

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



KIẾN TẬP CÔNG NGHIỆP
TÌM HIỂU VỀ ORIENTDB

NGƯỜI HƯỚNG DẪN:
NGUYỄN MINH TUẤN

SINH VIÊN THỰC HIỆN:
BÀNH ĐẠI NAM - 51503279
HUỲNH THỊ LIỄU - 51503078

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2018

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



KIẾN TẬP CÔNG NGHIỆP
TÌM HIỂU VỀ ORIENTDB

NGƯỜI HƯỚNG DẪN:
NGUYỄN MINH TUẤN

SINH VIÊN THỰC HIỆN:
BÀNH ĐẠI NAM - 51503279
HUỲNH THỊ LIỄU - 51503078

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2018

LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập ở giảng đường đại học đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý thầy cô, gia đình và bạn bè. Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý thầy cô ở Khoa Công nghệ thông tin – Trường Đại Học Tôn Đức Thắng đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Và đặc biệt, trong môn *Kiến tập công nghiệp*, thầy đã tổ chức cho chúng em được tiếp cận với môn học mà theo em là rất hữu ích đối với sinh viên ngành **Công nghệ phần mềm** cũng như tất cả các sinh viên thuộc khoa **Công nghệ thông tin**. Em xin chân thành cảm ơn thầy *Nguyễn Minh Tuấn* đã tận tâm hướng dẫn chúng em qua từng buổi học trên lớp cũng như những buổi nói chuyện, thảo luận về lĩnh vực sáng tạo trong nghiên cứu khoa học. Nếu không có những lời hướng dẫn, dạy bảo của thầy thì em nghĩ bài thu hoạch này của em rất khó có thể hoàn thiện được. Một lần nữa, em xin chân thành cảm ơn thầy. Bài thu hoạch được thực hiện trong khoảng thời gian gần 10 tuần. Bước đầu đi vào thực tế, tìm hiểu về lĩnh vực sáng tạo trong nghiên cứu khoa học, kiến thức của em còn hạn chế và còn nhiều ngỡ ngàng. Do vậy, không tránh khỏi những thiếu sót là điều chắc chắn, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô và các bạn học cùng lớp để kiến thức của em trong lĩnh vực này được hoàn thiện hơn.

Lời cảm tạ thầy *Nguyễn Minh Tuấn*. Sau cùng, em xin kính chúc quý thầy cô trong khoa **Công nghệ thông tin** thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm dự án công nghệ thông tin 1 (xem mẫu trên sakai) của riêng chúng tôi và được sự hướng dẫn của thầy/cô. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày....tháng....năm 2018

Sinh viên

(Ký và ghi rõ họ tên):

GIỚI THIỆU

PHẦN NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của người hướng dẫn

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Tp.Hồ Chí Minh, ngày tháng.... năm.....
(Ký và ghi rõ họ tên)

Phần đánh giá của GV chấm bài

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Tp.Hồ Chí Minh, ngày tháng.... năm.....
(Ký và ghi rõ họ tên)

Mục lục

LỜI CẢM ƠN	i
ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG	ii
GIỚI THIỆU	iii
PHẦN NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN	iv
Mục lục	v
Danh sách bảng	vii
Danh sách hình vẽ	viii
1 GIỚI THIỆU	1
1.1 Giới thiệu OrientDB – phiên bản 2.2.x	2
1.1.1 Sơ lược tổng quát	2
1.1.2 Mở đầu	2
1.1.3 Mô hình NoSQL	3
1.1.4 Cài đặt OrientDB version 2.2.x,Run the Server,Run the Console,Run the studio	5
2 MÔ HÌNH DỮ LIỆU	19
2.0.1 Các khái niệm cơ bản	20
2.0.2 Đồ thị nhất quán	21
3 API và Driver	23
3.1 Hàm	23
3.1.1 Tạo hàm trong OrientDB	24
3.1.2 Sử dụng hàm trong OrientDB	25
3.1.3 Truy cập database từ function	29
3.2 Java API	30

3.2.1	Kiến trúc Component	30
3.2.2	Graph API	31
3.2.3	Cạnh và đỉnh trong đồ thị	34
3.2.4	Graph Schema	37
3.2.5	Document API	42
3.2.6	Object API	46
3.2.7	Traverse	51
3.2.8	Chiến lược duyệt đồ thị	52
3.2.9	Live Query	54
3.2.10	Sự lan truyền transaction	58
3.3	JDBC driver	60
3.3.1	JDBC là gì	60
3.3.2	Thêm JDBC driver vào project	62
3.3.3	Hiện thực JDBC driver và code như thế nào	62
4	SQL	65
4.0.1	Tự động sử dụng chỉ mục	65
4.0.2	Lệnh JOIN	66
4.0.3	Thực hiện truy vấn từ nhiều đối tượng	66
4.0.4	Truy vấn lướt đồ	66
4.0.5	So khớp trong SQL	68
	Tài liệu tham khảo	70

Danh sách bảng

BẢNG	Trang
1.1 Các chủ đề trong chính trong OrientDB	2
3.1 Thuộc tính hàm	24
3.2 Các Java API	31
3.3 Phương thức chung của cạnh và đỉnh	36
3.4 Ràng buộc trong thuộc tính	40

Danh sách hình vẽ

HÌNH	Trang
1.1 OrientDB	1
1.2 set thuộc tính ORIENTDB_ROOT_PASSWORD	5
1.3 Download orientdb trên docker	6
1.4 Chạy orientDb trên docker	8
1.5 Chạy orientDb trên docker bằng console	9
1.6 Kết nối với DB	9
1.7 Xem danh sách các database đang có	10
1.8 udio	10
1.9 Truy cập Studio bằng lệnh	11
1.10 Tất cả danh sách class trên hệ thống của bạn	12
1.11 Tạo lớp	13
1.12 Thêm một thuộc tính vào class	14
1.13 Hiển thị thông tin của class.	15
1.14 Lệnh ALTER PROPERTY	16
1.15 Trường(record)	17
2.1 Định dạng JSON	20
2.2 Tập cấu hình OrientDB	22
3.1 Lấy ra danh sách API	27
3.2 Kiến trúc Component	30
3.3 Tìm kiếm 10 bản ghi book	52
3.4 Tìm kiếm theo chiều sâu	53
3.5 Tìm kiếm theo chiều rộng	53
3.6 Mô hình truy vấn truyền thống	55
3.7 Mô hình Live Query	55
3.8 LIVE select from book	57
3.9 JDBC API	60
3.10 Kết nối database	61
3.11 DriverManager	62

4.1	Lấy ra tất cả các lớp	67
4.2	Lấy ra tất cả các cấu hình của lớp OUser	67
4.3	Lấy ra tất cả các cấu hình của chỉ mục	67
4.4	select from person	69
4.5	Đồ thị diễn tả quan hệ giữa các Person	69

Chương 1

GIỚI THIỆU

OrientDB là một hệ thống quản lý CSDL NoSQL mã nguồn mở được viết bằng Java. Nó là một CSDL đa mô hình hỗ trợ graph, document, key/value và object, nhưng các mối quan hệ được quản lý như trong CSDL đồ thị với kết nối trực tiếp giữa các bảng ghi. Nó hỗ trợ các kiểu schema-less, schema-full và schema-mixed. Nó có một hệ thống gia công bảo mật mạnh mẽ dựa trên người dùng, vai trò và hỗ trợ truy vấn với Gremlin cùng với SQL mở rộng cho việc truyền tải đồ thị.



HÌNH 1.1. OrientDB

(a) <https://orientdb.com/docs/>

OrientDB sử dụng một số cơ chế lập chỉ mục dựa trên B-tree và hàm băm mở rộng, phần cuối cùng được gọi là “chỉ số băm”, có kế hoạch cho việc thực hiện các chỉ mục dựa trên LSM-tree và Fractal tree index. Mỗi bảng ghi có Surrogate key (khóa thay thế) cho biết vị trí của bảng ghi bên trong danh sách Mảng, liên kết giữa các bảng ghi được lưu trữ dưới dạng giá trị duy nhất vị trí của bảng ghi được lưu trữ bên trong liên kết referrer hoặc B-tree của các vị trí bảng ghi (được gọi là ID bảng ghi hoặc RID) truyền tải nhanh chóng (với $O(1)$ độ phức tạp) của quan hệ one-to-many và nhanh chóng bổ sung/loại bỏ các liên kết mới. OrientDB là CSDL đồ thị phổ biến thứ ba theo xếp hạng CSDL đồ thị DB-Engines tính đến tháng 9 năm 2017.

Sự phát triển của OrientDB vẫn dựa vào một cộng đồng mã nguồn mở do công ty OrientDB

LTD tạo ra bởi tác giả ban đầu là Luca Garulli. Dự án sử dụng GitHub để quản lý mã nguồn, cộng tác viên và phiên bản, Google Group và Stack Overflow để cung cấp hỗ trợ miễn phí cho người dùng trên toàn thế giới. Orient cũng cung cấp một khóa học miễn phí Udemey cho những người có hi vọng tìm hiểu những điều cơ bản về OrientDB.

1.1 Giới thiệu OrientDB – phiên bản 2.2.x

1.1.1 Sơ lược tổng quát

Bảng 1.1: Các chủ đề trong chính trong OrientDB

Mở đầu	Chủ đề chính	Lập trình
Giới thiệu OrientDB	Các khái niệm cơ bản	SQL
Cài đặt OrientDB	Các kiểu dữ liệu được hỗ trợ	Gremlin
Những bước đầu tiên	Sự kế thừa	HTTP API
Khắc phục sự cố	Bảo mật	Java API và Drivers

1.1.2 Mở đầu

Chúng ta đã quá quen với SQL Server, MySQL, Oracle... khi làm việc với database điểm chung của chúng là đều sử dụng SQL là ngôn ngữ truy vấn, những năm gần đây có 1 dạng database khác với những đặc tính khác biệt được gọi chung dưới cái tên là NoSQL. Ý nghĩa của NoSQL không phải là không sử dụng SQL làm ngôn ngữ truy vấn nữa mà là Not only SQL – Không chỉ SQL nó ám chỉ đến những CSDL không dùng mô hình dữ liệu quan hệ để quản lý dữ liệu nhằm mở rộng tầm nhìn cho các lập trình viên cũng như các nhà kinh doanh để họ tiếp cận mô hình mới hơn.

Các giải pháp thay thế cho các hệ thống quản lý CSDL quan hệ đã tồn tại trong nhiều năm, nhưng chúng đã bị xuống hạng thấp và chủ yếu được sử dụng trong các trường hợp thích hợp như viễn thông, y học, CAD... và các trường hợp khác. Sự quan tâm đến các lựa chọn thay thế NoSQL như OrientDB đang gia tăng đáng kể. Không ngạc nhiên, nhiều công ty web đứng đầu như Google, Facebook, Amazon, Foursquare, Twitter đang sử dụng các giải pháp dựa trên NoSQL trong môi trường sản xuất của họ.

Điều gì thúc đẩy các công để mang lại sự thoải mái của một thế giới CSDL quan hệ được thiết lập tốt? Đó là nhu cầu cơ bản lớn để giải quyết vấn đề dữ liệu ngày nay tốt hơn. Cụ thể có vài lĩnh vực chính:

- Hiệu suất
- Khả năng mở rộng (thường rất lớn)

- Vùng chứa nhỏ
- Năng suất cao và thân thiện với lập trình viên
- Độ linh hoạt của lược đồ

Hầu hết các vùng nhớ này cũng là các yêu cầu của các ứng dụng web hiện đại. Một vài năm trước, các lập trình viên đã thiết kế các hệ thống có thể xử lý hàng trăm người dùng đồng thời. Ngày nay, không có gì khác thường khi có một mục tiêu tiềm năng có thể kết nối hàng nghìn hoặc hàng triệu người dùng và phục vụ cùng một lúc.

Thay đổi những yêu cầu công nghệ đã được xem xét đưa vào tài khoản trên app front (ứng dụng mà người dùng tương tác trực tiếp) bằng cách tạo ra frameworks, giới thiệu các tiêu chuẩn và tận dụng các phương pháp hay nhất. Tuy nhiên, trong thế giới CSDL, trạng thái vẫn tồn tại lớn hoặc nhỏ hơn 30 năm qua. Từ những năm 1970 cho đến gần đây, các DBMS quan hệ đã đóng vai trò chủ đạo. Những ngôn ngữ lập trình và những phương pháp luận đã phát triển, nhưng khái niệm ổn định dữ liệu và DBMS vẫn không đổi đối với hầu hết các phần: đó là tất cả các bảng tĩnh, bản ghi và các liên kết.

1.1.3 Mô hình NoSQL

Các giải pháp dựa trên NoSQL nói chung cung cấp một cách mạnh mẽ, có khả năng mở rộng và linh hoạt để giải quyết các nhu cầu dữ liệu và các trường hợp sử dụng, mà trước đó đã được quản lý bởi các CSDL quan hệ. Để tóm tắt các tùy chọn NoSQL, chúng ta sẽ kiểm tra các mô hình hoặc danh mục phổ biến nhất:

- **Key/Value databases:** Lưu trữ kiểu key/value là kiểu lưu trữ dữ liệu NoSQL đơn giản nhất sử dụng từ một API. Chúng ta có thể nhận được giá trị cho khóa, đặt một giá trị cho một khóa, hoặc xóa một khóa từ dữ liệu. Ví dụ, giá trị là 'blob' được lưu trữ thì chúng ta không cần quan tâm hoặc biết những gì ở bên trong. Từ các cặp giá trị được lưu trữ luôn luôn sử dụng truy cập thông qua khóa chính và thường có hiệu năng truy cập tốt và có thể dễ dàng thu nhỏ lại. Một vài CSDL key/value phổ biến là Riak, Redis (thường dùng phía server), memcached, HamsterDB.... Tất cả các CSDL kiểu key/value đều không giống nhau, có rất nhiều điểm khác nhau giữa các sản phẩm. Ví dụ, dữ liệu của memcached không được nhất quán trong khi ngược lại với Riak. Đây là những điểm nổi bật quan trọng khi chọn giải pháp phù hợp để sử dụng. Cụ thể hơn là khi chúng ta cần cài đặt caching cho nội dung yêu thích của người dùng, cài đặt sử dụng memcached có nghĩa là khi các nút hỏng hết dẫn tới dữ liệu bị mất và cần phải làm mới lại từ hệ thống nguồn. Tuy nhiên, nếu chúng ta lưu trữ cùng dữ liệu đó trong Riak, chúng ta không cần lo lắng về việc mất dữ liệu nhưng cần phải xem xét việc cập nhật trạng thái của dữ liệu như thế nào. Điều này là quan trọng

không chỉ cho việc chọn CSDL key/value cho hệ thống mà còn quan trọng cho việc chọn bất kỳ CSDL key/value nào.

- **Column-oriented databases:**CSDL column-oriented databases lưu trữ dữ liệu trong nhiều cột trong mỗi dòng với key cho từng dòng. Column-oriented databases là một nhóm các dữ liệu liên quan được truy cập cùng với nhau.Ví dụ,với khách hàng,chúng ta thường xuyên sử dụng thông tin cá nhân trong cùng một lúc chứ không phải hóa đơn của họ.Cassandra là một trong số CSDL column-oriented databases phổ biến.Ngoài ra,còn có một số CSDL khác như Hbase,Hypertable,Amazon DynamoDB.Cassandra có thể được miêu tả nhanh và khả năng mở rộng dễ dàng với các thao tác viết thông qua các cụm.Các cụm không có node master,vì thế bất kỳ việc đọc và ghi nào đều có thể được xử lý bởi bất kỳ node nào trong cụm.
- **Document databases:**Tài liệu là nguyên lý chính của CSDL kiểu dữ liệu.Dữ liệu lưu trữ và lấy ra là các tài liệu định dạng XML,JSON,BSON...Tài liệu miêu tả chính nó,kế thừa từ cấu trúc dữ liệu cây.Có thể nói CSDL tài liệu là 1 phần của Key/Value.CSDL kiểu tài liệu như MongoDB cung cấp ngôn ngữ truy vấn đa dạng và cấu trúc như là CSDL như đánh index...Một số CSDL tài liệu phổ biến mà chúng ta hay gặp là MongoDB,CouchDB,OrientDB...
- **Graph databases:**Kiểu đồ thị này cho phép bạn lưu trữ các thực thể và quan hệ giữa các thực thể.Các đối tượng này còn được gọi là các nút,trong đó có các thuộc tính.Mỗi nút là một thể hiện của một đối tượng trong ứng dụng.Quan hệ được gọi là các cạnh,có thể có các thuộc tính.Cạnh có ý nghĩa định hướng,các nút được tổ chức bởi các mối quan hệ.Các tổ chức của đồ thị cho phép các dữ liệu được lưu trữ một lần và được giải thích theo nhiều cách khác nhau dựa trên các mối quan hệ.Thông thường,khi chúng ta lưu trữ một cấu trúc đồ thị giống như trong RDBMS,nó là một loại duy nhất của mối quan hệ.Việc tăng thêm một mối quan hệ có nghĩa là rất nhiều thay đổi sơ đồ và di chuyển dữ liệu,mà không phải là trường hợp khó khăn chúng ta đang sử dụng CSDL đồ thị.Trong CSDL đồ thị,bằng qua các thành phần tham gia hoặc các mối quan hệ là rất nhanh.Các mối quan hệ giữa các node không được tính vào thời gian truy vấn nhưng thực sự tồn tại như là một mối quan hệ.Đi qua các mối quan hệ là nhanh hơn so với tính toán cho mỗi truy vấn.Có rất nhiều CSDL đồ thị có sẵn,chẳng hạn như Neo4J,OrientDB...

OrientDB là một CSDL document-graph,có nghĩa là nó có đầy đủ các khả năng đồ thị gốc cùng với các tính năng thông thường của document databases.

Mỗi hạng mục hoặc mô hình có đặc điểm riêng biệt,điểm mạnh và giới hạn riêng.Không có danh mục hoặc mô hình đơn lẻ nào tốt hơn các loại khác.Tuy nhiên,một số loại CSDL nhất định sẽ giải quyết các vấn đề cụ thể hơn.Điều này dẫn đến khẩu hiệu “chọn công cụ tốt

nhất cho trường hợp sử dụng cụ thể của bạn”.

Mục tiêu của những công nghệ Orient trong việc xây dựng OrientDB là tạo ra các CSDL thiết thực có khả năng mở rộng có thể thực hiện tối ưu trong các trường hợp sử dụng rộng rãi nhất có thể. Sản phẩm được thiết kế để trở thành một giải pháp “đi tới” một kết quả tuyệt vời cho tất cả các nhu cầu kiên trì dữ liệu. Trong các phần sau chúng ta sẽ xem xét kỹ OrientDB, một trong những sản phẩm NoSQL mã nguồn mở, đa mô hình, thể hệ tiếp theo tốt nhất trên thị trường hiện nay.

1.1.4 Cài đặt OrientDB version 2.2.x, Run the Server, Run the Console, Run the studio

- OrientDB có hai phiên bản:
 - Community Edition (phiên bản cộng đồng) là được phát hành dưới dạng một dự án mã nguồn mở bản quyền Apache 2. Bản quyền này cho phép sử dụng miễn phí không giới hạn cho cả dự án mã nguồn mở và thương mại.
 - Enterprise Edition (phiên bản doanh nghiệp) là phần mềm thương mại được xây dựng trên phiên bản cộng đồng. Doanh nghiệp được phát triển bởi cùng một nhóm phát triển động cơ OrientDB. Nó hoạt động như một phần mở rộng của phiên bản cộng đồng, cung cấp các tính năng Enterprise (doanh nghiệp) chẳng hạn như:
 - * Không ngừng sao lưu và khôi phục.
 - * Đã được lên lịch đầy đủ và gia tăng sao lưu.
 - * Truy vấn sơ lược.
 - * Được phân phối hình dạng cụm.
 - * Ghi số liệu.
 - Phiên bản cộng đồng là một thư viện dạng package có thể tải về hoặc lấy source code trên GitHub. Phiên bản doanh nghiệp thì phải mua mới sử dụng được.
 - Cài đặt Orient

```
huynhlieu@LIEU-K501LB ~ $ docker run -d --name orientdb -p 2424:2424 -p 2480:2480 -e ORIENTDB_ROOT_PASSWORD=root orientdb:2.2.36
Unable to find image 'orientdb:2.2.36' locally
2.2.36: Pulling from library/orientdb
8e3ballec2a2: Pull complete
311ad0da4533: Pull complete
df312c74ce16: Downloading [=====] 65.39MB/70.58MB
24f24a3fc688: Download complete
271f35ad3bda: Downloading [=====] 29.8MB/46.47MB
```

HÌNH 1.2. trên Docker: set thuộc tính ORIENTDB_ROOT_PASSWORD để thay đổi mật khẩu truy cập của bạn.

- Điều kiện tiên quyết: Để phiên bản của OrientDB chạy trên bất kỳ hệ điều hành nào triển khai máy ảo
 - * Linux,tất cả sự phân bố,bao gồm ARM. Mac OS X.
 - * Microsoft Windows,từ 95/NT và later.
 - * HP-UX
 - * IBM AIX
- Cài đặt gói nhị phân
 - * OrientDB cung cấp một trình biên dịch nhị phân package cài đặt database vào hệ thống. Phụ thuộc vào hệ điều hành của bạn,nó là một gói tarred hoặc zip chứa tất cả các tệp có liên quan mà bạn cần cho OrientDB. Cài đặt trên Desktop,bạn cần chọn package tốt nhất và hợp lý nhất cho hệ thống của bạn.
 - * Cài đặt trên Server,bạn có thể sử dụng wget sẽ tiện ích hơn: Cho dù bạn sử dụng web browser hay wget,giải nén hoặc nén tệp đã tải xuống thì nên lưu vào một thư mục để tiện quản lý(ví dụ: /opt/orientdb/ trên Linux) .Nó tạo một thư mục có tên là orientdb-community-2.2.36 với các tệp và các dữ liệu có liên quan mà chúng ta cần để chạy OrientDB trên hệ thống của chúng ta.
- Cài đặt mã nguồn
 - * Ngoài cài đặt các gói nhị phân,bạn cũng có thể tùy chọn biên dịch OrientDB từ mã nguồn của phiên bản cộng đồng,có sẵn trên GitHub.Quá trình yêu cầu bạn phải cài đặt git và Apache Maven trong hệ thống của bạn
 - * Để biên dịch OrientDB từ mã nguồn,sao chép kho lưu trữ phiên bản cộng đồng,sau đó chạy maven(mvn) trong thư mục mới được tạo ra:



```

huynhlieu@LIEU-K501LB ~ $ wget https://s3.us-east-2.amazonaws.com/orientdb3/releases/2.2.36/orientdb-community-2.2.36.zip -O orientdb-community-2.2.36.zip
--2018-08-07 16:14:12-- https://s3.us-east-2.amazonaws.com/orientdb3/releases/2.2.36/orientdb-community-2.2.36.zip
Resolving s3.us-east-2.amazonaws.com (s3.us-east-2.amazonaws.com)... 52.219.88.194
Connecting to s3.us-east-2.amazonaws.com (s3.us-east-2.amazonaws.com)|52.219.88.194|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 52878735 (50M) [application/zip]
Saving to: 'orientdb-community-2.2.36.zip'
orientdb-community-2.2.36.zip      0%[                               ] 8,00K  24,8KB/s

huynhlieu@LIEU-K501LB ~ $ git clone https://github.com/orientechnologies/orientdb
Cloning into 'orientdb'...
remote: Counting objects: 428374, done.
remote: Compressing objects: 100% (223/223), done.
Receiving objects: 0% (1/428374)
  
```

HÌNH 1.3. Download orientdb trên docker

- Nhánh phát triển chứa mã cho phiên bản tiếp theo của OrientDB phiên bản,phiên bản ổn định được gắn trên nhánh chính,đối với mỗi phiên bản duy trì OrientDB có nhánh hotfix riêng của nó.Khi viết ghi chú này trạng thái của các nhánh là:
 - * Phát triển:Công việc đang tiến hành cho bản phát hành 3.0.x tiếp theo(3.0.x-SNAPSHOT).

- * 2.2.x: hot fix cho phiên bản phát hành ổn định 2.2.x tiếp theo(2.2.x-SNAPSHOT).
 - * 2.1.x:hot fix cho phiên bản phát hành ổn định 2.1.x tiếp theo(2.1.x-SNAPSHOT).
 - * 2.0.x:hot fix cho phiên bản phát hành ổn định 2.0.x tiếp theo(2.0.x-SNAPSHOT).
- Nhiệm vụ sau khi cài đặt: Đối với những Desktop cài đặt nhị phân,OrientDB hiện đã cài đặt và có thể chạy qua các tệp shell được tìm thấy trong thư mục bin của gói đã được cài đặt,có một số bước bổ sung mà bạn cần thực hiện để quản lý máy chủ CSDL cho orientDB như một dịch vụ.Quy trình này phụ thuộc vào hệ điều hành của bạn.
 - Nâng cấp. Khi thời gian đến để nâng cấp lên phiên bản mới của OrientDB,các phương pháp thường khác nhau tùy thuộc vào cách chọn phiên bản ở lần đầu tiên.Nếu đã cài đặt nhị phân,hãy lặp lại quy trình tải xuống ở trên và cập nhật mọi liên kết hoặc phím tắt tượng trưng để trở đến thư mục mới. Đối với các hệ thống được phiên dịch từ mã nguồn,hãy pull về mã nguồn mới nhất và phiên dịch chúng.
 - Chạy server. Sau khi hoàn thành việc cài đặt OrientDB,cho dù bạn build nó từ mã nguồn hay tải xuống gói nhị phân,bạn đã sẵn sàng để khởi động máy chủ CSDL.Bạn có thể khởi động nó thông qua trình nền hệ thống hoặc thông qua tập lệnh máy chủ được cung cấp. Nếu bạn muốn chạy OrientDB như một dịch vụ trên hệ thống của bạn,có một số bước bổ sung mà bạn cần phải thực hiện.Điều này cung cấp các phương thức thay thế để khởi động máy chủ và cho phép bạn khởi chạy nó như một daemon khi khởi động hệ thống của bạn.
 - * Khởi chạy máy chủ CSDL: Trong khi bạn có thể chạy máy chủ CSDL dưới dạng daemon hệ thống,bạn cũng có thể chọn khởi động trực tiếp.Trong thư mục cài đặt OrientDB(nó là \$ORIENTDB_HOME),dưới mục bin,là 1 file server.sh trên hệ thống dựa trên Unix và server.bat trên Windows.Thực thi file để bắt đầu chạy server.
 - * Để khởi động máy chủ CSDL OrientDB,chạy câu lệnh. Máy chủ CSDL đang chạy.Nó có thể truy cập trên hệ thống của bạn thông qua các cổng 2424 và 2480.Lúc khởi động lần đầu tiên máy chủ sẽ yêu cầu mật khẩu người dùng root.Mật khẩu sẽ được lưu trữ trong tập tin cấu hình.
 - Dừng Server. Trên cửa sổ nơi mà server đang chạy cách đơn giản nhất là nhấn tổ hợp phím CTRL + c thì server sẽ shutdown. File shutdown.sh(shutdown.bat) có thể sử dụng để dừng server.

```

huyhnlwIEU-K501LB ~ % cd orientdb
huyhnlwIEU-K501LB ~ % ./orientdb $mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] Reactor Build Order:
[INFO]
[INFO] OrientDB
[INFO] OrientDB Test Commons
[INFO] OrientDB Core
[INFO] OrientDB Client
[INFO] OrientDB Object
[INFO] OrientDB Tools
[INFO] OrientDB Server
[INFO] OrientDB GraphDB
[INFO] OrientDB Tests
[INFO] OrientDB Distributed Server
[INFO] OrientDB Lucene full text index
[INFO] OrientDB JDBC Driver
[INFO] OrientDB ETL
[INFO] OrientDB Community Distribution
[INFO] OrientDB Community Distribution Tinker Pop 2
[INFO] OrientDB Crash Tests
[INFO]
[INFO]
[INFO] -----
[INFO] Building OrientDB 3.1.0-SNAPSHOT
[INFO] -----
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/3.1.0/maven-clean-plugin-3.1.0.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/3.1.0/maven-clean-plugin-3.1.0.pom (6 KB at 2.7 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/31/maven-plugins-31.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/31/maven-plugins-31.pom (11 KB at 10.1 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/31/maven-parent-31.pom (43 KB at 40.2 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/apache/19/apache-19.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/apache/19/apache-19.pom (16 KB at 29.6 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/3.1.0/maven-clean-plugin-3.1.0.jar
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/3.1.0/maven-clean-plugin-3.1.0.jar (30 KB at 49.9 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/mojo/buildnumber-maven-plugin/1.4/buildnumber-maven-plugin-1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/mojo/buildnumber-maven-plugin/1.4/buildnumber-maven-plugin-1.4.pom (14 KB at 30.5 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/mojo/buildnumber-maven-plugin/1.4/buildnumber-maven-plugin-1.4.jar
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/mojo/buildnumber-maven-plugin/1.4/buildnumber-maven-plugin-1.4.jar (48 KB at 64.1 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/felix/maven-bundle-plugin/3.5.0/maven-bundle-plugin-3.5.0.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/felix/maven-bundle-plugin/3.5.0/maven-bundle-plugin-3.5.0.pom (10 KB at 24.2 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/felix/felix-parent/4/felix-parent-4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/felix/felix-parent/4/felix-parent-4.pom (24 KB at 48.2 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/felix/maven-bundle-plugin/3.5.0/maven-bundle-plugin-3.5.0.jar
Downloaded: https://repo.maven.apache.org/maven2/org/apache/felix/maven-bundle-plugin/3.5.0/maven-bundle-plugin-3.5.0.jar (227 KB at 64.5 KB/sec)
[INFO]

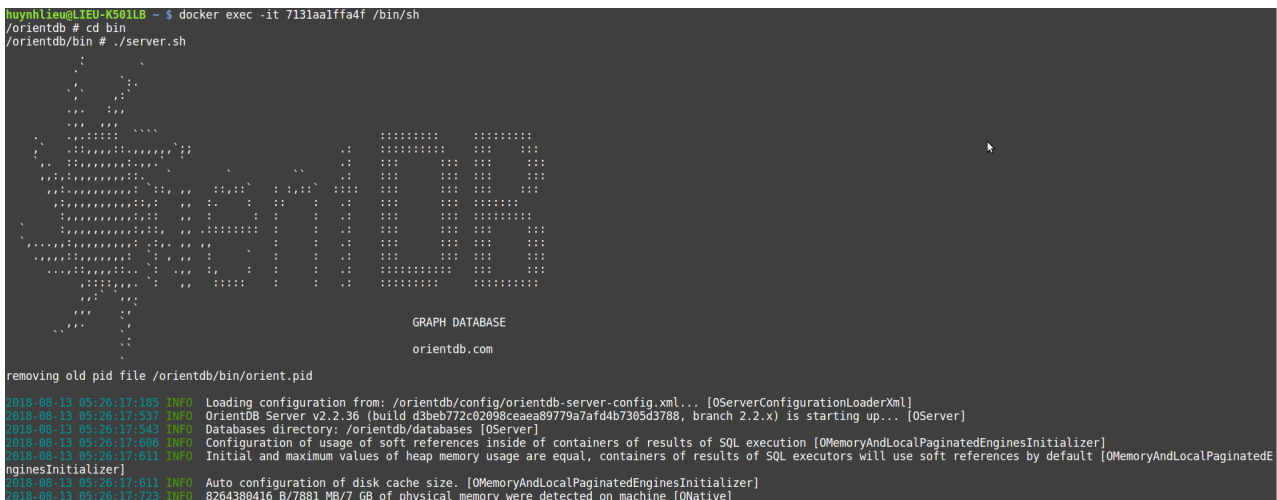
```

HÌNH 1.4. Chạy orientDb trên docker

- Trên hệ thống nix một cách đơn giản gọi shutdown.sh sẽ dừng việc server chạy trên localhost Danh sách các thông số.
 - * -h | --host HOSTNAME hoặc IP ADDRESS:host hoặc ip nơi mà OrientDB đang chạy,mặc định là localhost.
 - * -P | --ports PORT hoặc PORT RANGE: giá trị port đơn hoặc hạng của ports,mặc định từ 2424-2430.
 - * -u | --user ROOT USERNAME: username của root,mặc định là root.
 - * -p | --password ROOT PASSWORD: password user của root(có tính bắt buộc)
- Thông báo nhật ký máy chủ:
 - * Máy chủ CSDL bắt đầu in thông điệp tường trình thành đầu ra tiêu chuẩn,Điều này sẽ cung cấp tất cả các hành động của OrientDB khi nó hoạt động trên hệ thống của bạn.
 - * Máy chủ CSDL tải tệp cấu hình của nó từ file \$ORIENTDB_HOME/config/orientdb-server-config.xml.
 - * Máy chủ CSDL tải CSDL tạm thời vào bộ nhớ.Có thể sử dụng CSDL này để lưu trữ dữ liệu tạm thời.
 - * Máy chủ CSDL bắt đầu lắng nghe các kết nối nhậ phân từ cổng 2424 cho tất cả các mạng được cấu hình(0 . 0 . 0 . 0).
 - * Máy chủ CSDL bắt đầu lắng nghe các kết nối HTTP từ cổng 2480 cho tất cả các mạng được cấu hình(0 . 0 . 0 . 0).

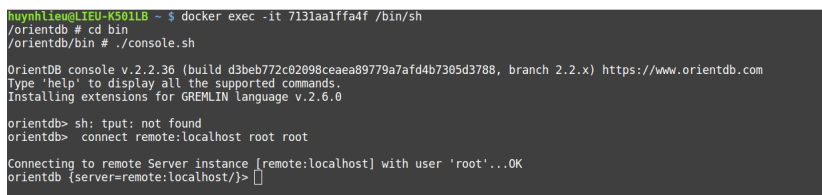
- Chạy OrientDB trên Console

- * Khởi chạy OrientDB Console Trong thư mục cài đặt OrientDB(có nghĩa là \$ORIENTDB_HOME,nơi bạn đã cài đặt CSDL) trong bin có một tệp gọi là console.sh cho các hệ thống dựa trên Unix hoặc console.bat đối với các User Windows.
- * Để khởi động Orient Console,chạy câu lệnh dưới đây sau khi bạn đã khởi động máy chủ CSDL.OrientDB console đang chạy.Từ đó bạn có thể nhanh chóng kết nối và quản lý mọi CSDL từ xa hoặc cục bộ có sẵn.



HÌNH 1.5. Chạy orientDb trên docker bằng console

- Sử dụng câu lệnh HELP. Trong trường hợp bạn không quen với OrientDB và những câu lệnh có sẵn hoặc nếu bạn cần hỗ trợ, bạn có thể sử dụng câu lệnh HELP hoặc gõ ? vào dấu nhắc console.
- Đối với mỗi lệnh console có sẵn cho bạn, HELP sử dụng tài liệu cơ bản của nó. Nếu bạn biết lệnh cụ thể và cần chi tiết về việc sử dụng nó, bạn có thể cung cấp các đối số cho HELP để có thể thấy rõ hơn.
- Kết nối đến Server. Để kết nối với cá thể CSDL trên máy của bạn



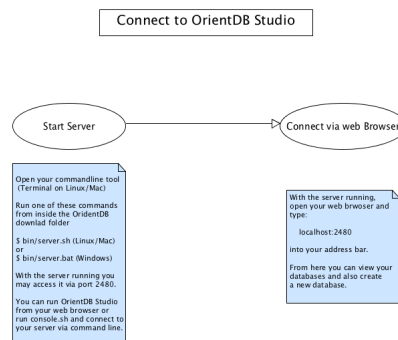
HÌNH 1.6. Kết nối với DB

- Xem danh sách các database đang có

```
orientdb {server=remote:localhost/}> list databases
Found 0 databases:
orientdb {server=remote:localhost/}> █
```

HÌNH 1.7. Xem danh sách các database đang có

- Chạy trên Studio. Nếu bạn muốn tiện lợi hơn khi có sự tương tác qua lại với hệ thống CSDL thì với một giao diện đồ họa sau đây bạn có thể hoàn thành tốt hơn công việc cộng đồng CSDL với OrientDB, giao diện web.

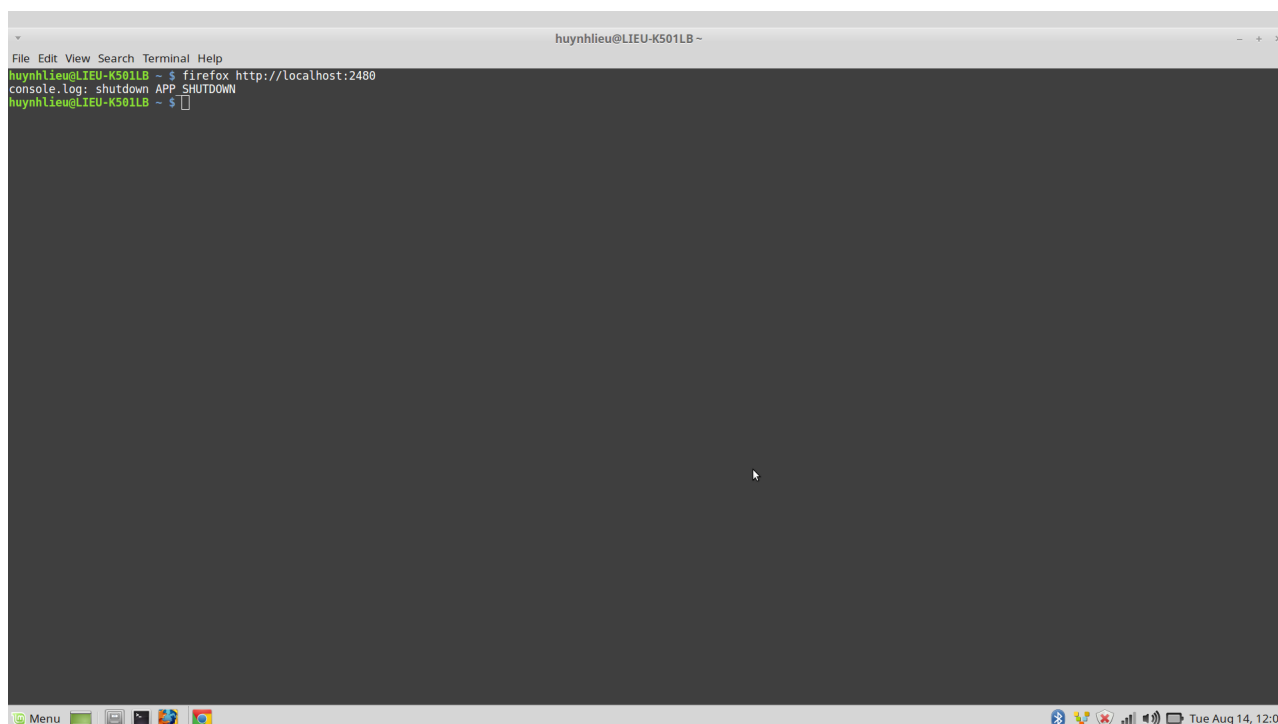


HÌNH 1.8. Kết nối Studio

(a) <https://orientdb.com/docs/>

- Kết nối Studio

- Mặc định, không cần thêm quá nhiều bước trong việc bạn khởi động OrientDB Studio. Khi bạn khởi động Server, dù là bạn bắt đầu với file server.sh hay là hệ thống daemon, giao diện web Studio mở tự động với nó.
- Từ đây bạn có thể tạo CSDL mới, kết nối hoặc xóa CSDL bạn muốn, thêm vào CSDL và điều khiển hay quản lý giao diện của Server.



HÌNH 1.9. Truy cập Studio bằng lệnh

(a) <https://orientdb.com/docs/>

- Giống nhiều hệ thống quản lý CSDL, OrientDB sử dụng record để lưu trữ, có nhiều loại bảng ghi, nhưng với Document Database API, bảng ghi luôn luôn sử dụng kiểu Document. Documents được tạo thành bởi tập hợp các cặp key/value được gọi là trường và thuộc tính và có thể thuộc lớp.
- Lớp là một khái niệm được rút ra từ mô hình hướng đối tượng. Nó là một kiểu của mô hình dữ liệu điều đó cho phép bạn định nghĩa các bản ghi thuộc về nó. Trong mô hình CSDL Document truyền thống nó có thể so sánh với bộ sưu tập. Trong khi trong mô hình CSDL quan hệ, nó có thể so sánh được với bảng.
- Tất cả danh sách cấu hình trên hệ thống của bạn, sử dụng câu LIST CLASSES để xem trên màn hình.

```

File Edit View Search Terminal Help
huynhlieu@LIEU-K501LB ~
/bin/sh: list: not found
/orientdb/bin # ./console.sh

OrientDB console v.2.2.36 (build d3beb772c02098ceaea89779a7afd4b7305d3788, branch 2.2.x) https://www.orientdb.com
Type 'help' to display all the supported commands.
Installing extensions for GREMLIN language v.2.6.0

orientdb> sh: tput: not found
orientdb> list classes

No database selected yet.
orientdb> list cluster

!Unrecognized command: 'list cluster'
orientdb> list clusters

No database selected yet.
orientdb> list clusters

No database selected yet.
orientdb> connect remote:localhost/Test root root

Connecting to database [remote:localhost/Test] with user 'root'...OK
orientdb (db=Test)> list classes

CLASSES
+-----+-----+-----+-----+
|#|NAME|SUPER-CLASSES|CLUSTERS|COUNT|
+-----+-----+-----+-----+
10|studio|_|studio(17)|2|
11|E|_|e(13),e 1(14),e 2(15),e 3(16)|0|
12|HoangXuanToan|_|hoangxuantan(18),hoangxuantan_1(19),hoangxuantan_2(20),hoangxuantan_3(21)|1|
13|OFunction|_|ofunction(6)|0|
14|OIdentity|_|_|0|
15|ORestricted|_|_|0|
16|ORole|[OIdentity]|orole(4)|3|
17|OSchedule|_|oschedule(8)|0|
18|OSequence|_|osequence(7)|0|
19|OTriggered|_|_|0|
10|OUser|[OIdentity]|ouser(5)|3|
11|V|_|v(9),v 1(10),v 2(11),v 3(12)|0|
+-----+-----+-----+-----+
|TOTAL|_|_|9|
+-----+-----+-----+-----+

orientdb (db=Test)>

```

HÌNH 1.10. Tất cả danh sách class trên hệ thống của bạn

– Làm việc với Classes

- * Để tạo mới một Classes,sử dụng câu lệnh

```

File Edit View Search Terminal Help
5b14ccb77004 hello-world "/hello" 7 weeks ago Exited (0) 7 weeks ago hello-world
ea36a6e9ebb6 bitnami/tomcat "/app-entrypoint.s..." 2 months ago Exited (0) 7 weeks ago tomcat
fe82a3e3b76d springio/gs-spring-boot-docker "java -Djava.secur..." 2 months ago Exited (143) 2 months ago helloworld
1fea6341113d mysql:5.7 "docker-entrypoint..." 4 months ago Exited (0) 6 weeks ago gifted_easley
huynhlieu@LIEU-K501LB ~ $ docker exec -it 7131aa1ffa4f /bin/sh
/orientdb # cd bin
/orientdb/bin # ./console.sh

OrientDB console v.2.2.36 (build d3beb772c02098ceaea89779a7afd4b7305d3788, branch 2.2.x) https://www.orientdb.com
Type 'help' to display all the supported commands.
Installing extensions for GREMLIN language v.2.6.0

orientdb> sh: tput: not found
orientdb> create class student
Error: com.orienttechnologies.common.exception.OSystemException: Database not selected. Use 'connect <database-name>' to connect to a database.

orientdb> create class test12
Error: com.orienttechnologies.common.exception.OSystemException: Database not selected. Use 'connect <database-name>' to connect to a database.

orientdb> connect remote:localhost/Test root root
Connecting to database [remote:localhost/Test] with user 'root'...OK
orientdb (db=Test)> create class student
Error: com.orienttechnologies.orient.core.exception.OSchemaException: Class 'student' already exists in current database
DB name="Test"
DB name="Test"

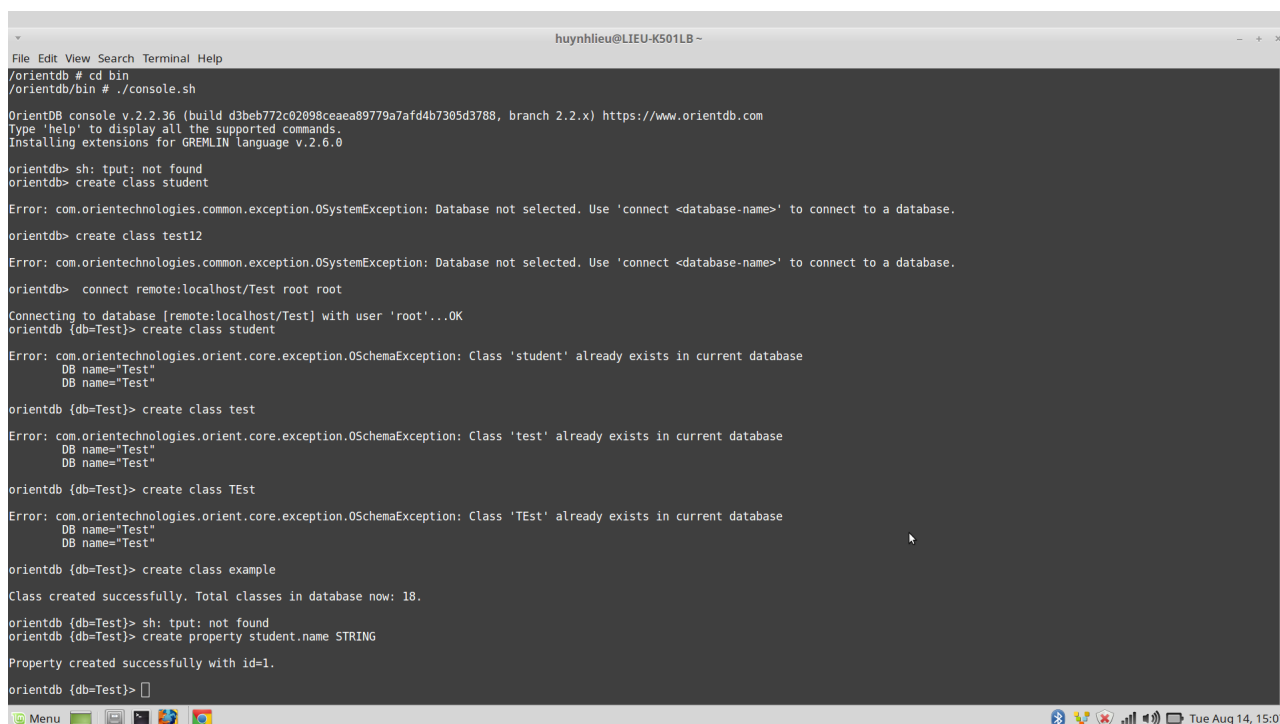
orientdb (db=Test)> create class test
Error: com.orienttechnologies.orient.core.exception.OSchemaException: Class 'test' already exists in current database
DB name="Test"
DB name="Test"

orientdb (db=Test)> create class Test
Error: com.orienttechnologies.orient.core.exception.OSchemaException: Class 'Test' already exists in current database
DB name="Test"
DB name="Test"

orientdb (db=Test)> create class example
Class created successfully. Total classes in database now: 18.
orientdb (db=Test)>
  
```

HÌNH 1.11. Tạo lớp

* Thêm một thuộc tính vào class.



```
File Edit View Search Terminal Help
huynhlieu@LIEU-K501LB ~
/orientdb # cd bin
/orientdb/bin # ./console.sh

OrientDB console v.2.2.36 (build d3beb772c02098ceaea89779a7afd4b7305d3788, branch 2.2.x) https://www.orientdb.com
Type 'help' to display all the supported commands.
Installing extensions for GREMLIN language v.2.6.0

orientdb> sh: tput: not found
orientdb> create class student
Error: com.orienttechnologies.common.exception.OSystemException: Database not selected. Use 'connect <database-name>' to connect to a database.
orientdb> create class test12
Error: com.orienttechnologies.common.exception.OSystemException: Database not selected. Use 'connect <database-name>' to connect to a database.
orientdb> connect remote:localhost/Test root root
Connecting to database [remote:localhost/Test] with user 'root'...OK
orientdb (db=Test)> create class student
Error: com.orienttechnologies.orient.core.exception.OSchemaException: Class 'student' already exists in current database
DB name="Test"
DB name="Test"
orientdb (db=Test)> create class test
Error: com.orienttechnologies.orient.core.exception.OSchemaException: Class 'test' already exists in current database
DB name="Test"
DB name="Test"
orientdb (db=Test)> create class TEst
Error: com.orienttechnologies.orient.core.exception.OSchemaException: Class 'TEst' already exists in current database
DB name="Test"
DB name="Test"
orientdb (db=Test)> create class example
Class created successfully. Total classes in database now: 18.
orientdb (db=Test)> sh: tput: not found
orientdb (db=Test)> create property student.name STRING
Property created successfully with id=1.
orientdb (db=Test)> 
```

HÌNH 1.12. Thêm một thuộc tính vào class

* Hiển thị thông tin của class.

```

File Edit View Search Terminal Help
huynhthieu@LIEU-K501LB ~
orientdb (db=Test)> sh: tput: not found
orientdb (db=Test)> create property student.name STRING
Property created successfully with id=1.
orientdb (db=Test)> sh: tput: not found
orientdb (db=Test)> info class example
CLASS 'example'
Records.....: 0
Default cluster.....: example (id=42)
Supported clusters...: example(42), example_1(43), example_2(44), example_3(45)
Cluster selection....: round-robin
Oversize.....: 0.0
orientdb (db=Test)> sh: tput: not found
orientdb (db=Test)> create property example.name STRING
Property created successfully with id=1.
orientdb (db=Test)> create property example.age INTEGER
Property created successfully with id=2.
orientdb (db=Test)> create property example.lop STRING
Property created successfully with id=3.
orientdb (db=Test)> info class example
CLASS 'example'
Records.....: 0
Default cluster.....: example (id=42)
Supported clusters...: example(42), example_1(43), example_2(44), example_3(45)
Cluster selection....: round-robin
Oversize.....: 0.0
PROPERTIES
+-----+-----+-----+-----+-----+-----+-----+-----+
|#|NAME|TYPE|LINKED-TYPE/CLASS|MANDATORY|READONLY|NOT-NULL|MIN|MAX|COLLATE|DEFAULT|
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|name|STRING|false|false|false|||default|
|1|lop|STRING|false|false|false|||default|
|2|age|INTEGER|false|false|false|||default|
+-----+-----+-----+-----+-----+-----+-----+
orientdb (db=Test)>

```

HÌNH 1.13. Hiển thị thông tin của class.

- * Thêm các ràng buộc vào các thuộc tính. Các ràng buộc tạo ra những giới hạn trên giá trị dữ liệu của những thuộc tính. Ví dụ như giá trị nhỏ nhất hoặc giá trị lớn nhất của kích thước, hoặc có hay không giá trị bắt buộc phải có hoặc là nếu giá trị là cho phép null.

```

CLASS 'example'
Records.....: 0
Default cluster.....: example (id=42)
Supported clusters....: example(42), example_1(43), example_2(44), example_3(45)
Cluster selection.....: round-robin
Oversize.....: 0.0

PROPERTIES
+-----+-----+-----+-----+-----+-----+-----+-----+
# | NAME | TYPE | LINKED-TYPE/CLASS | MANDATORY | READONLY | NOT-NULL | MIN | MAX | COLLATE | DEFAULT |
+-----+-----+-----+-----+-----+-----+-----+-----+
0 | name | STRING | | false | false | false | | | | default |
1 | lop | STRING | | false | false | false | | | | default |
2 | age | INTEGER | | false | false | false | | | | default |
+-----+-----+-----+-----+-----+-----+-----+-----+

orientdb (db=Test)> sh: tput: not found
orientdb (db=Test)> alter property example.name MIN 3

Error: com.orienttechnologies.orient.core.exception.OCommandExecutionException: Property 'example.name' not exists
DB name="Test"
DB name="Test"

orientdb (db=Test)> info class example

CLASS 'example'
Records.....: 0
Default cluster.....: example (id=42)
Supported clusters....: example(42), example_1(43), example_2(44), example_3(45)
Cluster selection.....: round-robin
Oversize.....: 0.0

PROPERTIES
+-----+-----+-----+-----+-----+-----+-----+-----+
# | NAME | TYPE | LINKED-TYPE/CLASS | MANDATORY | READONLY | NOT-NULL | MIN | MAX | COLLATE | DEFAULT |
+-----+-----+-----+-----+-----+-----+-----+-----+
0 | name | STRING | | false | false | false | | | | default |
1 | lop | STRING | | false | false | false | | | | default |
2 | age | INTEGER | | false | false | false | | | | default |
+-----+-----+-----+-----+-----+-----+-----+-----+

orientdb (db=Test)> alter property example.name MIN 3

Property updated successfully.

orientdb (db=Test)>

```

HÌNH 1.14. Lệnh ALTER PROPERTY

- Trường(record) ID Muốn tải trực tiếp record, bạn có thể sử dụng câu lệnh LOAD RECORD trên cửa sổ console.

```

27 | student_1 | 23 | Student | 0 |
28 | student_2 | 24 | Student | 0 |
29 | student_3 | 25 | Student | 0 |
30 | test | 34 | TEST | 0 |
31 | test1 | 38 | test1 | 0 |
32 | test1_1 | 31 | test1 | 0 |
33 | test1_2 | 32 | test1 | 0 |
34 | test1_3 | 33 | test1 | 0 |
35 | test_1 | 35 | TEST | 0 |
36 | test_2 | 36 | TEST | 0 |
37 | test_3 | 37 | TEST | 0 |
38 | test_classes | 38 | test_classes | 0 |
39 | test_classes_1 | 39 | test_classes | 0 |
40 | test_classes_2 | 40 | test_classes | 0 |
41 | test_classes_3 | 41 | test_classes | 0 |
42 | v | 9 | V | 0 |
43 | v_1 | 10 | V | 0 |
44 | v_2 | 11 | V | 0 |
45 | v_3 | 12 | V | 0 |
+-----+-----+-----+-----+
| TOTAL | | | 13 |
+-----+-----+-----+-----+

orientdb (db=Test)> select from example
0 item(s) found. Query executed in 0.001 sec(s).
orientdb (db=Test)> select from hoangxuanloan
+-----+-----+-----+-----+
|# |@RID |@CLASS |gioi_tinh|
+-----+-----+-----+-----+
|0 |#18:0 |HoangXuanLoan |false |
+-----+-----+-----+-----+

1 item(s) found. Query executed in 0.003 sec(s).
orientdb (db=Test)> load record #18:0
DOCUMENT @class:HoangXuanLoan @rid:#18:0 @version:1
+-----+-----+-----+-----+
|# |NAME |VALUE|
+-----+-----+-----+-----+
|0 |gioi_tinh |false|
+-----+-----+-----+-----+

OK
orientdb (db=Test)>

```

HÌNH 1.15. Trường(record)

- Một trong những chức năng quan trọng của CSDL đồ thị là nằm ở cách chúng quản lý các mối quan hệ. Nhiều người bắt đầu sử dụng OrientDB từ MongoDB do OrientDB hỗ trợ hiệu quả hơn cho các mối quan hệ.
 - * **Quan hệ trong CSDL quan hệ.** Hầu hết các nhà phát triển CSDL điều quen thuộc với mô hình CSDL quan hệ và với hệ thống quản lý CSDL quan hệ, chẳng hạn như MySQL và MS-SQL. Đã hơn 30 năm chiếm ưu thế, điều này từ lâu đã được cho là cách tốt nhất để xử lý các mối quan hệ. Ngược lại, CSDL đồ thị đề xuất một cách tiếp cận hiện đại hơn cho khái niệm này. Thường trong CSDL quan hệ sẽ có các mối quan hệ 1-1, 1-n và n-n.
 - * **Quan hệ trong OrientDB.** Trong OrientDB không có JOIN. Thay vào đó, nó sử dụng LINK. LINK là một mối quan hệ được quản lý bằng cách lưu trữ ID bản ghi đích trong bản ghi nguồn. Nó tương tự như lưu trữ con trỏ giữa hai đối tượng trong bộ nhớ. Trong mối quan hệ 1-n, OrientDB xử lý mối quan hệ như một tập hợp các ID bản ghi, giống như một tập hợp các ID bản ghi, giống như khi bạn quản lý các đối tượng trong bộ nhớ.
- OrientDB hỗ trợ một số loại mối quan hệ khác nhau:
 - * **LINK** : mối quan hệ chỉ vào một điểm ghi.

- * **LINKSET** : Mỗi quan hệ trỏ đến một số chỉ mục.Nó tương tự như các bộ trong Java,cùng một ID bản ghi chỉ có thể bao gồm 1 lần.Các con trỏ không có thứ tự
- * **LINKLIST** : Mỗi quan hệ trỏ đến một số hồ sơ.Nó tương tự như các danh sách Java,chúng được sắp xếp và có thể chứa bản sao.
- * **LINKMAP** : Mỗi quan hệ trỏ đến một số bản ghi với một khóa được lưu trữ trong bản ghi nguồn.Các giá trị Map là ID của bản ghi.Nó tương tự như Java Map<?,Record>.

Chương 2

MÔ HÌNH DỮ LIỆU

- **Đa mô hình(Multi-Model)** Cơ cấu mô hình OrientDB hỗ trợ Graph,Document,Key/Value và Object,vì thế bạn có thể sử dụng OrientDB như một sự thay thế cho một kết quả trong bất kỳ danh mục nào trong số chúng.Tuy nhiên,lý do chính tại sao người dùng chọn OrientDB là vì các khả năng DBMS đa mô hình thật sự của nó,kết hợp tất cả các tính năng của bốn mô hình vào.Những khả năng này không chỉ là giao diện cho CSDL mà đúng hơn là bản thân cơ cấu này được xây dựng để hỗ trợ cả bốn mô hình.Đây cũng là sự khác biệt chính đối với các DBMS đa mô hình khác khi chúng triển khai một lớp bổ sung với một API,sao chép các mô hình bổ sung,tuy nhiên chúng cũng chỉ là một mô hình hạn chế về tốc độ và khả năng mở rộng.
- **Mô hình tài liệu(The Document Model)** Dữ liệu trong mô hình này được lưu trữ bên trong Document.Document là tập hợp các cặp Key/Value(còn được gọi là trường hoặc thuộc tính),trong đó khóa cho phép truy cập vào giá trị.Giá trị có thể chứa kiểu dữ liệu nguyên thủy.Các Document thường không bắt buộc phải có lược đồ,có thể thuận lợi vì linh hoạt và dễ sửa đổi.Document được lưu trữ trong bộ sưu tập cho phép các nhà phát triển nhóm dữ liệu khi họ quyết định.OrientDB sử dụng classes và clusters như các bộ sưu tập của nó để nhóm các Document.Mô hình Document của OrientDB có thêm khái niệm “LINK” làm mối quan hệ giữa các tài liệu.Với OrientDB bạn có thể quyết định xem có nên nhúng hoặc liên kết trực tiếp tài liệu hay không.Khi bạn tìm một tài liệu thì tất cả các liên kết sẽ được OrientDB tự động giải quyết.Đây là sự khác biệt chính đối với các CSDL Document khác,như MongoDB hoặc CouchDB nơi mà các nhà phát triển phải xử lý mọi mối quan hệ giữa chính các tài liệu đó.
- **Mô hình dữ liệu đồ thị(The Graph Model)** Một biểu đồ đại diện cho một cấu trúc giống như mạng bao gồm các đỉnh(được gọi là nút) được kết nối bởi các cạnh(được gọi là cung).Mô hình đồ thị của OrientDB được biểu diễn bằng khái niệm của một đồ thị thuộc tính,nó được xác định nhờ vào đỉnh và cạnh.Ngoài các thuộc tính bắt buộc,mỗi đỉnh hoặc cạnh cũng có thể chứa một tập hợp các thuộc tính tùy chỉnh,các thuộc tính này có thể được xác định bởi người dùng,có thể làm cho các đỉnh và cạnh xuất hiện tương tự như các Documents.

- **Mô hình Key/Value** Đây là mô hình đơn giản nhất. Tất cả mọi thứ trong CSDL có thể tìm được bằng một khóa, trong đó các giá trị có thể là các loại đơn giản và phức tạp. OrientDB hỗ trợ Documents và Graph Elements làm cho mô hình phong phú hơn so với những gì thường thấy trong mô hình Key/Value cổ điển. Mẫu Key/Value cổ điển cung cấp “buckets” để nhóm các cặp key/value trong các containers khác nhau.
- **Mô hình Đối tượng** Đây là mô hình đã được thừa hưởng bởi lập trình hướng đối tượng và hỗ trợ thừa kế giữa các kiểu, đa hình khi bạn tham chiếu đến một lớp cơ sở ràng buộc trực tiếp from/to đối tượng được sử dụng trong ngôn ngữ lập trình.

2.0.1 Các khái niệm cơ bản

- Bản ghi(Record) là đơn vị nhỏ nhất mà bạn có thể tải và lưu trữ trong CSDL. Records có bốn loại:
 - Document(tài liệu)
 - RecordBytes
 - Vertex(đỉnh)
 - Edge(cạnh)
- Bản ghi là đơn vị nhỏ nhất có thể được tải và được lưu trữ vào CSDL. Một bản ghi có thể là một Document, một bản ghi RecordBytes(BLOB) thậm chí là một đỉnh hoặc một cạnh.
- Document là loại bản ghi linh hoạt nhất có sẵn trong OrientDB. Các tài liệu được gõ nhẹ nhàng và được định nghĩa bởi các lớp lược đồ với các ràng buộc đã định nghĩa, nhưng bạn cũng có thể sử dụng chúng trong một chế độ ít hơn. Xử lý các trường một cách linh hoạt. Bạn có thể dễ dàng nhập và xuất chúng theo định dạng JSON.

```
{
  "name" : "Jay",
  "surname" : "Miser",
  "job" : "Developer",
  "creations" : [
    {
      "name" : "Amiga 1000",
      "company" : "Commodore Inc."
    },
    {
      "name" : "Amiga 500",
      "company" : "Commodore Inc."
    }
  ]
}
```

HÌNH 2.1. Định dạng JSON

- Đối với Documents, OrientDB cũng hỗ trợ các mối quan hệ phức tạp. Từ quan điểm của các nhà phát triển, điều này có thể được hiểu là Map<String, Object>.
- BLOB ngoài kiểu bản ghi Document OrientDB cũng có thể tải và lưu trữ dữ liệu nhị phân. Kiểu bản ghi BLOB được gọi là RecordBytes trước phiên bản OrientDB 2.2

- Vertex: trong CSDL đồ thị, đơn vị dữ liệu cơ bản nhất là node, mà trong OrientDB gọi là đỉnh. Đỉnh lưu trữ thông tin cho CSDL. Có một loại bản ghi riêng biệt được gọi là Edge kết nối một đỉnh khác. Đỉnh cũng là Documents điều này nghĩa là chúng có thể chứa các bản ghi được nhúng và các thuộc tính tùy ý.
- Edge: trong CSDL đồ thị, một vòng cung là kết nối giữa hai nút, mà trong OrientDB gọi là cạnh. Các cạnh là hai chiều và chỉ có thể kết nối hai đỉnh. Edge thông thường lưu dưới dạng tài liệu.
- Record ID : khi OrientDB tạo ra một bản ghi, nó tự gán một định danh đơn vị duy nhất, được gọi là ID bản ghi hoặc RID. Cú pháp cho ID bản ghi là ký hiệu cho bảng định danh cụm vị trí.

2.0.2 Đồ thị nhất quán

Trước OrientDB v2.1.7 độ nhất quán của biểu đồ chỉ có thể được đảm bảo bằng cách sử dụng các giao dịch. Các vấn đề với việc sử dụng các giao dịch cho các hoạt động đơn giản như tạo các cạnh là:

- Speed giao dịch có chi phí so với các hoạt động phi giao dịch.
- Quản lý việc thử lại ở cấp ứng dụng. Hơn nữa, với các kết nối từ xa có nghĩa là độ trễ cao.
- Khả năng mở rộng thấp trên đồng thời cao.

Theo v2.1.7, OrientDB cung cấp một chế độ mới để quản lý các đồ thị mà không cần sử dụng các giao dịch. Nó sử dụng lớp Java OrientGraphNoTx hoặc thông qua SQL bằng cách thay đổi thiết lập toàn cục `sql.graphConsistencyMode`.

Cả 2 chế độ mới `notx_sync_repair` và `notx_async_repair` sẽ tự động quản lý xung đột, với RETRY có cấu hình (mặc định = 50). Trong trường hợp các thay đổi đối với biểu đồ xảy ra đồng thời thì xung đột sẽ được OrientDB thu thập và được lặp lại. Các hoạt động hỗ trợ tự động thử lại là:

- CREATE EDGE
- DELETE EDGE
- DELETE VERTEX

2.0.2.1 Cách sử dụng

Để sử dụng các chế độ nhất quán không sử dụng giao dịch, hãy đặt thiết lập toàn cầu `sql.graphConsistencyMode` thành `notx_sync_repair` hoặc `notx_async_repair` trong lệnh OrientDB `bin/server.sh` hoặc trong tệp `config/orientdb-server-config.xml` trong phần thuộc tính.

```
...  
<properties>  
...  
  <entry name="sql.graphConsistencyMode" value="notx_sync_repair"/>  
...  
</properties>
```

HÌNH 2.2. Tệp cấu hình OrientDB

Chương 3

API và Driver

3.1 Hàm

Hàm cho phép chúng ta tự định nghĩa một đoạn mã thực thi có hoặc không có tham số từ database hoặc query và trả về một tập hợp (set) kết quả.

Trong một vài trường hợp chúng ta cần một hiệu suất cao hơn hoặc một tính năng nào đó không có sẵn trong OrientDB. Cơ sở dữ liệu quan hệ giải quyết vấn đề này bằng phương pháp Stored Procedures, cho phép người dùng định nghĩa các đoạn mã thực thi trong ngữ ngôn ngữ lập trình dành riêng cho nhà cung cấp. Trong khi đó, OrientDB giải quyết vấn đề này bằng hàm.

Hàm trong OrientDB là một đoạn mã thực thi. Chúng là một mô hình của Functional Programming (lập trình hàm) để phát triển tùy chỉnh các tính năng thích hợp, tối ưu hơn đối với ứng dụng của bạn. Các tính chất của hàm trong OrientDB:

- **Tính ổn định - Persistent:** Dữ liệu được lưu trữ trên database và có thể gọi trên bất kỳ client nào.
- **Hỗ trợ đa ngôn ngữ - Multiple Language Support:** Hàm hỗ trợ đa ngôn ngữ. Hiện tại, chúng ta có thể viết trên OrientDB SQL, JavaScript. Hơn thế nữa chúng còn hỗ trợ cho các ngôn ngữ như Ruby, Scala, Java và một số ngôn ngữ khác.
- **Hỗ trợ đa thực thi - Multiple Execution Support:** OrientDb thực thi hàm thông qua SQL, Java, REST và Studio.
- **Đệ quy - Recursion**
- **Mapping:** Hàm sẽ tự động ánh xạ tham số thông qua vị trí và tên.
- **Khả năng mở rộng - Extensibility:** OrientDB Plugins có thể chèn một đối tượng mới vào hàm để sử dụng.

3.1.1 Tạo hàm trong OrientDB

OrientDb cung cấp một số hàm mặc định. Trong một số trường hợp cụ thể, hàm mặc định sẽ không đáp ứng đủ nhu cầu của người dùng, do đó chúng ta cần phải tạo thêm những hàm mới bằng cách sử dụng OrientDB SQL và JavaScript. Chúng ta có thể thực thi chúng như SQL, HTTP hoặc Java.

3.1.1.1 Tạo hàm

Khi chúng ta tạo một hàm mới trong database, OrientDB sẽ lưu chúng thông qua lớp OFunction. Chúng ta có thể query bản khi đã được lưu trong lớp OFunction. Lớp OFunction có những tham số sau:

BẢNG 3.1. Thuộc tính hàm

Thuộc tính	Mô tả
Tên	Định nghĩa tên hàm.
Code	Định nghĩa đoạn mã thực thi.
Tham số	Định nghĩa một tùy chọn EMBEDDEDLIST của một chuỗi, bao gồm tham số tên nếu có.

3.1.1.2 Sử dụng OrientDb Studio

OrientDB Studio là một giao diện thân thiện không chỉ với những người mới làm quen với OrientDb mà còn với những lập trình viên có kinh nghiệm trong lĩnh vực này bởi sự thuận tiện và dễ thao tác mà phần mềm này mang lại. Việc sử dụng Studio, chúng ta có thể tạo và quản lý **hàm** một cách dễ dàng hơn.

Khi chúng ta đăng nhập vào database trên Studio. Một loạt các liên kết chạy dọc theo đầu trang. Tab hàm là một trang mà chúng ta có thể tạo và quản lý các hàm trong database. Hãy tưởng tượng một tình huống mà bạn cần phải tính toán giai thừa một cách thường xuyên. Trong khi bạn có thể kéo thông tin này vào ứng dụng của bạn, nó sẽ tiết kiệm thời gian và lưu lượng mạng nếu bạn có thể có OrientDB xử lý toán học và trả về kết quả. Hàm OrientDB hỗ trợ đệ quy, vì vậy đây là một quá trình khá đơn giản.

Để quản lý điều này, bạn sẽ điều hướng đến tab Hàm và tạo một hàm mới, đặt tên nó là giai thừa hoặc bất kỳ tên nào bạn thấy phù hợp nhất. Hàm này được viết bằng JavaScript và lấy một đối số, được gọi là num. Sau đó, cung cấp mã sau:

```
if (num === 0)
return 1;
else
return num * factorial(num - 1);
```

Khi hoàn thành hãy save và để database cập nhật một hàm mới. Như hình ??, OrientDb hỗ trợ đệ quy. Khi factorial() được gọi, chúng sẽ tự tính và trả về kết quả.

Theueu hình

3.1.1.3 Sử dụng OrientDB SQL

Trong trường hợp bạn thích làm việc với môi trường Shell, OrientDb cung cấp lệnh SQL CREATE FUNCTION. Ví dụ, chúng ta cần tạo một hàm factorial()

```
orientdb> CREATE FUNCTION factorial "if (num === 0)
return 1;
else return num * factorial(num - 1)"
PARAMETERS [num]
LANGUAGE javascript
```

3.1.2 Sử dụng hàm trong OrientDB

3.1.2.1 Gọi hàm trong SQL

Việc gọi các hàm để thực thi giống như gọi các hàm SQL tiêu chuẩn mà OrientDB mặc định. Ví dụ, hãy xem xét ví dụ trước của hàm factorial ().

```
orientdb> SELECT factorial(5,3)
```

Bên cạnh việc nhập giá trị để hàm tự tính bạn cũng có thể lấy giá trị từ database và truyền vào hàm như một tham số. Hãy xem ví dụ sau:

```
orientdb> SELECT sum(salary, bonus) AS total FROM
Employee
```

3.1.2.2 Gọi hàm từ Java

Khi đang làm việc với Java chúng ta có thể sử dụng hàm từ OrientDB, trong cả việc tạo hàm trong ứng dụng và làm việc với nội dung từ database. Để sử dụng hàm từ database, chúng ta làm những bước sau (Hãy chắc rằng hàm đã được tạo trong database):

1. Tạo liên kết với bộ quản lý hàm.
2. Lấy hàm mà bạn muốn sử dụng.
3. Thực thi hàm.

Ví dụ, chúng ta tìm kiếm hàm factorial(). Việc đầu tiên là tạo kết nối với database và tìm kiếm hàm thông qua lớp OFunction.

```
// open Database
ODatabaseDocument db = new
    ODatabaseDocumentTx("plocal:/tmp/db");
db.open("admin", "admin");

// Retrieve Function from Database.
OFunction factorial =
    db.getMetadata().getFunctionLibrary().getFunction("factorial");

// Use Factorial Function
Number result = factorial.execute(24);
```

Một cách tiếp cận khác trong việc tìm kiếm hàm từ OrientDB bạn có thể định nghĩa một đối số và truyền vào hàm hoặc mapping từ tên đối số đến giá trị trong một map.

```
public void reportFactorials(List<Number> inputValues) {
    ODatabaseDocumentTx db = new
        ODatabaseDocumentTx("plocal:/tmp/db");
    db.open("admin", "admin");
    OFunction factorial =
        db.getMetadata().getFunctionLibrary().getFunction("factorial");

    for (int i = 0; i < inputValues.size(); i++) {
        Number number = list.get(i);
        Map<String, Object> params = new
            HashMap<String, Object>();
        params.put("num", number);
        Number result = factorial(number);
```

```

        System.out.println(String.format("Factorial of
            %d: %d", number, result));
    }
}

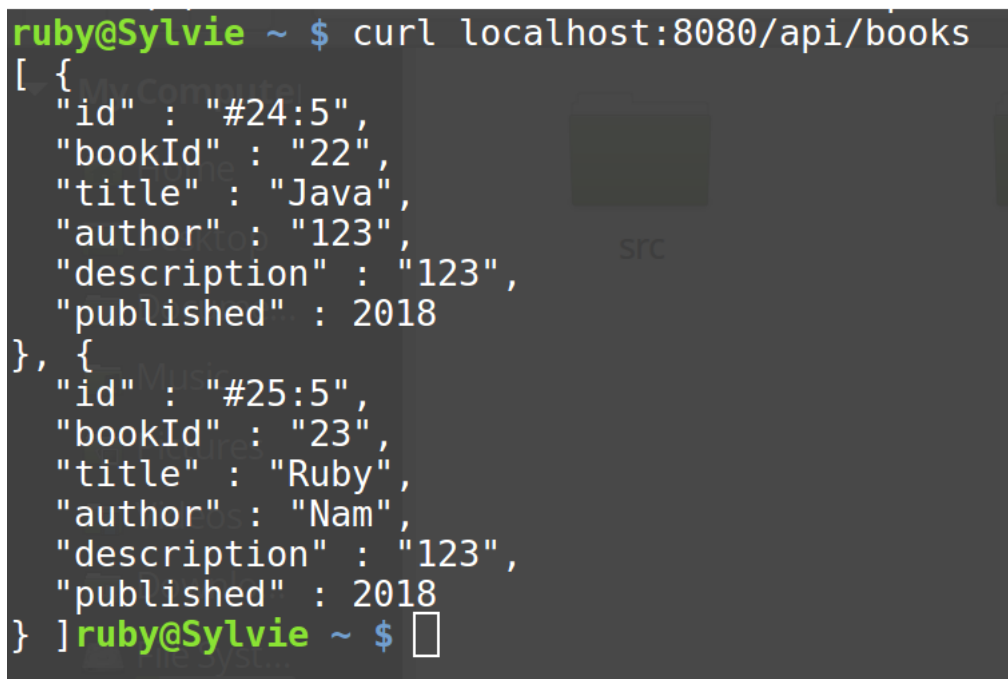
```

3.1.2.3 Gọi hàm từ HTTP REST API

Hàm trong database có thể được truy cập thông qua REST service. Nó nhận tham số thông qua các request payload (Json) từ URL. Trong ví dụ sau, chúng tôi lấy ra thông tin danh sách những quyển sách có trong databasr thông qua URL sau:

```
>curl localhost:8080/api/books
```

HÌNH 3.1. Lấy ra danh sách API



```

ruby@Sylvie ~ $ curl localhost:8080/api/books
[ {
  "id" : "#24:5",
  "bookId" : "22",
  "title" : "Java",
  "author" : "123",
  "description" : "123",
  "published" : 2018
}, {
  "id" : "#25:5",
  "bookId" : "23",
  "title" : "Ruby",
  "author" : "Nam",
  "description" : "123",
  "published" : 2018
} ]
ruby@Sylvie ~ $

```

Nếu OrientDB REST API trả về HTTP 202 OK, có nghĩa là nội dung bạn đang tìm đã có và được trả về. Chú ý rằng bạn có thể gọi một hàm không thay đổi giá trị bằng việc sử dụng phương thức HTTP GET. Nếu một hàm có giá trị thay đổi, chúng ta cần sử dụng phương thức HTTP POST. Để tạo một request HTTP POST, HTTP header phải có định dạng *"Content-Type: application/json"*.

3.1.2.4 Giá trị trả về của HTTP

Khi gọi một hàm thông qua REST service, OrientDB trả về kết quả ở định dạng JSON. Chúng ta có thể tự định nghĩa định dạng giá trị trả về trong hàm như sau:

- Ví dụ hàm trả về kết quả là số.

```
return 31;
```

- Kết quả trả về:

```
{"result": [{"@type": "d", "@version": 0, "value": 31}]}
```

- Hàm trả về đối tượng trong JavaScript.

```
return {"a": 1, "b": "foo"}
```

- Giá trị trả về:

```
{"result": [{"@type": "d", "@version": 0, "value": {"a": 1, "b": "foo"}}]}
```

- Hàm trả về một mảng.

```
return [1, 2, 3]
```

- Giá trị trả về

```
{"result": [{"@type": "d", "@version": 0, "value": [1, 2, 3]}]}
```

- Hàm trả về là một bản ghi sau khi đã query.

```
return db.query("SELECT FROM OUser")
```

- Giá trị trả về.

```
{
  "result": [
    {
      "@type": "d",
      "@rid": "#6:0",
      "@version": 1,
      "@class": "OUser",
      "name": "admin",
      "password": "1234",
      "status": "ACTIVE",
      "roles": [
```

```

        "#4:0"
    ],
    "@fieldTypes": "roles=n"
},
{
    "@type": "d",
    "@rid": "#6:1",
    "@version": 1,
    "@class": "OUser",
    "name": "reader",
    "password": "5678",
    "status": "ACTIVE",
    "roles": [
        "#4:1"
    ],
    "@fieldTypes": "roles=n"
}
]
}

```

3.1.3 Truy cập database từ function

Query là câu lệnh không đổi. Để thực thi query từ hàm, sử dụng phương thức query() như sau:

```
return orient.getDatabase().query("SELECT name FROM OUser");
```

3.1.3.1 Query với tham số ngoài

Tạo một hàm với tên là getUserRoles và trả về một user. Chúng ta làm như sau:

```
return orient.getDatabase().query("SELECT roles FROM OUser
    WHERE name = ?", name );
```

Hàm sẽ tìm kiếm mà tham số *name* như là một giá trị trong JavaScript. Bạn có thể sử dụng nó như là một giá trị trong câu query. OrientDB hỗ trợ SQL và JavaScript.

```
var gdb = orient.getGraph();
var results = gdb.command( "sql", "SELECT FROM Employee WHERE
    company = ?", [ "Orient Technologies" ] );
```

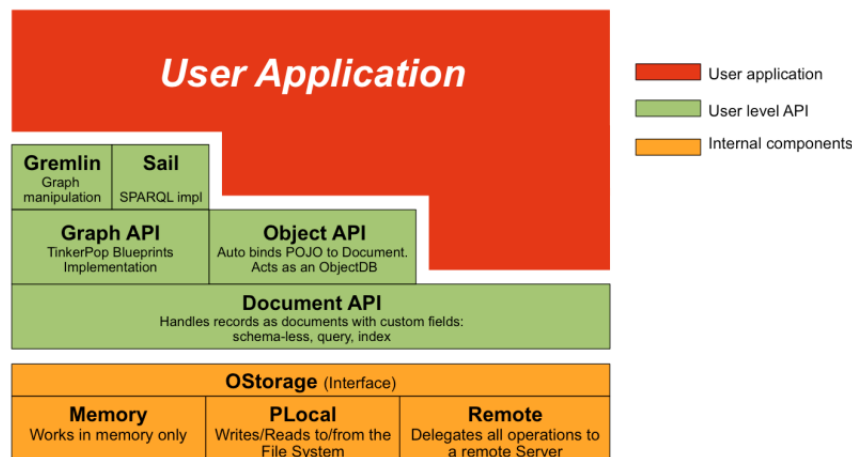
Giá trị trả về là một mảng của đối tượng.

- Trả về đỉnh. Kết quả trả về là một hiện thực của lớp OrientVertex.
- Trả về cạnh. Kết quả trả về là một hiện thực của lớp OrientEdge.
- Trả về một bản ghi. Kết quả trả về là một hiện thực của lớp OIdentifiable (hoặc lớp con của lớp này)

3.2 Java API

OrientDB được viết bằng Java. Điều đó có nghĩa là bạn có thể sử dụng Java API mà không cần phải cài đặt bất cứ thứ gì.

3.2.1 Kiến trúc Component



HÌNH 3.2. Kiến trúc Component

(a) <https://orientdb.com/docs/2.2.x/Java-API.html>

OrientDB cung cấp ba sự chọn lựa khác nhau cho Java API khi chúng ta làm việc với OrientDB

- Graph API: Sử dụng Java API nếu chúng ta làm việc với đồ thị và muốn tối ưu code hơn thông qua việc hiện thực framework TinkerPop Blueprints. Nếu sử dụng TinkerPop ở dạng chuẩn, chúng ta có thể hạn chế thông tin nhiễu.
- Document API: Sử dụng Java API nếu miền dữ liệu của bạn phù hợp với Document Databases với cấu trúc schema-less.
- Object API: Sử dụng Java API nếu chúng ta cần định nghĩa đầy đủ hướng đối tượng, hệ thống sẽ chuyển tất cả các thực thể thành POJO (Plain Old Java Objects). Đây là cách dễ dàng để chuyển Object khi lập trình các ứng dụng JPA.

Mỗi Java API đều có ưu và nhược điểm riêng của nó. Để biết thêm thông tin về việc xác định Java API nào sẽ sử dụng với ứng dụng của bạn, hãy xem ??.

Bản sau đây là sự so sánh giữa các Java API:

BẢNG 3.2. Các Java API

	Graph	Document	Object
API	Graph API	Document API	Object Database
Java class	OrientGraph	ODatabaseDocumentTx	OObjectDatabaseTx
Query	có	có	có
Schema Less	có	có	có
Schema full	có	có	có
Speed	90%	100%	50%

Chú ý 3.1. Tốc độ ở đây có nghĩa là quá trình tạo và vận hành *CRUD* (create, read, update và delete) như là những câu query, insert, update và delete. Tốc độ này được tính dựa vào kích thước tập dữ liệu đầu vào rất lớn và 100% là tốc độ nhanh nhất có thể đạt được.

Chú ý 3.2. Object Database cung cấp ở mức độ high-level of abstraction với dung lượng code thấp nhưng dễ phát triển và duy trì.

3.2.2 Graph API

Với Java API bạn có thể sử dụng OrientDB như là một Graph Database, cho phép chúng ta làm việc với cạnh và đỉnh. Graph API tuân thủ theo các chuẩn của TinkerPop. TinkerPop cung cấp đầy đủ một stack để làm việc với Graph Database.

- Blueprints cung cấp các interface và hiện thực thông dụng và đầy đủ các cấu trúc dữ liệu. Tóm lại, Blueprint cung cấp một store duy nhất cho các giao diện được triển khai để giúp các nhà phát triển tạo ra phần mềm mà không bị ràng buộc với các hệ thống quản lý dữ liệu cụ thể.
- Gremlin cung cấp một đồ thị dựa vào Turing-complete được thiết kế theo các cặp key/value trong biểu đồ đa quan hệ.

3.2.2.1 Với Blueprints

OrientDb hỗ trợ ba loại lưu trữ khác nhau, dựa vào Database URL được sử dụng:

- **Persistent Embedded Graph Database:** Liên kết đến những ứng dụng như là JAR (hoạt động trong môi trường không cần mạng). Sử dụng từ *plocal* đứng trước URL. Ví dụ:

```
plocal:/tmp/graph/demo
```

- **In-Memory Embedded Graph Database:** Giữ tất cả dữ liệu trên thanh RAM. Sử dụng từ khóa *memory* đứng trước URL. Ví dụ:

```
memory:/tmp/graph/demo
```

- **Persistent Remote Graph Database:** Sử dụng tính hiệu nhị phân để gửi và nhận dữ liệu thông qua việc điều khiển từ xa một server Orient. Sử dụng từ khóa *remote* đứng trước URL. Ví dụ

```
remote:172.17.0.2:2424/book
```

Đối với việc sử dụng Graph API, bạn cần phải hiện thực lớp `OrientGraph`. Cấu trúc nhận một Database URL diễn ra như sau: Nếu database đó tồn tại, Graph API sẽ thực hiện lệnh `open`. Nếu nó không tồn tại, Graph API sẽ thực hiện lệnh `create`.

Chú ý 3.3. *Khi thực hiện việc tạo một database bằng Graph API, bạn chỉ có thể tạo 2 loại plocal hoặc memoryoại database. Đối với Persistent Remote Graph Database sẽ không tạo được.*

Khi chạy một ứng dụng đa luồng, sử dụng một hiện thực của `OrientGraph` cho mỗi luồng. Hãy nhớ rằng tất cả các thành phần đồ thị, chẳng hạn như đỉnh và cạnh, không phải là luồng an toàn. Vì vậy, chia sẻ chúng giữa các luồng có thể dẫn đến kết quả không thể đoán trước.

Để ngắt một hiện thực của graph, dùng lệnh `shutdown()`. Ví dụ:

```
OrientGraph graph = new OrientGraph("plocal:/temp/graph/db");
try {
    ...
} finally {
    graph.shutdown();
}
```

3.2.2.2 Transactions -dịch giao dịch

OrientDB hỗ trợ giao dịch. Có nghĩa là nó không khóa những giao dịch đang chạy, nhưng ở mỗi lần commit mỗi phiên bản của graph đều kiểm tra xem liệu có hay không một bản cập nhập của một client nào đó hay không.

Vì lý do này, hãy viết mã của bạn để xử lý trường hợp cập nhật đồng thời:

```
for (int retry = 0; retry < maxRetries; ++retry) {
    try {
        // LOOKUP FOR THE INVOICE VERTEX
```

```

Iterable<Vertex> invoices =
    graph.getVertices("invoiceId", 2323);
Vertex invoice = invoices.iterator().next();

// CREATE A NEW ITEM
Vertex invoiceItem =
    graph.addVertex("class:InvoiceItem");
invoiceItem.field("price", 1000);
// ADD IT TO THE INVOICE
invoice.addEdge(invoiceItem);
graph.commit();

// OK, EXIT FROM RETRY LOOP
break;
} catch( ONeedRetryException e ) {
// SOMEONE HAVE UPDATE THE INVOICE VERTEX AT THE SAME
// TIME, RETRY IT
}
}

```

3.2.2.3 Sử dụng Non-Transactional Graphs

Trong trường hợp dữ liệu nhập vào quá lớn, chuẩn transaction OrientGraph trở nên chậm chạp. Để cải thiện tốc độ xử lý chúng ta có thể dùng non-Transactional thông qua lớp OrientGraphNoTx.

Với non-Transactional, mỗi hoạt động là một node và nó cập nhật dữ liệu trên mỗi hoạt động. Khi phương thức trả về, các bản cập nhật cơ bản đã được lưu trữ. Phù hợp cho lượng lớn dữ liệu nhập vào thông qua quá trình non-Transactional.

Chú ý 3.4. Việc sử dụng non-transaction graph có thể bị lạm dụng trong trường hợp thay đổi được thực hiện từ nhiều nguồn cùng một lúc. Sử dụng non-transaction chỉ nên thực hiện cho các hoạt động đơn luồng.

3.2.2.4 Cấu hình đồ thị

Những tham số để cấu hình cho Orient Graph như sau:

- **blueprints.orientdb.url:** Xác định database url.
- **blueprints.orientdb.password:** Xác định mật khẩu.

- **blueprints.orientdb.saveOriginalIds:** Xác định xem nó có lưu ID phần tử ban đầu hay không bằng cách sử dụng thuộc tính id. Bạn có thể thấy điều này hữu ích khi nhập biểu đồ để giữ nguyên ID gốc.
- **blueprints.orientdb.keepInMemoryReferences:** Xác định xem chương trình có ghi dữ liệu lên bộ nhớ hay không bằng cách sử dụng Record ID (@rid)
- **blueprints.orientdb.useCustomClassesForEdges:** Xác định xem chương trình có sử dụng cạnh hay không, nếu không nó sẽ tạo mới.
- **blueprints.orientdb.useCustomClassesForVertex:** Xác định xem chương trình có sử dụng đỉnh hay không, nếu không nó sẽ tạo mới.
- **blueprints.orientdb.useVertexFieldsForEdgeLabels:** Xác định xem mối quan hệ giữa các đỉnh thông qua lớp cạnh. Điều này giúp bạn traversals bằng cạnh nhanh hơn.
- **blueprints.orientdb.lightweightEdges:** Xác định xem chúng ta có sử dụng cạnh có trọng số yếu hay không. Điều này giúp bạn tránh tạo một document trên cạnh. Document chỉ được tạo khi cạnh có thuộc tính.
- **blueprints.orientdb.autoStartTx:** Xác định xem nó có tự động bắt đầu một giao dịch hay không khi đồ thị thay đổi (thêm hoặc xóa cạnh, đỉnh, thuộc tính).

3.2.3 Cạnh và đỉnh trong đồ thị

3.2.3.1 Đỉnh

Để tạo mới một đỉnh trong Graph Database chúng ta chỉ cần gọi phương thức *Vertex OrientGraph.addVertex(Object id)*.

Chú ý 3.5. Chúng ta sẽ bỏ qua tham số id, việc triển khai một tham số ID sẽ do OrientDB tự gán khi nó tạo ra một đỉnh mới. Để lấy ra tham số ID duy nhất, sử dụng phương thức *Vertex.getId()* trên đối tượng.

Ví dụ:

```
Vertex v = graph.addVertex(null);
System.out.println("Created vertex: " + v.getId());
```

3.2.3.2 Truy xuất đến đỉnh

Để truy xuất đến đỉnh trong một đối tượng, sử dụng phương thức *getVertices()*. Ví dụ:

```
for (Vertex v : graph.getVertices()) {  
    System.out.println(v.getProperty("name"));  
}
```

Để tìm kiếm các đỉnh bằng khóa (key), gọi phương thức *getVertices()* thông qua trường tên và giá trị để hệ thống thực hiện so khớp. Ví dụ:

```
for( Vertex v : graph.getVertices("Account.id", "23876JS2") ) {  
    System.out.println("Found vertex: " + v );  
}
```

3.2.3.3 Xóa đỉnh

Để xóa đỉnh trong Graph Database gọi phương thức *OrientGraph.removeVertex(Vertex vertex)*. Đầu tiên chúng sẽ ngắt kết nối đỉnh đó trong đồ thị, sau đó thực hiện xóa. Việc ngắt kết nối đồng nghĩa với xóa tất cả các cạnh thuộc đỉnh đã được chỉ định xóa. Ví dụ:

```
graph.removeVertex(luca);
```

3.2.3.4 Cạnh

Cạnh là một liên kết giữa hai đỉnh trong đồ thị. Để tạo một cạnh mới trong đồ thị, gọi hàm *Edge OrientGraph.addEdge(Object id, Vertex outVertex, Vertex inVertex, String label)*

Chú ý 3.6. Hệ thống trong *OrientDB* sẽ bỏ qua tham số *id*, nó sẽ gán một giá trị *ID* duy nhất do nó cung cấp khi nó tạo mới một cạnh. Để truy cập *ID* này, sử dụng hàm *Edge.getId()*. *OutVertex* đề cập đến đỉnh bắt đầu và *inVertex* đề cập đến đỉnh kết thúc. Nhãn cho biết tên cạnh. Mặc định là *null* nếu cạnh không được gán nhãn.

Ví dụ:

```
Vertex luca = graph.addVertex(null);  
luca.setProperty("name", "Luca");  
  
Vertex marko = graph.addVertex(null);  
marko.setProperty("name", "Marko");  
  
Edge lucaKnowsMarko = graph.addEdge(null, luca, marko, "knows");  
System.out.println("Created edge: " + lucaKnowsMarko.getId());
```

3.2.3.5 Truy xuất đến cạnh

Để truy xuất đến cạnh dùng phương thức *getEdges()*. Ví dụ:

```
for (Edge e : graph.getEdges()) {
    System.out.println(e.getProperty("age"));
}
```

Khi sử dụng cạnh có có trọng số yếu - Lightweight Edges, OrientDB sẽ lưu cạnh đó như là một Link hơn là một bản ghi. Điều này cải thiện hiệu suất. Mặc khác, phương thức *getEdges()* chỉ truy xuất duy nhất đến các bản ghi thuộc lớp super class E. Khi sử dụng Lightweight Edges, OrientDB chỉ tạo các bản ghi trong lớp E trong một số trường hợp nhất định, chẳng hạn như khi cạnh có các thuộc tính. Nếu không, các cạnh tồn tại dưới dạng các liên kết trên các đỉnh vào và ra. Nếu bạn muốn sử dụng *getEdges()* để trả về tất cả các cạnh nhưng có tính năng Lightweight Edges, chúng ta làm như sau:

```
orientdb> ALTER DATABASE my_db useLightweightEdges=FALSE
```

3.2.3.6 Xóa cạnh

Để xóa cạnh trong đồ thị, gọi hàm *OrientGraph.removeEdge(Edge edge)*. Nó xóa cạnh kết nối hai đỉnh. Ví dụ:

```
graph.removeEdge(lucaKnowsMarko);
```

3.2.3.7 Thuộc tính của cạnh và đỉnh

Cạnh và đỉnh có thể có nhiều thuộc tính. Khóa của thuộc tính là kiểu String, và giá trị có kiểu mà OrientDB hỗ trợ.

BẢNG 3.3. Phương thức chung của cạnh và đỉnh

Phương thức	Mô tả
setProperty(String key, Object value)	Cài đặt thuộc tính
Object getProperty(String key)	truy xuất đến thuộc tính
void removeProperty(String key)	xóa thuộc tính

Ví dụ:

```
vertex2.setProperty("x", 30.0f);
vertex2.setProperty("y", ((float) vertex1.getProperty("y")) /
    2);

for (String property : vertex2.getPropertyKeys()) {
```

```

        System.out.println("Property: " + property + "=" +
            vertex2.getProperty(property));
    }

    vertex1.removeProperty("y");

```

3.2.3.8 Cài đặt đa thuộc tính

OrientDB là hiện thực của Blueprints nên hỗ trợ cài đặt đa thuộc tính trên một dòng lệnh bằng cách sử dụng phương thức `setProperty(Object ...)`. Mục đích nhằm cải thiện hiệu suất, giúp bạn tránh việc lưu đi lưu lại một phần tử mỗi khi cập nhật một trong những thuộc tính của nó. Ví dụ:

```

vertex.setProperty("name", "Jill", "age", 33, "city",
    "Rome", "born", "Victoria, TX");

```

Bạn cũng có thể thông qua một map của những giá trị như là đối số. Trong trường hợp này, tất cả những giá trị trong cùng một map sẽ được đặt cùng một thuộc tính. Ví dụ:

```

Map<String, Object> props = new HashMap<String, Object>();
props.put("name", "Jill");
props.put("age", 33);
props.put("city", "Rome");
props.put("born", "Victoria, TX");
vertex.setProperty(props);

```

3.2.4 Graph Schema

Trong OrientDB chúng ta có thể sử dụng Graph Database ở chế độ *schema-less* (lược đồ giản lược), hoặc bạn có thể thực thi mô hình dữ liệu chặt chẽ thông qua một lược đồ. Khi sử dụng lược đồ, chúng ta có thể sử dụng lược đồ trên tất cả dữ liệu hoặc chỉ xác định ràng buộc trên các trường nhất định, cho phép người dùng thêm trường tùy chỉnh vào bản ghi.

Chế độ lược đồ bạn sử dụng được định nghĩa ở cấp lớp. Vì vậy, ví dụ bạn có thể có lớp `Book` ở chế độ lược đồ đầy đủ và `BookInformation` ở chế độ lược đồ giản lược.

Trong OrientDB lược đồ ở ba dạng sau:

- **Lược đồ đầy đủ - Schema-Full:** tất cả các thuộc tính đều ở dạng *not null*.
- **Lược đồ giản lược - Schema-Less:** Tạo các lớp không có thuộc tính. Theo mặc định, đây là chế độ không nghiêm ngặt, cho phép các bản ghi có các trường tùy ý.

- **Lượt đồ lai - Schema-Hybrid:** Tạo các lớp và định nghĩa một số trường, trong khi để lại các bản ghi để xác định các trường tùy chỉnh.

Để truy cập vào lượt đồ, chúng ta có thể dùng SQL hoặc API.

3.2.4.1 Lớp trong Graph Database

Lớp là một khái niệm được rút ra từ mô hình Object-Oriented Programming. OrientDb định nghĩa lớp là một bản ghi. Trong mô hình sơ đồ quan hệ, nó gần với khái niệm bản. Lớp có thể là lượt đồ giản lược, lượt đồ đầy đủ, hoặc dạng lai. Lớp có thể kế thừa từ một lớp khác, hoặc là một node của cây gồm tập hợp nhiều lớp. Trong trường hợp kế thừa, lớp con kế thừa lớp cha, kế thừa tất cả các thuộc tính của lớp cha.

Lớp có thể là một cluster và được định nghĩa như một cluster, nhưng nó có thể hỗ trợ các cluster khác. Trong trường hợp mặc định, lần đọc xảy ra trên một cụm xác định. Khi chúng ta tạo ra một lớp mới, nó tạo ra một cluster vật lý có cùng tên với tên lớp, được đổi thành chữ thường.

Trong OrientDB, v là lớp cơ sở cho các đỉnh, e là lớp cơ sở cho các cạnh. OrientDB xây dựng lớp tự động khi bạn tạo graph database.

3.2.4.2 Làm việc với lớp đỉnh và cạnh

Trong khi bạn có thể xây dựng các đồ thị bằng cách sử dụng các cá thể lớp V và E, bạn nên tạo các kiểu tùy chỉnh cho các đỉnh và các cạnh.

Để tạo tùy chỉnh lớp đỉnh sử dụng hàm *createVertexType(<name>)*. Ví dụ:

```
// Create Graph Database Instance
OrientGraph graph = new OrientGraph("plocal:/tmp/db");

// Create Custom Vertex Class
OrientVertexType account = graph.createVertexType("Account");
```

Trong Blueprints, các cạnh có khái niệm nhãn được sử dụng để phân biệt giữa các loại cạnh. OrientDB liên kết khái niệm nhãn cạnh với các lớp cạnh. Có một phương pháp tương tự trong việc tạo các kiểu cạnh tùy chỉnh, sử dụng *createEdgeType(<name>)*:

```
// Create Graph Database Instance
OrientGraph graph = new OrientGraph("plocal:/tmp/db");

// Create Custom Vertex Classes
```

```
OrientVertexType accountVertex =
    graph.createVertexType("Account");
OrientVertexType addressVertex =
    graph.createVertexType("Address");

// Create Custom Edge Class
OrientEdgeType livesedge = graph.createEdgeType("Lives");

// Create Vertices
Vertex account = graph.addVertex("class:Account");
Vertex address = graph.addVertex("class:Address");

// Create Edge
Edge e = account.addEdge("Lives", address);
```

3.2.4.3 Hiện thực cây

Hiện thực cây là tạo ra một hay nhiều lớp kế thừa hai lớp V và E. Ví dụ:

```
graph.createVertexType(<class>, <super-class>); // Vertex
graph.createEdgeType(<class>, <super-class>); // Edge
```

3.2.4.4 Truy xuất loại

Lớp có tính đa hình. Nếu bạn tìm kiếm một đỉnh chung, bạn cũng nhận được tất cả các trường hợp đỉnh tùy chỉnh. Để lấy các lớp tùy chỉnh, sử dụng phương thức `getVertexType()` và `getEdgeType`. Ví dụ, lấy ra từ cá thể cơ sở dữ liệu đồ thị:

```
OrientVertexType accountVertex = graph.getVertexType("Author");
OrientEdgeType livesEdge = graph.getEdgeType("Shelf");
```

3.2.4.5 Thuộc tính trong Graph Database

Các trường trong cùng một lớp được gọi là thuộc tính.

3.2.4.6 Làm việc với thuộc tính

Để sử dụng các thuộc tính trong ứng dụng của bạn, trước tiên bạn cần tạo lớp và đặt một cá thể. Một khi bạn đã thiết lập lớp, bạn có thể bắt đầu làm việc với các thuộc tính trên lớp đó.

Tạo thuộc tính

Bạn có thể định nghĩa các thuộc tính cho lớp đó. Ví dụ:

```
// Retrieve Vertex
OrientVertexType accountVertex = graph.getVertexType("Account");
// Create Properties
accountVertex.createProperty("id", OType.INTEGER);
accountVertex.createProperty("birthDate", OType.DATE);
```

Xóa thuộc tính Để xóa một thuộc tính lớp, sử dụng phương thức `Class.drop Property ()`. Ví dụ,

```
accountVertex.dropProperty("name");
```

Điều này không xóa hết tên thuộc tính. OrientDB không loại bỏ các thuộc tính bị loại bỏ khỏi bản ghi trừ khi bạn xóa chúng một cách rõ ràng bằng cách sử dụng lệnh SQL UPDATE với mệnh đề REMOVE.

```
// Drop the Property
accountVertex.dropProperty("name");

// Remove the Records
database.command(new OCommandSQL("UPDATE Account REMOVE
    name")).execute();
```

3.2.4.7 Các loại ràng buộc trong thuộc tính

OrientDb hỗ trợ một số ràng buộc sau:

BẢNG 3.4. Ràng buộc trong thuộc tính

Ràng buộc	Phương thức	Mô tả
Minimum Value	setMin()	Xác định giá trị tối thiểu. Thuộc tính chấp nhận chuỗi và cũng hoạt động trên phạm vi ngày.
Maximum Value	setMax()	Xác định giá trị tối đa. Thuộc tính chấp nhận chuỗi và cũng hoạt động trên phạm vi ngày.
Mandatory	setMandatory()	Xác định xem thuộc tính phải được chỉ định.
Read Only	setReadonly()	Các thuộc tính chỉ có quyền đọc.
Not Null	setNotNull()	Xác định xem thuộc tính có chấp nhận các giá trị null hay không.
Unique		Xác định thuộc tính là duy nhất.
Regex		Xác định giá trị của trường có thỏa biểu thức chính quy hay không.
Ordered	setOrdered()	Xác định xem danh sách cạnh phải được sắp xếp, đảm bảo rằng Danh sách không được sử dụng thay cho Tập hợp.

Ví dụ:

```
// Create Unique Nickname
profile.createProperty("nick",
    OType.STRING).setMin("3").setMax("30")
    .setMandatory(true).setNotNull(true);
profile.createIndex("nickIdx", OClass.INDEX_TYPE.UNIQUE,
    "nick");

// Create User Properties
profile.createProperty("name",
    OType.STRING).setMin("3").setMax("30");
profile.createProperty("surname",
    OType.STRING).setMin("3").setMax("30");
profile.createProperty("registeredOn",
    OType.DATE).setMin("2010-01-01 00:00:00");
profile.createProperty("lastAccessOn",
    OType.DATE).setMin("2010-01-01 00:00:00");
```

3.2.4.8 Chỉ mục như là một ràng buộc

Để đặt giá trị thuộc tính thành duy nhất, hãy sử dụng chỉ mục UNIQUE như một ràng buộc bằng cách chuyển đối tượng Parameter với kiểu khóa. Ví dụ,

```
graph.createKeyIndex("id", Vertex.class, new Parameter("type",
    "UNIQUE"));
```

OrientDB áp dụng ràng buộc này cho tất cả các trường hợp đỉnh và lớp con. Để chỉ định một chỉ mục dựa vào một kiểu tùy chỉnh, hãy sử dụng tham số lớp.

```
graph.createKeyIndex("name", Vertex.class, new
    Parameter("class", "Member"));
```

Chúng ta có thể định nghĩa chỉ mục duy nhất như sau:

```
graph.createKeyIndex("id", Vertex.class, new Parameter("type",
    "UNIQUE"),
new Parameter("class", "Member"));
```

Sau đó bạn có thể lấy một đỉnh hoặc một cạnh bằng cách thêm tiền tố tên lớp vào trường. Ví dụ, bằng cách sử dụng về Member.name thay cho tên duy nhất, cho phép bạn sử dụng chỉ mục được tạo ra đối với tên được lưu trong lớp Member.

```
for( Vertex v : graph.getVertices("Member.name", "Jay")) {
```

```
        System.out.println("Found vertex: " + v)
    }
```

Nếu tên lớp không được truyền, thì nó sử dụng V cho các đỉnh và E cho các cạnh.

```
graph.getVertices("name", "Jay");
graph.getEdges("age", 20);
```

3.2.5 Document API

Document API là nền tảng triển khai ở mức cao hơn, giống như Object OrientDB và Graph API, Document API có hỗ trợ:

- Multi-thread Access.
- Transactions.
- Queries
- Traverse

Nó cũng rất linh hoạt khi sử dụng schema-full, schema-less và schema-hybrid. Ví dụ:

```
// OPEN THE DATABASE
ODatabaseDocumentTx db = new ODatabaseDocumentTx(
    "remote:localhost/petshop").open("admin", "admin_passwd");

// CREATE A NEW DOCUMENT AND FILL IT
ODocument doc = new ODocument("Person");
doc.field("name", "Luke");
doc.field("surname", "Skywalker");
doc.field("city", new ODocument("City")
    .field("name", "Rome")
    .field("country", "Italy"));

// SAVE THE DOCUMENT
doc.save();

db.close();
```

3.2.5.1 Sử dụng Document Database

Khi ứng dụng của bạn làm việc với tài liệu, điều đầu tiên là bạn cần phải mở database và tạo code hiện thực chúng. Nếu muốn kết nối Document DB từ xa, chúng ta làm như sau:

```
ODatabaseDocumentTx db = new ODatabaseDocumentTx
("remote:localhost/petshop")
.open("admin", "admin_passwd");
```

Khi bạn đã mở cơ sở dữ liệu, bạn có thể tạo, cập nhật và xóa tài liệu cũng như dữ liệu tài liệu truy vấn để sử dụng trong ứng dụng của mình.

3.2.5.2 Quản lý database

Khi chúng ta kết thúc với một cá thể cơ sở dữ liệu, chúng ta phải đóng nó để giải phóng tài nguyên hệ thống mà nó sử dụng. Để đảm bảo điều này, cách phổ biến nhất là dùng lệnh *try/finally* để khóa chúng lại.

```
// Open the /tmp/test Document Database
ODatabaseDocumentTx db = new
    ODatabaseDocumentTx("plocal:/tmp/test");
db.open("admin", "admin");

try {
    // Enter your code here...
} finally {
    db.close()
}
```

Trong phần layout, database sẽ tự động đóng khi kết thúc thực thi code. Nếu chúng ta muốn mở database trong bộ nhớ, sử dụng tiền tố *memory*. Nếu chúng ta muốn kết nối từ xa, sử dụng tiền tố *remote* thay vì sử dụng tiền tố *plocal*

3.2.5.3 Tạo một Document DB mới

Trong trường hợp cơ sở dữ liệu không tồn tại, bạn có thể tạo một cơ sở dữ liệu thông qua Java API. Từ hệ thống tệp cục bộ, sử dụng *plocal*:

```
ODatabaseDocumentTx db = new ODatabaseDocumentTx
    ("plocal:/tmp/database/petshop")
    .create();
```

Khi bạn tạo một db lần đầu tiên, OrientDB tạo ba người dùng và ba vai trò cho bạn

- User:
 - User admin (password "admin") với vai trò admin
 - User reader (password "reader") với vai trò reader
 - User writer (password "writer") với vai trò writer

- Vai trò
 - Vai trò admin - đầy đủ quyền thao tác trên DB.
 - Vai trò reader - chỉ có quyền đọc.
 - Vai trò writer - chỉ có quyền đọc và ghi, không có quyền thao tác với lượt đồ.

Để tạo các cá thể cơ sở dữ liệu trên các máy chủ từ xa, quá trình này phức tạp hơn một chút. Bạn cần người dùng và mật khẩu để truy cập cá thể OrientDB Server từ xa. Theo mặc định, OrientDB tạo người dùng root khi máy chủ khởi động lần đầu tiên. Kiểm tra tập tin trong \$ ORIENTDB_HOME / config / orientdb-server-config.xml, nó cũng cung cấp mật khẩu.

3.2.5.4 Truy xuất Document

Với Java API, chúng ta có thể truy xuất tài liệu thông qua đối tượng hoặc truy xuất đến DB khi sử dụng lớp *ODocument*. Những tính năng truy xuất mà *ODocument* mang lại như sau:

- Tìm kiếm tất cả tài liệu trong một cluster.

```
for (ODocument doc :  
    database.browseCluster("CityCars")) {  
    System.out.println(doc.field("model"));  
}
```

- Tìm kiếm tất cả bản ghi trong cùng một lớp.

```
for (ODocument animal :  
    database.browseClass("Animal")) {  
    System.out.println(animal.field("name"));  
}
```

- Đếm số bản ghi trong cùng lớp.

```
long cars = database.countClass("Cars");
```

- Đếm số bản ghi trong cluster.

```
long cityCars = database.countCluster("CityCar");
```

3.2.5.5 Truy vấn trong Document

OrientDB là cơ sở dữ liệu NoSQL, nhưng nó lại hỗ trợ một phần của SQL. Điều này cho phép nó xử lý các liên kết đến tài liệu và đồ thị. Ví dụ,

```
List<ODocument> result = db.query(
    new OSQLSynchQuery<ODocument>("SELECT FROM Animal WHERE id = 10
    AND NAME LIKE 'G%'")
);
```

3.2.5.6 Truy vấn không đồng bộ

Ngoài các truy vấn SQL chuẩn, OrientDB cũng có hỗ trợ cho các truy vấn không đồng bộ. Ở đây, kết quả không được thu thập và trả về một cách đồng bộ, nhưng thay vào đó nó sử dụng một cuộc gọi lại mỗi khi nó tìm thấy một bản ghi thỏa mãn.

```
database.command(
    new OSQLAsyncQuery<ODocument>("SELECT FROM Animal WHERE name = 'Gipsy'",
    new OCommandResultListener() {
        resultCount = 0;
        @Override
        public boolean result(Object iRecord) {
            resultCount++;
            ODocument doc = (ODocument) iRecord;

            // ENTER YOUR CODE TO WORK WITH DOCUMENT

            return resultCount > 20 ? false: true;
        }

        @Override
        public void end() {}
    })
).execute();
```

Khi OrientDB thực hiện một truy vấn không đồng bộ, nó chỉ cần cấp phát bộ nhớ cho từng callback riêng lẻ khi nó gặp phải chúng. Bạn có thể thấy đây là một lợi ích to lớn trong

trường hợp bạn cần làm việc với các bộ kết quả lớn. Truy vấn không đồng bộ không được thiết kế để thao tác dữ liệu, tránh thực hiện cập nhật hoặc giao dịch bên trong mã.

3.2.5.7 Trường trong Document DB

OrientDB hỗ trợ một cách mạnh mẽ để trích xuất các phần của một trường Document. Điều này áp dụng cho API Java, SQL ở câu điều kiện. Để phân tách từng phần chúng ta cần sử dụng dấu ngoặc vuông.

3.2.5.8 Phân tách chính xác từng phần tử

Đơn phân tử

Ví dụ: Trường tags có kiểu EMBEDDEDSET có giá trị như sau: ['Smart', 'Geek', 'Cool']. tags[0] sẽ trả về 'Smart'. Bên trong dấu ngoặc vuông các item được đặt cách nhau bởi dấu phẩy ",". Chúng ta có thể biểu diễn tags[0,2] sẽ trả về kết quả [Smart, 'Cool']. **Dãy phần tử** Bên trong dấu ngoặc vuông đặt giới hạn dưới và trên, được phân tách bằng dấu "-". Theo các ví dụ tags ở trên, các thẻ biểu thức [1-2] trả về ['Geek', 'Cool'].

3.2.6 Object API

Object API hoạt động dựa vào Document-Database và nó hoạt động như một cơ sở dữ liệu đối tượng: quản lý các đối tượng trực tiếp bởi Java. Nó sử dụng Java Reflection để tạo ra các lớp và Javassist để quản lý việc chuyển đổi từ Object sang Document.

Các đối tượng được ODocument ràng buộc với chúng và tái tạo một cách minh bạch các sửa đổi đối tượng. Nó cũng cho phép tải chậm các trường: chúng sẽ không được tải từ tài liệu cho đến lần truy cập đầu tiên. Để làm như vậy, đối tượng **PHẢI** thực hiện getters và setters vì Proxy Javassist bị ràng buộc với chúng. Trong trường hợp tải đối tượng, chỉnh sửa cập nhật tất cả các trường không được tải sẽ không bị mất. Ví dụ:

```
// OPEN THE DATABASE
OObjectDatabaseTx db = new OObjectDatabaseTx
    ("remote:localhost/petshop").open("admin", "admin");

// REGISTER THE CLASS ONLY ONCE AFTER THE DB IS OPEN/CREATED
db.getEntityManager().registerEntityClasses("foo.domain");

// CREATE A NEW PROXIED OBJECT AND FILL IT
Account account = db.newInstance(Account.class);
account.setName("Luke");
```

```
account.setSurname( "Skywalker" );

City rome = db.newInstance(City.class,"Rome",
    db.newInstance(Country.class,"Italy"));
account.getAddresses().add(new Address("Residence", rome,
    "Piazza Navona, 1"));

db.save( account );

// CREATE A NEW OBJECT AND FILL IT
Account account = new Account();
account.setName( "Luke" );
account.setSurname( "Skywalker" );

City rome = new City("Rome", new Country("Italy"));
account.getAddresses().add(new Address("Residence", rome,
    "Piazza Navona, 1"));

// SAVE THE ACCOUNT: THE DATABASE WILL SERIALIZE THE OBJECT AND
    GIVE THE PROXIED INSTANCE
account = db.save( account );
```

3.2.6.1 Tạo kết nối

Một trong những trường hợp sử dụng phổ biến nhất là sử dụng lại cơ sở dữ liệu để tránh tạo ra nó mỗi lần. Nó cũng là kịch bản điển hình của các ứng dụng Web.

```
// OPEN THE DATABASE
OOobjectDatabaseTx db=
    OOobjectDatabasePool.global().acquire("remote:localhost/petshop",
    "admin", "admin");

// REGISTER THE CLASS ONLY ONCE AFTER THE DB IS OPEN/CREATED
db.getEntityManager().registerEntityTypeClass("org.petshop.domain");

try {
    ...
} finally {
    db.close();
}
```

Phương thức `close()` không đóng database mà nó đóng kết nối giữa ứng dụng với DB.

3.2.6.2 Tính kế thừa

Khi tạo một lớp mới, Orient cũng sẽ tạo lược đồ thừa kế đúng nếu chưa được tạo. Ví dụ:

```
public class Account {
    private String name;
    // getters and setters
}

public class Company extends Account {
    private int employees;
    // getters and setters
}
```

Khi chúng ta lưu đối tượng `Company`, OrientDB sẽ lưu đối tượng như là một Document duy nhất trong một cluster thuộc lớp `Company`, khi bạn cần tìm kiếm `Account` với lệnh:

```
SELECT FROM account
```

Hệ thống sẽ tìm kiếm tất cả các Document của `Account` và `Company` thỏa điều kiện câu truy vấn.

3.2.6.3 Làm việc với POJO

Object Database có thể sử dụng lược đồ giản lược, điều đó có nghĩa là lớp được tạo ra sẽ không có thuộc tính. Ví dụ:

```
ObjectDatabaseTx db = new
    ObjectDatabaseTx("remote:localhost/petshop").open("admin",
        "admin");
db.getEntityManager().registerEntityClass(Person.class);

Person p = db.newInstance(Person.class);
p.setName("Luca");
p.setSurname("Garulli");
p.setCity(new City("Rome", "Italy"));

db.save(p);
db.close();
```

Đây là ví dụ đầu tiên. Trong khi mã này khá rõ ràng và dễ hiểu, xin lưu ý rằng chúng tôi đã không khai báo cấu trúc "Person" trước đó. Tuy nhiên Orient đã có thể nhận ra đối

tượng mới và lưu nó theo cách liên tục. **Tạo mới một đối tượng**

Cách tốt nhất để tạo một đối tượng Java là sử dụng API `OObjectDatabaseTx.newInstance()`:

```
public class Person {
    private String name;
    private String surname;

    public Person(){
    }

    public Person(String name){
        this.name = name;
    }

    public Person(String name, String surname){
        this.name = name;
        this.surname = surname;
    }
    // getters and setters
}

OObjectDatabaseTx db =
    new OObjectDatabaseTx("remote:localhost/petshop")
        .open("admin", "admin");

db.getEntityManager().registerEntityClass(Person.class);

// CREATES A NEW PERSON FROM THE EMPTY CONSTRUCTOR
Person person = db.newInstance(Person.class);
person.setName( "Antoni" );
person.setSurname( "Gaudi" );
db.save( person );

// CREATES A NEW PERSON FROM A PARAMETRIZED CONSTRUCTOR
Person person = db.newInstance(Person.class, "Antoni");
person.setSurname( "Gaudi" );
db.save( person );
```

```
// CREATES A NEW PERSON FROM A PARAMETRIZED CONSTRUCTOR
Person person = db.newInstance(Person.class, "Antoni", "Gaudi");
db.save( person );
```

Tuy nhiên, bất kỳ đối tượng Java nào cũng có thể được lưu lại bằng cách gọi phương thức `db.save()`, nếu không được tạo với API cơ sở dữ liệu sẽ được tuần tự hóa và lưu lại. Trong trường hợp này, người dùng phải gán kết quả của phương thức `db.save()` để có được cá thể được proxy, nếu không cơ sở dữ liệu sẽ luôn coi đối tượng là đối tượng mới. Thí dụ:

```
// REGISTER THE CLASS ONLY ONCE AFTER THE DB IS OPEN/CREATED
db.getEntityManager().registerEntityTypeClass(Animal.class);
```

```
Animal animal = new Animal();
animal.setName( "Gaudi" );
animal.setLocation( "Madrid" );
animal = db.save( animal );
```

Cập nhật đối tượng

Bất kỳ đối tượng proxy nào cũng có thể được cập nhật bằng ngôn ngữ Java và sau đó gọi phương thức `db.save()` để đồng bộ hóa các thay đổi đối với kho lưu trữ. Ví dụ:

```
animal.setLocation( "Nairobi" );
db.save( animal );
```

Xóa đối tượng

Để xóa một đối tượng, hãy gọi phương thức `db.delete()` trên một đối tượng proxy. Nếu được gọi trên một đối tượng không có, cơ sở dữ liệu sẽ không làm bất cứ điều gì.

```
db.delete( animal );
```

3.2.6.4 Attach

Với phương thức `attach` tất cả dữ liệu được chọn sẽ được copy trong một liên kết document. Hệ thống sẽ tự ghi đè những thông tin đã tồn tại trước đó.

```
Animal animal = database.newInstance(Animal.class);
animal.name = "Gaudi" ;
animal.location = "Madrid";
database.attach(animal);
database.save(animal);
```

Theo cách này, tất cả các thay đổi được thực hiện trong đối tượng mà không sử dụng setter.

3.2.6.5 Detach

Với phương thức Detach, tất cả dữ liệu chứa trong tài liệu sẽ được sao chép trong đối tượng liên quan, ghi đè tất cả các thông tin hiện có. Phương thức Detach(Object) trả về một đối tượng.

```
Animal animal = database.load(rid);  
database.detach(animal);
```

Trong ví dụ này sẽ sao chép tất cả các thông tin tài liệu được nạp vào đối tượng, mà không cần phải gọi tất cả các getters.

3.2.7 Traverse

OrientDB là một cơ sở dữ liệu đồ thị - graph database. Điều đó có nghĩa là trọng tâm của OrientDB là mối quan hệ (links) và cách chúng quản lý. Chuẩn SQL không đủ để làm việc với cây và đồ thị bởi vì nó thiếu khái niệm *đệ quy*. Đó là lý do tại sao OrientDB cung cấp một lệnh mới "traverse - duyệt đồ thị" khi duyệt qua cây và đồ thị. Duyệt đồ thị là một thao tác vượt qua mối quan hệ giữa các bản ghi (documents, vertexes, nodes,...). Thao tác này sẽ nhanh hơn thực hiện phép JOIN trong cơ sở dữ liệu quan hệ.

Khái niệm chính của duyệt đồ thị bao gồm:

- **target:** làm điểm bắt đầu để ghi lại các bản ghi. Có thể:
 - **class**
 - **cluster**
 - **Tập hợp các bản ghi**, đặt biệt là record ID.
 - **sub-command:** trả về *Iterable<OIdentifiable>*. Chúng ta có thể thực hiện nhiều câu select và duyệt đồ thị cùng nhau.
- **Trường:** duyệt qua các trường. Sử dụng ***, *any()* hoặc *all()* để duyệt qua các trường trong document.
- **Giới hạn - limit:** Con số tối đa của các bản ghi có thể tìm kiếm. Trong ví dụ sau đây chúng tôi sẽ trả về 10 kết quả

```
select from book limit 10
```

```
orientdb {db=Library}> select from book limit 10
```

#	@RID	@CLASS	work_id	book_id	ratings_2	ratings_1	ratings_4	ratings_3	ratings_5	isbn	books_cou	best_book	language
0	#21:0	Book	2792775	1	127936	66715	1481305	560092	2706317	439023483	272	2767052	eng
1	#21:1	Book	245494	5	197621	86236	936012	606158	947718	743273567	1356	4671	eng
2	#21:2	Book	3338963	9	145740	77841	716569	458429	680175	141652...	311	960	en-CA
3	#21:3	Book	153313	13	86425	41845	692021	324874	908229	451524934	995	5470	eng
4	#21:4	Book	6171458	17	48030	10492	687238	262010	980309	439023491	201	6148028	eng
5	#21:5	Book	2809203	21	31577	9528	494427	180210	1124806	439358078	307	2	eng
6	#21:6	Book	2963218	25	22245	9363	383914	113646	1318227	545010225	263	136251	eng
7	#21:7	Book	3349450	29	153179	57980	519822	452673	489235	743477111	1937	18135	eng
8	#21:8	Book	1558965	33	59033	23500	517157	258700	559782	739326228	220	929	eng
9	#21:9	Book	4790821	37	55542	19309	513366	262038	734629	60764899	474	100915	eng

```
10 item(s) found. Query executed in 0.027 sec(s).
orientdb {db=Library}>
```

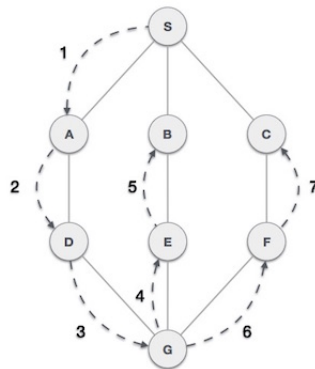
HÌNH 3.3. Tìm kiếm 10 bản ghi book

- **Vị từ - predicate:** làm vị từ để thực thi đối với từng tài liệu được duyệt qua. Nếu biến vị từ trả về true, tài liệu được trả về, nếu không nó sẽ bị bỏ qua.
- **Chiến lược - strategy:** chỉ ra là bạn duyệt qua đồ thị như thế nào:
 - DEPTH FIRST, mặc định.
 - BREADTH FIRST.

3.2.8 Chiến lược duyệt đồ thị

3.2.8.1 Tìm kiếm theo chiều sâu - DEPTH FIRST

Đây là chiến lược mặc định được OrientDB sử dụng để truyền tải. Nó khám phá càng xa càng tốt dọc theo từng nhánh trước khi quay ngược lại. Nó được thực hiện bằng cách sử dụng đệ quy. Đối với giải thuật này chúng ta sử dụng stack để lưu lại các node chưa được duyệt qua.

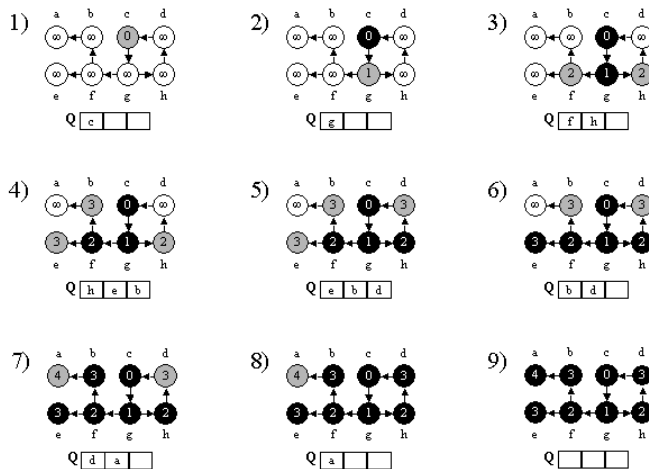


HÌNH 3.4. Tìm kiếm theo chiều sâu

(a) https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm

3.2.8.2 Tìm kiếm theo chiều rộng - BREADTH FIRST

Nó kiểm tra tất cả các nút lân cận, sau đó đối với mỗi nút lân cận lần lượt, nó kiểm tra các nút lân cận của chúng chưa được duyệt qua, v.v. Dãy thuật tìm kiếm theo chiều rộng sâu sắc hơn nhưng hiệu quả hơn với bộ nhớ.



HÌNH 3.5. Tìm kiếm theo chiều rộng

(a) http://alumni.cs.ucr.edu/~tmauch/old_web/cs141/cs141_pages/breadth_first_search.html

3.2.8.3 Nội dung giá trị

Trong duyệt đồ thị có một vài nội dung được quản lý và có thể được sử dụng bởi câu điều kiện của duyệt đồ thị.

- **\$depth** dưới dạng số nguyên chứa độ sâu làm của đồ thị trong quá trình duyệt. Cấp độ đầu tiên là 0.

- **\$path** dưới dạng biểu diễn chuỗi của vị trí hiện tại dưới dạng tổng các nút đi qua.
- **\$stack** dưới dạng stack hiện tại của nút được duyệt.
- **\$history** như toàn bộ tập hợp các nút đã viếng thăm.

3.2.8.4 SQL Traverse

Cách có sẵn đơn giản nhất để thực thi traversal là sử dụng lệnh SQL Traverse. Ví dụ, để lấy tất cả các bản ghi được kết nối từ và tới các bản ghi Movie lên đến mức độ sâu thứ 5:

```
for (OIdentifiable id : new OSQLSynchQuery<ODocument>(
    "traverse in, out from Movie while $depth <= 5")) {
    System.out.println(id);
}
```

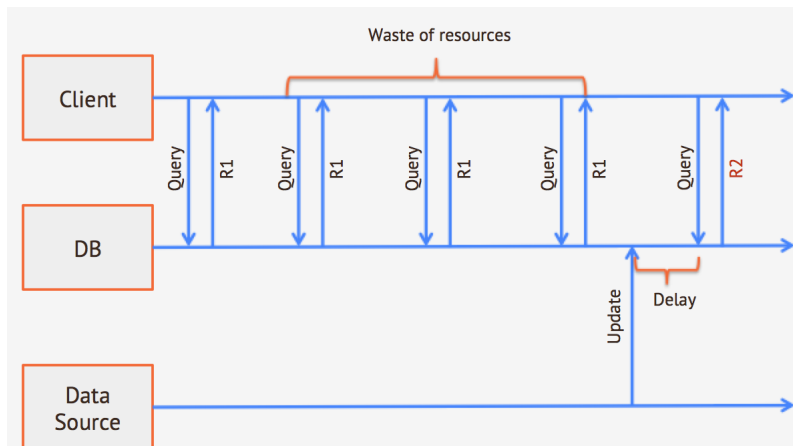
3.2.9 Live Query

Khi chúng ta muốn viết một ứng dụng có tính chất **chạy thời gian thực** và **đáp ứng nhanh** là nhiệm vụ khó khăn với mô hình truy vấn truyền thống. Hãy suy nghĩ về một trường hợp sử dụng đơn giản như cập nhật một trang web với dữ liệu mới đến từ cơ sở dữ liệu và giữ nó cập nhật theo thời gian; cũng xem xét rằng các bản cập nhật có thể được thực hiện bởi các nguồn dữ liệu khác nhau (nhiều ứng dụng, các thao tác DBA thủ công ...).

Với cách tiếp cận truyền thống, client phải kiểm tra vòng cơ sở dữ liệu để lấy dữ liệu mới. Cách tiếp cận này có ba vấn đề cơ bản:

- Client không bao giờ biết liệu có điều gì đó đã thay đổi trong DB hay không, do đó, nó sẽ thực hiện truy vấn vòng ngay cả khi không có gì thay đổi. Điều này có thể là một sự lãng phí tài nguyên lớn, đặc biệt khi truy vấn tốn kém.
- Nếu bạn cần dữ liệu thời gian thực, client sẽ thực hiện truy vấn vòng đến cơ sở dữ liệu rất thường xuyên.
- Kết quả đến khách hàng tại các khoảng thời gian cố định, vì vậy nếu thay đổi xảy ra trong cơ sở dữ liệu ở giữa khoảng thời gian đó, kết quả sẽ chỉ đến khách hàng tại truy vấn tiếp theo.

Quy trình hoạt động của truy vấn truyền thống được minh họa trong hình sau:



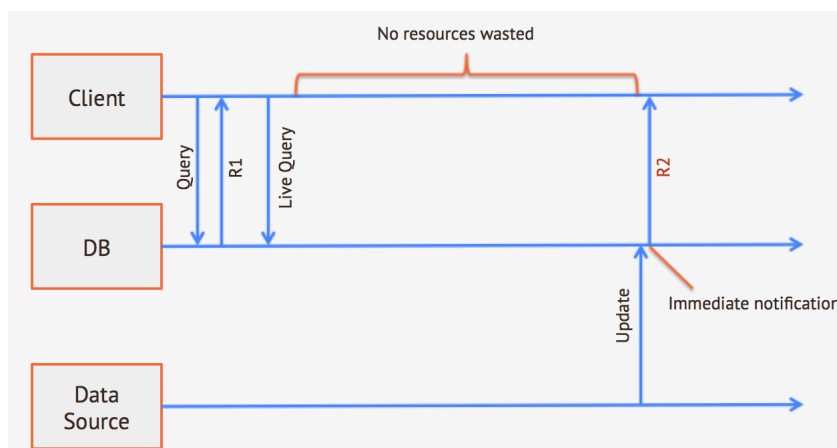
HÌNH 3.6. Mô hình truy vấn truyền thống

(a) <https://orientdb.com/docs/2.2.x/Live-Query.html>

Bạn phải lựa chọn ở như sau:

1. Bạn có thể quyết định có khoảng thời gian truy vấn dài, giảm chi phí thực thi, nhưng có kết quả được cập nhật sau.
2. Bạn có thể quyết định có khoảng thời gian truy vấn ngắn, có kết quả được cập nhật sớm hơn, nhưng với chi phí thực thi cao.

Với Live Query bạn có thể cập nhật những sự thay đổi trên lớp cụ thể (hoặc là trên tập con của bản ghi dựa vào câu điều kiện Where). OrientDB sẽ **push** sự thay đổi đến client ngay khi nó đang diễn ra ở DB.



HÌNH 3.7. Mô hình Live Query

(a) <https://orientdb.com/docs/2.2.x/Live-Query.html>

3.2.9.1 Sự khác nhau giữa truy vấn truyền thống và Live Query

Khi thực hiện câu lệnh `SELECT` truy vấn (cả đồng bộ và không đồng bộ) bạn mong rằng hệ thống sẽ trả ra kết quả là dữ liệu hiện tại trong database và khớp với tiêu chí lựa chọn của bạn. Bạn mong đợi kết quả của bạn được đặt là hữu hạn và truy vấn của bạn để thực thi trong một thời gian nhất định.

Nhưng Live query hoạt động theo cách hơi khác:

- Nó không trả về dữ liệu như câu truy vấn truyền thống.
- Nó trả sự thay đổi trong cơ sở dữ liệu từ thời điểm đó và phù hợp với tiêu chí của chúng ta.
- Nó không bao giờ kết thúc (trừ khi bạn chấm dứt nó hoặc một lỗi xảy ra).
- Máy chủ sẽ gửi cho bạn dữ liệu ngay sau khi chúng có sẵn, bạn chỉ cần cung cấp một callback.

Để làm cho sự khác biệt rõ ràng, đây là một ví dụ đơn giản (chỉ là luồng kết quả trong một siêu ngôn ngữ - meta language.(xem hình 3.3)

```
select from Book
```

Mặc khác, client chèn một dữ liệu mới vào DB.

```
insert into book set authors = "Ruby"
```

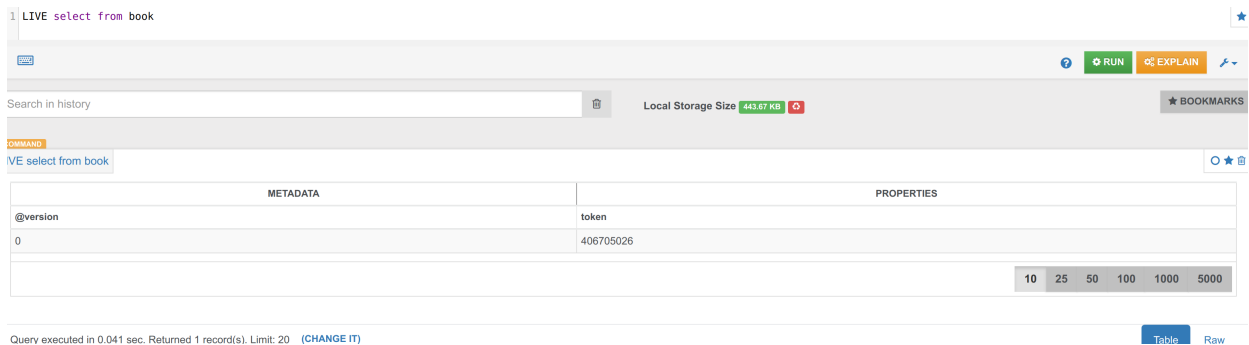
Client đầu tiên không nhận được bản ghi mới này, vì tập kết quả `SELECT` bị đóng. Tóm lại, thao tác `INSERT` này sẽ không ảnh hưởng đến truy vấn trước đó.

3.2.9.2 Live query

Client thực thi câu truy vấn:

```
LIVE select from book
```

Kết quả ngay lập tức của truy vấn này chỉ là số nhận dạng duy nhất của chính truy vấn đó (không có dữ liệu nào được trả lại, ngay cả khi dữ liệu có trong DB)



HÌNH 3.8. LIVE select from book

Client khác chèn dữ liệu mới vào DB

```
INSERT INTO book SET authors = 'Jenny'
```

Client đầu tiên sẽ nhận được tin nhắn với nội dung như sau:

```
content: {@rid: #21:0, name: 'Jenny'}
operation: insert
```

3.2.9.3 Các trường hợp sử dụng Live Query

LiveQuery đặc biệt hữu ích trong các trường hợp sau:

- Khi bạn cần cập nhật liên tục (realtime) và bạn có nhiều khách hàng truy cập vào các tập con dữ liệu khác nhau. Có hàng nghìn khách hàng thực hiện truy vấn liên tục có thể gặp bất kỳ máy chủ nào.
- Khi bạn có nhiều nguồn dữ liệu được chèn / cập nhật: nếu bạn có một nguồn dữ liệu đơn lẻ cư trú trong cơ sở dữ liệu, thì bạn có thể chặn nó và để nó trực tiếp thông báo cho khách hàng về các thay đổi.
- Khi bạn phát triển dựa trên cơ sở hạ tầng dựa trên push / reactive:

3.2.9.4 Live Query trong Java

Để hiện thực Live query trong java sử dụng các thành phần sau:

- Câu lệnh được thực thi ở lớp *OLiveQuery*.
- Nhận dữ liệu đầu ra ở lớp *OLiveResultListener*.

```
class MyLiveQueryListener implements OLiveResultListener {

    public List<ORecordOperation> ops =

        new
            ArrayList<ORecordOperation>();

    @Override
    public void onLiveResult(int iLiveToken,
        ORecordOperation iOp) throws OException {
        System.out.println("New result from server for
            live query "+iLiveToken);
        System.out.println("operation: "+iOp.type);
        System.out.println("content: "+iOp.record);
    }

    public void onError(int iLiveToken) {
        System.out.println("Live query terminate due to
            error");
    }

    public void onUnsubscribe(int iLiveToken) {
        System.out.println("Live query terminate with
            unsubscribe");
    }

}
```

3.2.10 Sự lan truyền transaction

Trong quá trình phát triển ứng dụng, có một số tình huống khi một giao dịch bắt đầu bằng một phương thức nên được truyền sang phương thức khác.

```
public void method1() {
    database.begin();
    try {
        method2();
        database.commit();
    } catch(Exception e) {
        database.rollback();
    }
}
```

```
}  
  
public void method2() {  
    database.begin();  
    try {  
        database.commit();  
    } catch(Exception e) {  
        database.rollback();  
    }  
}
```

Như bạn có thể thấy giao dịch được bắt đầu trong phương thức đầu tiên và sau đó phương thức mới được bắt đầu theo phương thức thứ hai. Vậy các giao dịch này nên tương tác với nhau như thế nào. Giao dịch đầu tiên đã được khôi phục và giao dịch thứ hai đã được bắt đầu vì vậy rủi ro là tất cả các thay đổi sẽ bị mất.

Có thể có hai trường hợp có thể xảy ra tại đây:

Đầu tiên:

- Bắt đầu một giao dịch bên ngoài.
- Bắt đầu các giao dịch lồng vào nhau.
- Commit giao dịch lồng vào nhau.
- Commit giao dịch bên ngoài.

Khi giao dịch lồng nhau được bắt đầu, tất cả các thay đổi của giao dịch bên ngoài được hiển thị trong giao dịch lồng nhau và sau đó khi giao dịch lồng nhau được thực hiện thay đổi trong giao dịch lồng nhau không được commit tại thời điểm giao dịch bên ngoài sẽ được thực hiện.

Thứ hai:

- Bắt đầu một giao dịch bên ngoài.
- Bắt đầu các giao dịch lồng vào nhau.
- Rollback giao dịch lồng vào nhau.
- Commit giao dịch bên ngoài.

Khi giao dịch lồng nhau được khôi phục, các thay đổi được thực hiện trong giao dịch lồng nhau không được khôi phục. Nhưng khi chúng tôi commit giao dịch bên ngoài, tất cả các

thay đổi sẽ được khôi phục.

Vì vậy, những trường hợp cơ sở dữ liệu nào chúng ta nên sử dụng để tận dụng lợi thế của tính năng lan truyền giao dịch:

1. Cùng một cơ sở dữ liệu nên được sử dụng cùng các phương thức.
2. Có thể sử dụng pool cơ sở dữ liệu, trong trường hợp này tất cả các phương thức yêu cầu kết nối db trong cùng một luồng sẽ có cùng một cá thể cơ sở dữ liệu giống nhau.

3.3 JDBC driver

Orientdb là một NoSQL DBMS nhưng hỗ trợ SQL và ngôn ngữ truy vấn.

3.3.1 JDBC là gì

JDBC (Java Database Connectivity) là một API tiêu chuẩn dùng để tương tác với các loại cơ sở dữ liệu quan hệ. JDBC có một tập hợp các class và các Interface dùng cho ứng dụng Java có thể nói chuyện với các cơ sở dữ liệu.



HÌNH 3.9. JDBC API

(a) <https://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc>

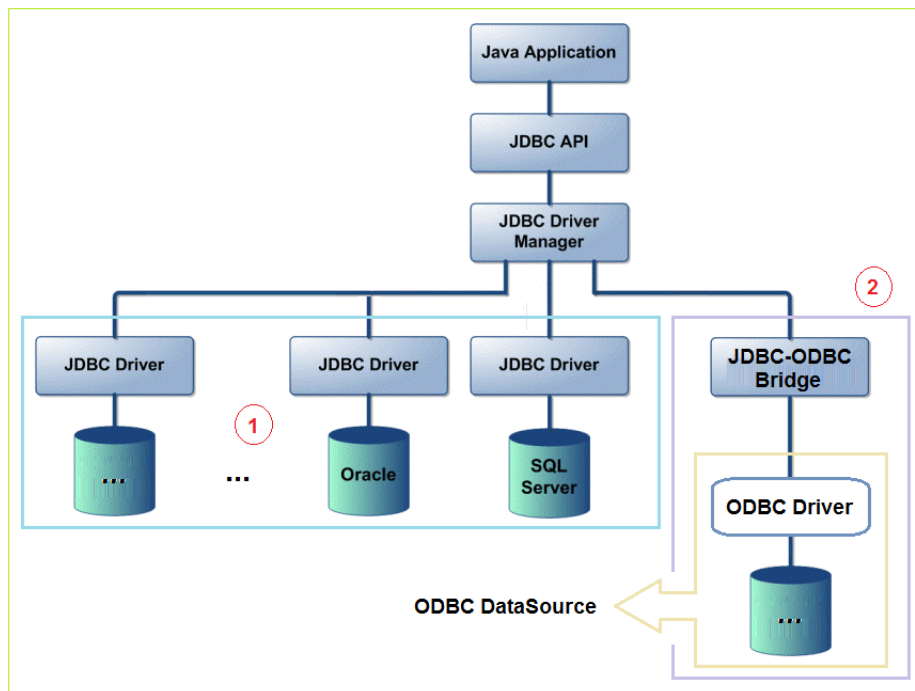
Các thành phần của JDBC Api về cơ bản bao gồm:

1. **DriverManager**: Là một class, nó dùng để quản lý danh sách các Driver (database drivers).
2. **Driver**: Là một Interface, nó dùng để liên kết các liên lạc với cơ sở dữ liệu, điều khiển các liên lạc với database. Một khi Driver được tải lên, lập trình viên không cần phải gọi nó một cách cụ thể.
3. **Connection**: Là một Interface với tất cả các method cho việc liên lạc với database. Nó mô tả nội dung liên lạc. tất cả các thông tin liên lạc với cơ sở dữ liệu là thông qua chỉ có đối tượng Connection.

4. **Statement:** Là một Interface, gói gọn một câu lệnh SQL gửi tới cơ sở dữ liệu được phân tích, tổng hợp, lập kế hoạch và thực hiện.
5. **ResultSet:** đại diện cho tập hợp các bản ghi lấy do thực hiện truy vấn.

3.3.1.1 Java kết nối với database dựa trên nguyên tắc nào?

Java sử dụng JDBC để làm việc với các cơ sở dữ liệu.

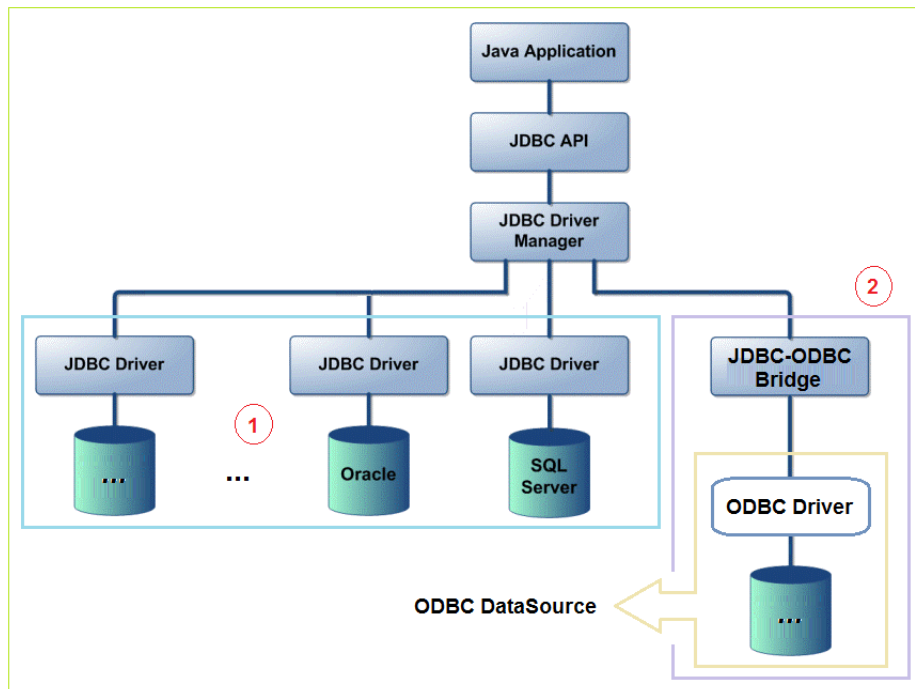


HÌNH 3.10. Kết nối database

(a) <https://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc>

Ví dụ bạn làm việc với cơ sở dữ liệu Oracle từ Java bạn cần phải có Driver (Đó là class điều khiển việc kết nối với loại cơ sở dữ liệu bạn muốn). Trong JDBC API chúng ta có `java.sql.Driver`, nó chỉ là một interface, và nó có sẵn trong JDK. Như vậy bạn phải download thư viện Driver ứng với loại Database mà bạn mong muốn. (xem `refJDBC`).

`java.sql.DriverManager` là một class trong JDBC API. Nó làm nhiệm vụ quản lý các Driver.



HÌNH 3.11. DriverManager

(a) <https://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc>

Chúng ta có 2 cách để làm việc với một loại cơ sở dữ liệu cụ thể nào đó.

1. Bạn hãy cung cấp thư viện Driver điều khiển loại cơ sở dữ liệu đó, đây là cách trực tiếp. Nếu bạn dùng DB oracle (hoặc DB khác) bạn phải download thư viện dành cho loại DB này.
2. Khai báo một "ODBC DataSource", và sử dụng cầu nối JDBC-ODBC để kết nối với "ODBC DataSource" kia. Cầu nối JDBC-ODBC là thứ có sẵn trong JDBC API.

3.3.2 Thêm JDBC driver vào project

```

<dependency>
    <groupId>com.orienttechnologies</groupId>
    <artifactId>orientdb-jdbc</artifactId>
    <version>1.7</version>
</dependency>
    
```

3.3.3 Hiện thực JDBC driver và code như thế nào

Trình điều khiển được đăng ký với Java SQL DriverManager và có thể được sử dụng để làm việc với tất cả các kiểu cơ sở dữ liệu OrientDB:

- memory
- plocal
- remote

Lớp của trình điều khiển là `com.orienttechnologies.orient.jdbc.OrientJdbcDriver`. Sử dụng kiến thức của bạn về JDBC API để làm việc với OrientDB.

3.3.3.1 Tạo kết nối

```
Properties info = new Properties();
info.put("user", "admin");
info.put("password", "admin");

Connection conn = (OrientJdbcConnection)
    DriverManager.getConnection("jdbc:orient:remote:localhost/test",
        info);
```

Sau đó thực thi câu lệnh và trả về một tập kết quả:

```
Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT stringKey, intKey,
    text, length, date FROM Item");

rs.next();

rs.getInt("@version");
rs.getString("@class");
rs.getString("@rid");

rs.getString("stringKey");
rs.getInt("intKey");

rs.close();
stmt.close();
```

Trình điều khiển truy xuất siêu dữ liệu OrientDB (@rid, @class và @version) chỉ trên các truy vấn trực tiếp. Hãy xem mã kiểm tra để xem các ví dụ chi tiết hơn.

3.3.3.2 Các tính năng tiên tiến

Theo mặc định, một cá thể cơ sở dữ liệu mới được tạo ra mỗi khi bạn yêu cầu kết nối JDBC. Trình điều khiển JDBC OrientDB cung cấp một Connection Pool. Đặt các tham số nhóm kết nối trước để yêu cầu kết nối:

```
Properties info = new Properties();
info.put("user", "admin");
info.put("password", "admin");

info.put("db.usePool", "true"); // USE THE POOL
info.put("db.pool.min", "3");   // MINIMUM POOL SIZE
info.put("db.pool.max", "30");  // MAXIMUM POOL SIZE

Connection conn = (OrientJdbcConnection)
    DriverManager.getConnection("jdbc:orient:remote:localhost/test",
        info);
```

Chương 4

SQL

Khi nói đến các ngôn ngữ truy vấn, SQL là tiêu chuẩn được công nhận rộng rãi nhất. Phần lớn các nhà phát triển có kinh nghiệm và cảm thấy thoải mái với SQL. Vì lý do này, OrientDB sử dụng SQL làm ngôn ngữ truy vấn và thêm một số tiện ích mở rộng để phục vụ một số hàm trong đồ thị. Có một vài khác biệt giữa cú pháp SQL chuẩn và được hỗ trợ bởi OrientDB, nhưng đối với hầu hết các phần, nó sẽ cảm thấy rất tự nhiên. Các khác biệt này được đề cập trong phần phương ngữ SQL OrientDB.

Nếu bạn đang tìm cách hiệu quả nhất để duyệt qua biểu đồ, chúng tôi khuyên bạn nên sử dụng SQL-Match thay thế.

Nhiều lệnh SQL có điều kiện WHERE. Các từ khóa và các tên lớp trong OrientDB SQL không phân biệt chữ hoa chữ thường. Tên trường và giá trị phân biệt chữ hoa chữ thường. Trong các ví dụ sau, từ khóa được viết hoa nhưng điều này không bắt buộc.

Ví dụ, nếu bạn có một lớp Book với một trường có tên là id, thì các câu lệnh SQL sau đây là tương đương:

```
SELECT FROM BOOK WHERE id = 1
select from book where id = 1
```

Chú ý 4.1. Tên trường 'ID' không giống với 'id'.

4.0.1 Tự động sử dụng chỉ mục

OrientDB cho phép bạn thực hiện các truy vấn đối với bất kỳ trường nào, được lập chỉ mục hoặc không được lập chỉ mục. Công cụ SQL tự động nhận ra nếu bất kỳ chỉ mục nào có thể được sử dụng để tăng tốc độ thực thi. Bạn cũng có thể truy vấn trực tiếp bất kỳ chỉ mục nào bằng cách sử dụng INDEX: <index-name> làm đích. Ví dụ:

```
SELECT FROM INDEX:myIndex WHERE key = 'Jay'
```

4.0.2 Lệnh JOIN

Sự khác biệt quan trọng nhất giữa OrientDB và một cơ sở dữ liệu quan hệ là các mối quan hệ được đại diện bởi LINKS thay vì JOIN.

Vì lý do này, cú pháp JOIN cổ điển không được hỗ trợ. OrientDB sử dụng ký hiệu "dấu chấm (.)" để điều hướng LINKS. Ví dụ 1: Trong SQL bạn có thể tạo một phép join như:

```
SELECT *
FROM Employee A, City B
WHERE A.city = B.id
AND B.name = 'Rome'
```

Trong OrientDB một thao tác tương đương sẽ là:

```
SELECT * FROM Employee WHERE city.name = 'Rome'
```

Nếu bạn sử dụng nhiều JOIN, thì tương đương SQL OrientDB sẽ là một lợi ích lớn hơn. Ví dụ 2: Trong SQL bạn có thể tạo một phép join như:

```
SELECT *
FROM Employee A, City B, Country C,
WHERE A.city = B.id
AND B.country = C.id
AND C.name = 'Italy'
```

Trong OrientDB một thao tác tương đương sẽ là:

```
SELECT * FROM Employee WHERE city.country.name = 'Italy'
```

4.0.3 Thực hiện truy vấn từ nhiều đối tượng

OrientDB chỉ cho phép một lớp trái ngược với SQL, cho phép nhiều bảng làm đích. Nếu bạn muốn chọn từ 2 lớp, bạn phải thực hiện 2 truy vấn con và nối chúng với hàm UNIONALL:

```
SELECT FROM E, V
```

Trong OrientDB, bạn có thể thực hiện điều này với một vài định nghĩa biến và bằng cách sử dụng hàm mở rộng union:

```
SELECT EXPAND( $c ) LET $a = ( SELECT FROM E ), $b = ( SELECT
FROM V ), $c = UNIONALL( $a, $b )
```

4.0.4 Truy vấn lượt đồ

Lấy ra tất cả các lớp được cấu hình:

```
select expand(classes) from metadata:schema
```

```
orientdb (db=Library)> select expand(classes) from metadata:schema
```

#	overSize	abstract	shortName	strictMod	superClass	name	descripti	superClass	customFile	clusterId	defaultCl	clusterSe	properties
0	0.0	false		false	OIdentity	ORole		[OIden...		[4]	4	round...	[(name:inheritedRole,type:13,globalId...
1	0.0	false		false	OSequence	V		[V]		[7]	7	round...	[(name:incr,type:1,globalId:16,manda...
2	0.0	false		false	V	Tag		[V]		[9,10,...	9	round...	[(name:tag_id,type:1,globalId:23,man...
3	0.0	false		false	OIdentity	OUser		[OIden...		[5]	5	round...	[(name:status,type:7,globalId:6,mand...
4	0.0	false		false	E	hasTag		[E]		[33,34,...	33	round...	[(name:book_id,type:1,globalId:24,ma...
5	0.0	false		false	V	Book		[V]		[21,22,...	21	round...	[(name:book_id,type:1,globalId:24,ma...
6	0.0	false		false	E	E		[E]		[13,14,...	13	round...	[(name:book_id,type:1,globalId:24,ma...
7	0.0	false		false	E	hasBook		[E]		[29,30,...	29	round...	[(name:book_id,type:1,globalId:24,ma...
8	0.0	false		false	OIdentity	OFunction		[OIden...		[6]	6	round...	[(name:parameters,type:10,globalId:1...
9	0.0	true		false	OFunction	studio		[OIden...		[37]	37	round...	[(name:parameters,type:10,globalId:1...
10	0.0	false		false	OFunction	OFunction		[OIden...		[1]	1	round...	[(name:parameters,type:10,globalId:1...
11	0.0	false		false	OFunction	OFunction		[OIden...		[8]	8	round...	[(name:parameters,type:10,globalId:1...
12	0.0	true		false	OFunction	OFunction		[OIden...		[25,26,...	25	round...	[(name:parameters,type:10,globalId:1...
13	0.0	false		false	OFunction	OFunction		[OIden...		[1]	1	round...	[(name:parameters,type:10,globalId:1...
14	0.0	false		false	V	book_tag		[V]		[25,26,...	25	round...	[(name:parameters,type:10,globalId:1...
15	0.0	true		false	ORestr...	ORestr...		[OIden...		[1]	1	round...	[(name:allowRead,type:15,globalId:8...

16 item(s) found. Query executed in 0.037 sec(s).

```
orientdb (db=Library)> 
```

HÌNH 4.1. Lấy ra tất cả các lớp

Lấy ra tất cả các cấu hình của lớp OUser:

```
select expand(properties) from (
    select expand(classes) from metadata:schema
) where name = 'OUser'
```

```
select expand(properties) from ( select expand(classes) from metadata:schema ) where name = 'OUser'
```

METADATA				PROPERTIES										
@version	regex	type	globalId	mandatory	readonly	notNull	name	min	max	linkedClass	customFields	collate	description	defaultValue
0		15	5	false	false	false	roles			ORole		default		
0		7	6	true	false	true	status					default		
0	\S+(\."S+")	7	0	true	false	true	name	1				ci		
0		7	4	true	false	true	password					default		

10255010010005000

HÌNH 4.2. Lấy ra tất cả các cấu hình của lớp OUser

4.0.4.1 Truy vấn chỉ mục có sẵn

Lấy ra tất cả các cấu hình của chỉ mục

```
select expand(indexes) from metadata:indexmanager
```

```
orientdb (db=Library)> select expand(indexes) from metadata:indexmanager
```

#	algorithm	indexVers	name	type	valueContains	clusters	metadata	indexDefinitionClass	indexDefinition
0	SBTREE	1	ORole...	UNIQUE	NONE	[orole]		com.orienttechnologies.ori...	{className:ORole,field:name,keyTyp...
1	SBTREE	1	OUser...	UNIQUE	NONE	[ouser]		com.orienttechnologies.ori...	{className:OUser,field:name,keyTyp...
2	HASH I...	2	OFunc...	UNIQUE	NONE	[ofunction]		com.orienttechnologies.ori...	{className:OFunction,field:name,ke...
3	SBTREE	1	Book.b...	UNIQUE	NONE	[book_1,boo...		com.orienttechnologies.ori...	{className:Book,field:book_id,keyT...
4	SBTREE	1	dictio...	DICTIO...	NONE	[i]	{durableInNonTx...	com.orienttechnologies.ori...	{keyTypes:[1],collate:default,null...
5	SBTREE	1	Tag.ta...	UNIQUE	NONE	[itag,tag_1,...		com.orienttechnologies.ori...	{className:Tag,field:tag_id,keyTyp...

HÌNH 4.3. Lấy ra tất cả các cấu hình của chỉ mục

4.0.5 So khớp trong SQL

Truy vấn cơ sở dữ liệu theo cách khai báo, sử dụng so khớp mẫu. Cú pháp đơn giản

```
MATCH
{
    [class: <class>],
    [as: <alias>],
    [where: (<whereCondition>)]
}
.<functionName>() {
    [class: <className>],
    [as: <alias>],
    [where: (<whereCondition>)],
    [while: (<whileCondition>)],
    [maxDepth: <number>],
    [optional: (true | false)]
}*
RETURN <expression> [ AS <alias> ] [, <expression> [ AS <alias>
    ]]*
LIMIT <number>
```

- **<class>** Xác định lớp được chọn.
- **<alias>** Xác định alias của một nút trong một mẫu.
- **<whereCondition>** Định nghĩa điều kiện lọc để so khớp tổng mẫu.
- **<functionName>** Xác định một graph function đại diện cho kết nối giữa hai nút. Ví dụ, out (), in (), outE (), inE (), v.v. Đối với out (), in (), all() cũng có một cú pháp mũi tên rút ngắn được hỗ trợ.
- **<whileCondition>** Xác định điều kiện mà câu lệnh phải đáp ứng để cho phép duyệt trên path. Nó hỗ trợ mệnh đề SQL WHERE bình thường.
- **maxDepth** Xác định độ sâu tối đa của một đường đơn.
- **RETURN <expression> [AS <alias>]** Xác định các phần tử trong mẫu mà bạn muốn trả về. Nó có thể sử dụng một trong các cách sau:
 1. **\$Aliases** xác định như là một block.
 2. **\$matches** chỉ ra tất cả các định nghĩa trên alias.
 3. **\$paths** chỉ ra tất cả các đường được duyệt qua.

4. **\$elements** Chỉ ra rằng tất cả các phần tử sẽ được trả về bởi \$matches phải được trả duy nhất, không có bản sao.
 5. **\$pathElements** Chỉ ra rằng tất cả các phần tử sẽ được trả về bởi các \$path phải được trả về duy nhất, không trùng lặp.
- **optional** Nếu được đặt thành true, cho phép đánh giá và trả lại mẫu ngay cả khi nút cụ thể đó không khớp với chính mẫu đó (ví dụ: không có giá trị cho nút đó trong mẫu).

4.0.5.1 Ví dụ

COMMAND
select from person

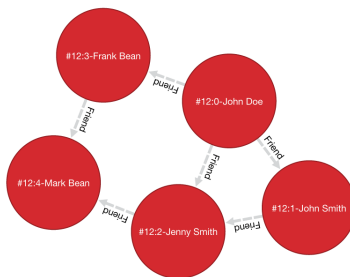
METADATA			PROPERTIES			IN	OUT
@rid	@version	@class	surname	name	fullName	Friend	Friend
#12:0	8	Person	Doe	John	#12:0-John Doe		#12:1 #12:3 #12:2
#12:1	7	Person	Smith	John	#12:1-John Smith	#12:0	#12:2
#12:2	8	Person	Smith	Jenny	#12:2-Jenny Smith	#12:1 #12:3	#12:4
#12:3	7	Person	Bean	Frank	#12:3-Frank Bean	#12:0	#12:4
#12:4	7	Person	Bean	Mark	#12:4-Mark Bean	#12:3 #12:2	

10 25 50 100 1000 5000

Query executed in 0.009 sec. Returned 5 record(s). Limit: 20 (change it)

Table Raw

HÌNH 4.4. select from person



HÌNH 4.5. Đồ thị diễn tả quan hệ giữa các Person

Tài liệu tham khảo

[1] : <https://orientdb.com/docs/2.2.x/>