_LEVEL2_SIZE	Integer	This expresses the size of the level 2 cache in terms of the number of records to be kept in memory. -1 means no limit, but if the system will run out of heap space, the cache will be freed.

Parameter	OGlobalConfiguration	Type	Description
cache. level2. impl	CACHE_LEVEL2_IMPL	String	This is the canonical name of the class that implements the level 2 cache. By default, it is <code>com.orienttechnologies.orient.core.cache.ODefaultCache</code> . If you want to write your own cache class, it must implement the <code>com.orienttechnologies.orient.core.cache.OCache</code> interface.
cache. level2. strategy	CACHE_LEVEL2_STRATEGY	Integer	<p>This specifies the strategy to be adopted when a record is read from the cache. 0 = pop the record from the cache, 1 = copy the record from the cache. The default is 0.</p> <p>With the POP strategy, the record is removed from the cache and passed to the requesting thread. Once the thread's connection is closed, the record will be pushed again in the cache.</p> <p>With the COPY strategy, a new record will be created and passed to the requesting thread. The original record remains in the cache and it will also be available to the other threads.</p> <p>If <code>environment.concurrent</code> is set to <code>FALSE</code>, no COPY will be performed.</p>

Database

The following are database-specific settings:

Parameter	OGlobalConfiguration	Type	Description
object. saveOnlyDirty	OBJECT_SAVE_ONLY_DIRTY	Boolean	The object database saves only the object bound to dirty records. Default is <code>false</code> .

Parameter	OGlobalConfiguration	Type	Description
db.mvcc	DB_MVCC	Boolean	Multi-Version Concurrency Control (MVCC) is the check that the DB engine performs against the version field to prevent stale data updates even outside the transactions (in this case, an <code>OConcurrentModificationExceptions</code> exception will be thrown). If you disable this feature, no check will be performed and performance will also increase, but in a multithreaded environment, there is no guarantee that a thread does not update an older version of a record.
db.mvcc.throwfast	DB_MVCC_THROWFAST	Boolean	This allows us to speed up the throw of an <code>OConcurrentModificationExceptions</code> exception. In this case, no context is generated and it is useful when details are unnecessary. Default is <code>false</code> .
db.validation	DB_VALIDATION	Boolean	If set to <code>false</code> , this allows us to skip the constraints defined in the classes schema. Default is <code>true</code> .
db.use.distributedVersion	DB_USE_DISTRIBUTED_VERSION	String	In a distributed environment, use a "distributed convention" for the version field, that is not just a counter, but which contains a timestamp and the MAC address of the server.

Transactions

The following table shows how you can modify the transactions' behaviors:

Parameter	OGlobalConfiguration	Type	Description
<code>nonTX. recordUpdate. synch</code>	<code>NON_TX_RECORD_ UPDATE_SYNCH</code>	Boolean	If set to <code>true</code> , the DB executes a sync against the filesystem at every record operation. This slows down the records' updates, but ensures reliability on unreliable drives. Default is <code>false</code> .
<code>nonTX.clusters. sync.immediately</code>	<code>NON_TX_CLUSTERS_ SYNC_IMMEDIATELY</code>	String	This is a list of clusters to sync immediately after updates, separated by commas. The default is <code>manindex</code> , which is the default cluster for manual indices.
<code>tx.useLog</code>	<code>TX_USE_LOG</code>	Boolean	By default, transactions use a logfile to rollback the data in case of a crash. If it is set to <code>false</code> , it disables the use of the transaction logfile.
<code>tx.log.fileType</code>	<code>TX_LOG_TYPE</code>	String	This is the type of the transaction logfile (<code>mmap</code> or <code>classic</code> ; <code>mmap</code> stands for memory mapping, that is, the file is used with the memory mapping technique to speed up the read/write operation). Default is <code>classic</code> .
<code>tx.log.synch</code>	<code>TX_LOG_SYNCH</code>	Boolean	If set to <code>true</code> , the DB executes a sync against the filesystem at every log operation. This slows down the transactions, but ensures reliability on unreliable drives. Default is <code>false</code> .

Parameter	OGlobalConfiguration	Type	Description
<code>tx.commit.synch</code>	<code>TX_COMMIT_SYNCH</code>	Boolean	This synchronizes the storage immediately after a transaction commit. Default is <code>false</code> .

Indices

These are the parameters that are used to change the behaviors of both the manual and automatic indices:

Parameter	OGlobalConfiguration	Type	Description
<code>index.auto.rebuildAfterNotSoftClose</code>	<code>INDEX_AUTO_REBUILD_AFTER_NOTSOFTCLOSE</code>	Boolean	This automatically rebuilds the indices if they are not previously closed correctly in case of a crash.
<code>index.auto.lazyUpdates</code>	<code>INDEX_AUTO_LAZY_UPDATES</code>	Integer	This sets the maximum number of item updates for automatic indices to cache before they are flushed. -1 indicates that the flush is performed on a transaction commit or a DB close.
<code>index.manual.lazyUpdates</code>	<code>INDEX_MANUAL_LAZY_UPDATES</code>	Integer	This sets the maximum number of item updates for manual indices to cache before they are flushed. -1 indicates that the flush is performed on a transaction commit or a DB close.

The following properties are used internally by the engine for indices management. Indices are built using an MVRB-TREE, an OrientDB proprietary data structure. You can try to modify their values to increase performance, but in general, this is not necessary:

Parameter	OGlobalConfiguration	Type	Description
<code>mvrmtree.timeout</code>	<code>MVRBTREE_TIMEOUT</code>	Integer	This specifies the timeout (in milliseconds) to acquire a lock against an index.

Parameter	OGlobalConfiguration	Type	Description
mvrbtree. nodePageSize	MVRBTREE_NODE_ PAGE_SIZE	Integer	This specifies the page size (in entries) of each node of the MVRB-TREE.
mvrbtree. loadFactor	MVRBTREE_LOAD_ FACTOR	Float	This is the HashMap load factor.
mvrbtree. optimizeThreshold	MVRBTREE_OPTIMIZE_ THRESHOLD	Integer	This performs an automatic optimization of the index structure after the specified number of operations. -1 means never.
mvrbtree. entryPoints	MVRBTREE_ ENTRYPOINTS	Integer	This specifies the number of entry points to start the searching entries.
mvrbtree. optimizeEntry PointsFactor	MVRBTREE_OPTIMIZE_ ENTRYPOINTS_FACTOR	Float	This is the multiplying factor to be applied to the entry points list to determine if an optimization is needed.
mvrbtree. entryKeys InMemory	MVRBTREE_ENTRY_ KEYS_IN_MEMORY	Boolean	If it is true, it keeps the unserialized keys in the memory.
mvrbtree. entryValues InMemory	MVRBTREE_ENTRY_ VALUES_IN_MEMORY	Boolean	If this is true, it keeps the unserialized values in the memory.
mvrbtree. ridBinary Threshold	MVRBTREE_RID_ BINARY_THRESHOLD	Integer	This defines the threshold for sets of RIDs as a number of entries to use the binary streaming instead of classic string streaming. -1 never uses binary streaming.
mvrbtree. ridNodePageSize	MVRBTREE_RID_NODE_ PAGE_SIZE	Integer	For sets of RIDs, page size (in entries) of each node of the MVRB-TREE.
hashIndex. bufferSize	HASH_INDEX_BUFFER_ SIZE	Integer	This specifies the size of a page buffer in MB.
mvrbtree. ridNodeSaveMemory	MVRBTREE_RID_NODE_ SAVE_MEMORY	Boolean	If it is true, it does not keep the RIDs in memory.

Collection

The following is the only setting available to alter the behavior of how OrientDB acts on a collection (generally, you do not have to touch it):

Parameter	OGlobalConfiguration	Type	Description
lazysset. workOnStream	LAZYSSET_WORK_ON_ STREAM	Boolean	If this is <code>true</code> , it works directly on the streamed buffer to reduce memory footprint and improve performance.

I/O

The following are filesystem-specific settings, from file lock behaviour to filesystem access strategies:

Parameter	OGlobalConfiguration	Type	Description
file.lock	FILE_LOCK	Boolean	This locks the files when they are in use.
file.defrag. strategy	FILE_DEFRAG_ STRATEGY	Integer	This is the strategy to adopt for file defragmentation. 0: sync defrag, 1: async defrag.
file.defrag. holeMax Distance	FILE_DEFRAG_HOLE_ MAX_DISTANCE	Integer	This specifies the maximum distance in bytes between holes in datafiles that trigger a defrag. -1 means dynamic size.
file.mmap. useOld Manager	FILE_MMAP_USE_OLD_ MANAGER	Boolean	There are two MMAP file managers. The old one is deprecated but still available.
file.mmap. autoFlush. timer	FILE_MMAP_ AUTOFLUSH_TIMER	Integer	This auto flushes the memory mapped blocks after every specified second. It is available only to the new map manager.
file.mmap. autoFlush. unusedTime	FILE_MMAP_ AUTOFLUSH_UNUSED_ TIME	Integer	This removes the unused memory mapped blocks after the specified number of seconds. It is available only to the new map manager.
file.mmap. lockMemory	FILE_MMAP_LOCK_ MEMORY	Boolean	This specifies if the allocated memory can be swapped or not. It is available only to the new map manager.

Parameter	OGlobalConfiguration	Type	Description
file.mmap.strategy	FILE_MMAP_STRATEGY	Integer	<p>This is only available with the deprecated MMAP manager. The new one specifies the value; 4 to disable MMAP, any other value enables it:</p> <p>0: Always use MMAP</p> <p>1: Use MMAP on writes and reads, only if the block pool is free</p> <p>2: Use MMAP on writes and reads, only when the block pool is already available</p> <p>3: Use MMAP on writes and reads, only if the block pool is already available</p> <p>4: Never use MMAP</p>
file.mmap.blockSize	FILE_MMAP_BLOCK_SIZE	Integer	<p>This specifies the size of the memory mapped block in bytes. It is available only to the old map manager.</p>
file.mmap.bufferSize	FILE_MMAP_BUFFER_SIZE	Integer	<p>This specifies the size of the buffer for direct access to the file through the channel in bytes.</p>
file.mmap.maxMemory	FILE_MMAP_MAX_MEMORY	Long	<p>Max memory allocable by the memory mapping manager in bytes. The limit will vary depending on the system. It is available only to the old map manager.</p>
file.mmap.overlapStrategy	FILE_MMAP_OVERLAP_STRATEGY	Integer	<p>This specifies the strategy to use when a request overlaps the in-memory buffers: 0: Use the channel access, 1: Force the in-memory buffer and use the channel access, 2: Always create an overlapped in-memory buffer. It is available only to the old map manager.</p>

Parameter	OGlobalConfiguration	Type	Description
file.mmap.forceDelay	FILE_MMAP_FORCE_DELAY	Integer	It specifies the delay time in milliseconds to wait between forced flushes of the memory-mapped block to disk. This value is used also when the engine tries to delete the DB files in case of DROP DATABASE.
file.mmap.forceRetry	FILE_MMAP_FORCE_RETRY	Integer	It specifies the number of times the memory-mapped block will try to flush to disk. This value is used also when the engine tries to delete the DB files in case of DROP DATABASE.
jna.disable.system.library	JNA_DISABLE_USE_SYSTEM_LIBRARY	Boolean	It disables the system-provided JNA support and uses those shipped with OrientDB.
file.cluster.useLHPEPS	USE_LHPEPS_CLUSTER	Boolean	It indicates whether the cluster file should be saved as a simple persistent list or as a hash map.
file.cluster.useMemory LHCluster	USE_LHPEPS_MEMORY_CLUSTER	Boolean	It indicates whether the cluster file should be saved as a simple persistent list or as a hash map.

Network

The following are network and remote protocol-related settings:

Parameter	OGlobalConfiguration	Type	Description
network.socketBufferSize	NETWORK_SOCKET_BUFFER_SIZE	Integer	It is the buffer size of the TCP/IP socket.
network.socketTimeout	NETWORK_SOCKET_TIMEOUT	Integer	It specifies the timeout of the TCP/IP socket in milliseconds.
network.retry	NETWORK_SOCKET_RETRY	Integer	It specifies the number of attempts to obtain a connection against the server.
network.retryDelay	NETWORK_SOCKET_RETRY_DELAY	Integer	It specifies the number of milliseconds to wait between attempts to obtain a connection against a server.

Parameter	OGlobalConfiguration	Type	Description
network.binary.loadBalancing.enabled	NETWORK_BINARY_DNS_LOADBALANCING_ENABLED	Boolean	It tells the engine the DNS that resolves the OrientDB server address as a .txt record for DNS load balancing.
network.binary.loadBalancing.timeout	NETWORK_BINARY_DNS_LOADBALANCING_TIMEOUT	Integer	It specifies the timeout (in milliseconds) to wait for the answer from the DNS about the .txt record for load balancing.
network.binary.maxLength	NETWORK_BINARY_MAX_CONTENT_LENGTH	Integer	It tells the TCP/IP to chunk the size in bytes for binary requests.
network.binary.debug	NETWORK_BINARY_DEBUG	Boolean	If it is true, it prints all the data incoming on the binary channel.
network.http.maxLength	NETWORK_HTTP_MAX_CONTENT_LENGTH	Integer	It specifies the maximum content length in bytes for HTTP requests.
network.http.charset	NETWORK_HTTP_CONTENT_CHARSET	Integer	It represents an HTTP response charset; default is UTF-8.
network.http.sessionExpireTimeout	NETWORK_HTTP_SESSION_EXPIRE_TIMEOUT	Integer	It specifies the HTTP session timeout in seconds.

Profiler

The following are internal profiler and statistics settings that are useful for performance tuning:

Parameter	OGlobalConfiguration	Type	Description
profiler.enabled	PROFILER_ENABLED	Boolean	It enables/disables the internal profiler, statistics, and counters.
profiler.config	PROFILER_CONFIG	String	It consists of comma-separated values of profiler-specific configuration parameters. They are: <seconds-for-snapshot>,<archive-snapshot-size>,<summary-size>.

Parameter	OGlobalConfiguration	Type	Description
profiler. autoDump. interval	PROFILER_AUTODUMP_ INTERVAL	Integer	It specifies the time interval in seconds between the automatic dump of profiler metrics. 0 disables the auto dump.

Log

The following are file and console log level settings:

Parameter	OGlobalConfiguration	Type	Description
log. console. level	LOG_CONSOLE_LEVEL	String	It is the console logging level. Its values are the same as that of <code>java.util.logging.Level</code> .
log.file. level	LOG_FILE_LEVEL	String	It is the file logging level.

Client

The following settings are specific only for Java clients:

Parameter	OGlobalConfiguration	Type	Description
client.channel. minPool	CLIENT_CHANNEL_ MIN_POOL	Integer	It specifies the minimum pool size in number of connections.
client.channel. maxPool	CLIENT_CHANNEL_ MAX_POOL	Integer	It specifies the maximum pool size in number of connections.
client. connectionPool. waitTimeout	CLIENT_CONNECT_ POOL_WAIT_TIMEOUT	Integer	It specifies the time in milliseconds to wait to obtain a connection from the pool, otherwise an <code>OLockException</code> exception is raised.
client.channel. dbReleaseWait Timeout	CLIENT_DB_RELEASE_ WAIT_TIMEOUT	Integer	It specifies the delay in milliseconds between attempts to resend a command to a frozen DB.

Server

The following settings are specific for JVMs that run an OrientDB server:

Parameter	OGlobalConfiguration	Type	Description
<code>server.channel.cleanDelay</code>	<code>SERVER_CHANNEL_CLEAN_DELAY</code>	Integer	It specifies the time (delay) in milliseconds to check the pending closed connections.
<code>server.cache.staticFile</code>	<code>SERVER_CACHE_FILE_STATIC</code>	Boolean	It enables/disables the cache for static resources for the HTTP embedded server.
<code>server.log.dumpClientExceptionFullStackTrace</code>	<code>SERVER_LOG_DUMP_CLIENT_EXCEPTION_FULLSTACKTRACE</code>	Boolean	If <code>true</code> , it dumps the full stack trace of any exception that is sent to the client.
<code>server.log.dumpClientExceptionLevel</code>	<code>SERVER_LOG_DUMP_CLIENT_EXCEPTION_LEVEL</code>	String	It defines the level of dumps of the exceptions sent to the clients. The values are the same as that of <code>java.util.logging.Level</code> .

We have seen the dozens of configuration parameters available to modify almost every aspect of both OrientDB Java clients and servers. In the next chapter, there are some links to the most useful resources available on the Internet.

Bibliography and resources

In this section there are some links to useful resources available on the Internet. I suggest you always keep an eye always on the official forum hosted by Google Groups. Furthermore, if you don't find something on the official wiki on GitHub, try to perform a search on the old wiki on Google Code:

- **OrientDB official site:** This is the OrientDB home site. Here you can find the latest stable builds and links to other useful stuff (<http://www.orientdb.org/>).
- **OrientDB official documentation wiki:** The wiki and the source code are hosted on GitHub at <https://github.com/nuvolabase/orientdb/wiki>.

- **The OrientDB official code repository is found at:**
<https://github.com/nuvolabase/orientdb>
- **The deprecated wiki site:**
<http://code.google.com/p/orient/wiki/Main?tm=6>
- **OrientDB official discussion forum:** This is the official discussion forum. This is the place where OrientDB users and its development team meet each other. If you have some problem, try to search the forum and then ask a question. I'm sure you will find your answer here:
<https://groups.google.com/forum/?fromgroups#!forum/orient-database>
- **OrientDB tutorials:** Getting started and step-by-step How to's:
<https://github.com/nuvolabase/orientdb/wiki/Tutorials>
<http://devdocs.inightmare.org/introduction-to-orientdb-graph-edition/>
 - In Italian at <http://www.html.it/articoli/nosql-in-java-introduzione-ad-orientdb-1/>
- **OrientTechnologies:** This is the company behind the development of OrientDB. You can start from its website to explore the OrientDB world:
<http://www.orienttechnologies.com/>
- **NuvolaBase, the OrientDB in the cloud:** NuvolaBase offers an OrientDB cloud solution. You can use OrientDB without installing it, even for free at <http://www.nuolabase.com/site/>.
- **OrientDB in the cloud and OrientDB Data as a service provider:**
http://www.nuolabase.com/site/index_cloud.html
- **Luca Garulli's (CEO at NuvolaBase LDT) SlideShare account:** Luca Garulli is the CEO of NuvolaBase Ltd and the OrientDB team leader. He gives talks and presentations around the world on a regular basis, so I suggest you follow him on SlideShare, because there is always something new to learn about graph database design and OrientDB uses (<http://www.slideshare.net/lvca>).
- **Luca Garulli's blog:**
<http://zion-city.blogspot.it/>
- **Official OrientDB Twitter account:**
<https://twitter.com/nuvolabase>

- **Google+ page:**
<https://plus.google.com/u/0/108716918285358206552/posts>
- **Google+ community page:**
<https://plus.google.com/u/0/communities/101799593865844060425>
- **The TinkerPop stack website:** The TinkerPop project groups an entire set of specifications and frameworks to use against any graph database that supports its default standard (<http://www.tinkerpop.com/>).

OrientDB 1.5.0

Recently the OrientDB team released a new major stable version: Version 1.5.0.

This release introduces some new features and signals a big milestone in the history of this database. In the official download page, now you will find the 1.5.0 package; meanwhile the other releases, including the 1.3.0 version, are available at <https://code.google.com/p/orient/downloads/list?can=1&q=&colspec=Filename+Summary+Uploaded+Size>.

The graph database

The team intent is to provide a full compliance to the TinkerPop Blueprints, so it has been decided to deprecate the OrientDB native Java API and to promote the Blueprint interface since Version 1.4.0.

You can read the announcement at <http://nuvolabase.blogspot.it/2013/04/orientdb-new-graphdb-engine-in-beta.html>.

However, the support for the native Java API will be assured.

The first consequence of this decision is that the new Graph API are available only in the Graphed Edition, and the Standard Edition does not manage graph-related commands. So, if you plan to model your data domain as a graph, you must use the Graphed Edition. Anyway, if your client uses the deprecated native Java API, it will still work, although you need to upgrade the client libraries.

The `OGraphVertex` and the `OGraphEdge` classes were removed from the graph databases. Now only the aliases `V` and `E` are supported.

Another big difference is in the management of edges. In previous releases, each vertex had two collections: `in` and `out`.

The first one contains pointers to the incoming edges, and the second one contains the pointers to outgoing edges. Many times, edges do not have their own attributes or properties, so to create and manage a data structure adhoc is a waste of space and time. So, from now on, when only one edge is present instead of a collection of edges, and if that edge has no attributes, a link is created. When another edge is created, automatically OrientDB transforms that link in a collection. From your point of view, this behavior is completely transparent and you do not have to worry about it, however it is important that you know how OrientDB manages edges.

Another improvement has been introduced in the management of the edges: you can extend the `E` class to create a new edge class (similar to the one in the previous release), but now when you create an edge using the new class, OrientDB creates a new property called `_EDGE_CLASS_NAME` (or `out_EDGE_CLASS_NAME`). This means that a vertex can have different edge properties depending on their type.



To alter the database's behavior so that it is still compatible with Version 1.3.0, one of the way to manage graph objects is by entering the following commands:


```
alter database custom useLightweightEdges=false;
alter database custom useClassForEdgeLabel=false;
alter database custom useClassForVertexLabel=false;
alter database custom useVertexFieldsForEdgeLabels=
false;
```

Let's look at a simple example. Enter the following code in the console:

```
orientdb-graphed-1.5.0/bin>console
OrientDB console Version 1.5.0 (build @BUILD@) www.orienttechnologies.com
Enter help to display all the supported commands.
Installing extensions for the GREMLIN language Version 2.4.0-SNAPSHOT
orientdb>
```

First of all let's create a graph database and create some vertices as follows:

```
create database local:../databases/mytestgraph admin admin local
graph;
create vertex set name = 'john';
create vertex set name = 'mike';
```

 You can create a vertex (or a document) by providing its content in JSON format as follows:

```
create vertex content { "name" : "John", "surname"
: "Doe" }
```

Now, let's create an edge type as follows:

```
create class son extends e;
```

We can create a relationship as follows:

```
create edge son from (select from v where name = 'mike') to (select
from v where name = 'john')
```

The result is as follows:

```
orientdb> select from v
-----+-----+-----+-----+-----
#  |@RID|name|out_son|in_son
-----+-----+-----+-----+-----
0  |#9:1|mike|#9:2  |null
1  |#9:2|john|null  |#9:1
-----+-----+-----+-----+-----
```

As you can see, the vertex `mike` has a link named `out_son` pointing to the `john` vertex, and this one has an `in_son` link with a reference to the first one.

Now let's create the mother vertex for mike:

```
create vertex set name = 'mary';
```

And we link them as follows:

```
create edge son from (select from v where name = 'mike') to (select
from v where name = 'mary');
```

The change in the vertices is as follows:

```
orientdb> select from v
-----+-----+-----+-----
#   |@RID|name|in_son|out_son
-----+-----+-----+-----
0   |#9:0|mary|#9:1  |null
1   |#9:1|mike|null   |[2]
2   |#9:2|john|#9:1  |null
-----+-----+-----+-----
```

Note how the `out_son` property of the `mike` vertex has been changed: from a link, it became a collection. This is managed completely by OrientDB.


New plocal storage engine

A new storage engine has also recently been introduced: the **Paginated Local (plocal)** engine.

It has been completely rewritten from scratch and it now provides a better support for concurrency and is more durable (it does not use the MMAP). The file structure is completely different from the local engine and takes up far less space on the storage device. To create a database using the new engine, from the console, enter the following command:

```
create database remote:[server][:port]/<database_name> root <ROOT_
PASSWORD> plocal graph
```

Remember that you can find your root password within the `{ORIENTDB_HOME}/config/orientdb-server-config.xml` file.

 To create a local database not using a remote database, from within the console, the syntax is as follows:

```
create database plocal:[path]/<database_name>
<user> <password> plocal graph
```

If you want to embed the OrientDB core in your own Java application, and you want to use the new plocal engine, you must include in your classpath the `snappy-0.3.jar` file located in the `{ORIENTDB_HOME}/config/lib` directory in addition to the OrientDB JAR files.

The internal scheduler

OrientDB now comes with an internal scheduler. This means that you can write your custom JavaScript functions and then schedule them to be executed when you decide. Version 1.5.0 does not work on plocal databases. For example, let's assume that we want to store the number of registered users in a class every minute. First of all, we have to create a class as follows to store the values:

```
create class users;
```

Next, let's create a function to store all the users, date, and time. Go to the OrientDB Web Studio and create the JavaScript function, `insertUsers` with the following code:

```
var qryResult = db.query("select count(*) as count from ouser");
var count = db.getRecord(qryResult[0]).field("count");
var now=new Date();
var command="insert into Users set count=" + count + " , date=" + now.getTime();
db.command( command );
return;
```

This function takes the current date and time and the number of users and then stores this information into the `users` class. Now we can set the scheduler. In the console, enter the following code:

```
insert into oschedule set function=(select from ofunction where
name='insertUsers'), rule='*/1 * * * *', name = "insertUsers",
start=true
```

Lets check our work:

```
orientdb> select from oschedule

-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
#  |@RID|function|rule          |name          |start|status
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
0  |#7:0|#6:0    |*/1 * * * *|insertUsers|true |WAITTING
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

1 item(s) found. Query executed in 0.0050 sec(s).
```

The special internal class, `OSchedule` contains the references to the scheduled tasks. It has the following properties:

- `function`: This is a link to a function stored in the `OFunction` class
- `rule`: This is a cron-like rule
- `name`: This is the name of the task
- `status`: This is the status of the scheduled task, its values can be: `STOPPED`, `RUNNING`, or `WAITTING` (currently this is a typo in the OrientDB 1.5.0 code, it should be `WAITING`)
- `start`: This is a Boolean value indicating if the task is enabled or disabled

Now we must activate the feature. Open the `{ $ORIENTDB_HOME }/config/orientdb-server-config.xml` file and into the `handlers` section, enter the following code:

```
<handler class="com.orienttechnologies.orient.server.schedule.
OScheduleHandler">
  <parameters>
    <parameter name="databaseName" value="YOUR_DATABASE_NAME_GOES_
HERE"/>
    <parameter name="user" value="admin"/>
    <parameter name="pass" value="admin"/>
    <parameter name="enabled" value="true"/>
  </parameters>
</handler>
```

Replace the `YOUR_DATABASE_NAME_GOES_HERE` string with the name of your database, put the right value into the `user` and `pass` parameters, and start the server.

When OrientDB restarts, it prints something as follows:

```
2013-08-31 23:18:01:562 INFO OrientDB Server v1.5.0 (build @BUILD@) is
starting up... [OServer]
2013-08-31 23:18:01:741 INFO Listening binary connections on
0.0.0.0:2424 (protocol v.17) [OServerNetworkListener]
2013-08-31 23:18:01:746 INFO Listening http connections on
0.0.0.0:2480 (protocol v.10) [OServerNetworkListener]
2013-08-31 23:18:01:779 INFO Installing GREMLIN language v.2.4.0-
SNAPSHOT - graph.pool.max=50 [OGraphServerHandler]
2013-08-31 23:18:01:841 INFO OrientDB Server v1.5.0 is active.
[OServer]
2013-08-31 23:18:01:843 WARN Schedule Timer Started [TimerThread]
```

The last line states that the scheduler is up and running. After about a minute, you should see that our task is completed. OrientDB logs the operation showing the following two lines:

```
2013-08-31 23:46:00:047 WARN execute : OSchedule
<name:insertUsers,rule:*/1 * * * *,current status:WAITTING,func:insert
Users,start:true> at 2013-08-31
  23:46:00:047 [OScheduler]
2013-08-31 23:46:00:122 WARN Job : OSchedule
<name:insertUsers,rule:*/1 * * * *,current status:RUNNING,func:insertU
sers,start:true> Finished! [OSchedu
ler]
```

You can execute a select statement in the `Users` class to see how many users are logged.

The UML class diagram

The OrientDB Web Studio now has a UML class diagram viewer. It allows you to see the database schema from a class diagram point of view. Assuming that you are running the 1.5.0 Graphed Edition, you should be able to log in to the TinkerPop database. In the main page, there is the link for a UML class diagram. Go to the link. If you are connected to the Internet, you should see the class diagram of the TinkerPop database.

Migrating from 1.3.0 databases

If you have to migrate from OrientDB 1.3.0 to the new 1.5.0 release, the best way to do so is to perform an export from Version 1.3.0, and then an import into the new server. OrientDB manages all the necessary stuff to perform the conversion.

In the 1.3.0 console, enter the following code:

```
OrientDB console v.1.3.0 - www.orienttechnologies.com
Type 'help' to display all the commands supported.

orientdb> connect local:databases/mydb admin admin
Connecting to database [local:./databases/mydb] with user 'admin'...
OK

orientdb> export database /mydb_export.json.gz
Exporting current database to: database /mydb_export.json.gz...
.
.
.
Database export completed in 534ms
```

Now from the 1.5.0 console, enter the following code:

```
OrientDB console v.1.5.0 (build @BUILD@) www.orienttechnologies.com
Enter help to display all the supported commands.
Installing extensions for the GREMLIN language v.2.4.0-SNAPSHOT
orientdb> create database local:databases/newmydb admin admin local
Creating database [local:databases/newmydb] using the storage type
[local]...
Database created successfully.

Current database is: local:databases/newmydb
orientdb> import database /mydb-export.json.gz
.
.
.
Database import completed in 862 ms
orientdb>
```

The complete migration guide can be found at the OrientDB official wiki page located at <https://github.com/orientechnologies/orientdb/wiki/Migration-from-1.3.x-to-1.4.x>.

It is about Version 1.4.0, but it is also applicable to Version 1.5.0.

Summary

In this chapter, we have seen the main differences between Versions 1.3.0 and 1.5.0. We have seen how the graph database engine has been optimized to better manage the edges, to be more durable in case of a system crash, and to occupy less space on storage devices. We have also seen some new features such as the internal scheduler and the UML class diagram viewer. Finally, we have seen how to migrate a database between two versions.