TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

KHOA CÔNG NGHỆ THÔNG TIN

Báo cáo nghiên cứu hệ quản trị cơ sở dữ liệu

Chủ đề: Oracle NoSQL database

Giảng viên hướng dẫn: Ths Dư Phương Hạnh

Nhóm thực hiện (Nhóm 7):

Trần Quang Tuấn – 16021218

Đoàn Thị Hoài Thu – 16021167

Trần Hữu Tuân – 16021661

Nguyễn Đức Thiện – 16021164

Mục Lục

Danh mục các bảng

Danh mục các hình ảnh

Phần 1. Tổng quan NoSQL:

I. Đặt vấn đề

Với sự phát triển không ngừng nghỉ của công nghệ thông tin, dữ liệu được người dùng tạo ra ngày càng lớn dẫn đến dữ liệu cần được xử lý trong ứng dụng cũng lớn dần. Đặc biệt với sự bùng nổ của công nghệ web 2.0, nơi các mạng xã hội cho phép người dùng tự tạo ra nội dung web, dẫn đến dữ liệu tăng rất nhanh vượt qua giới hạn xử lý của các Hệ quản trị cơ sở dữ liệu truyền thống.

RDBMS (Relative Database management system) vẫn còn một số khuyết điểm:

- Việc mapping giữa các bảng trong database với cá object trong code khá rắc rối và phức tạp. (Mặc dù 1 số ORM như Entity Framework, Hibernate đã đơn giản hóa chuyện này).
- Hiệu năng không cao khi phải join nhiều bảng đê lấy dữ liệu (Đó là lý do ta "giảm chuẩn" để tăng hiệu suất cho RDBMS)
- Việc thay đôi cấu trúc dữ liệu(Thêm/xóa bảng hoặc field) rất tốn công sức và làm ảnh hưởng đến cấu trúc của toàn bộ cơ sở dữ liệu.
- Không làm việc được với dữ liệu không có cấu trúc.
- RDBMS được thiết kế để chạy trên một máy chủ. Khi muốn mở rộng, nó khó có thể chạy trên nhiều máy.

Vì những lí do trên, để đáp ứng được nhu cầu của xã hội đòi hỏi một hệ quản trị cơ sở dữ liệu mới có thể lưu trữ và sử lý dữ liệu một cách nhanh chóng và linh hoạt hơn. NoSQL là một giải pháp để giải quyết vấn đề trên.

II. Giới thiệu NoSQL

NoSQL là một khái niệm chỉ về một lớp cá hệ cơ sở dữ liệu không sử dụng mô hình quan hệ (RDBMS). NoSQL đánh dấu bước phát triển của thế hệ CSDL mới: phân tán (distributed) + không rằng buộc (non-relational). Đây là 2 đặc tính quan trọng nhất của NoSQL.

NoSQL ra đời với những ưu điểm sau:

- Dữ liệu trong NoSQL DB được lưu dưới dạng document, object. Truy vấn dễ dàng và nhanh hơn RDBMS nhiều.
- NoSQL có thể làm việc hoàn toàn ổn với dữ liệu dạng không có cấu trúc.
- Việc thay đổi cấu trúc dữ liệu (Thêm, xóa trường hoặc bảng) rất dễ dàng và nhanh gọn trong NoSQL.
- Vì không đặt nặng tính ACID của transactions và tính nhất quán của dữ liệu, NoSQL DB có thể mở rộng, chạy trên nhiều máy một cách dễ dàng

III. So sánh so với RDBMs

Bảng 1: So sánh tính năng của RDBMs và NoSQL

Tính năng	RDBMs	NoSQL
Hiệu suất	Kém SQL Relational-database	Tốt hơn NoSQL Non-relational database
Khả năng mở rộng	Hạn chế về lượng	Hỗ trợ một lượng lớn các node

Hiệu suất đọc – ghi	Kém do thiết kế đảm bảo dự vào ra liên tục của dữ liệu	Tốt do mô hình xử lý lô và các tối ưu về đọc ghi dữ liệu
Thay đổi số node trong hệ thống	Phải shutdown cả hệ thống, việc thay đổi node phức tạp do relational-database	Không cần phải shutdown cả hệ thống, việc thay đổi node đơn giản không ảnh hưởng tới hệ thống
Yêu cầu phần cứng	Đòi hỏi cao về phần cứng	Đòi hỏi thấp hơn về giá trị và tính đồng nhất của phần cứng

IV. Đặc điểm

- **High scalability:** Gần như không có giới hạn cho dữ liệu và người dùng trong hệ thống.
- **High availability:** Chấp nhận sự trùng lặp trong lưu trữ, có thể một số node lưu trữ cùng một thông tin giống. Nếu một node nào đó bị chết cũng không ảnh hưởng đến hệ thống.
- **Consistency:** Chấp nhận nhất quán yếu, cập nhật mới không đảm bảo các truy suất ngay sau đó thấy được sự thay đổi. Do hệ thống lưu trữ phân tán chấp nhận bị ảnh hưởng theo phương thức truyền cần có một khoảng thời gian nhất định cho đến khi hệ thống đi vào trạng thái nhất quán.
- **Deployment Flexibility:** Việc thêm/ xóa node sẽ được hệ thống tự nhận biết mà lưu trữ không cần thông qua can thiệp bằng tay(shutdown hệ thống). Hệ thống không đòi hỏi phần cứng mạnh.
- **Non-relational:** Trong khi các RDBMs sử dụng các quan hệ để ràng buộc các tables (primary key + foreign key) để thể hiện tính nhất quán của dữ liệu thì Non-relational là không sử dụng rằng buộc cho nhất quán dữ liệu.

- **Modeling Flexibility:** Việc lưu trữ dữ liệu linh hoạt dưới nhiều dạng: Key-value pair, Hierarchical data, Graphs.
- **Query Flexibility:** Có thể load một tập giá trị dựa vào dãy các khóa.
- **High performance:** Hiệu suất cao.

V. Các dạng NoSQL cơ bản

- **Key – value data stores:** Dữ liệu lưu dưới dạng cặp key – value. Giá trị được truy suất thông qua khóa (key).
    - **Ví dụ:** Redis, Dynomite, MemcacheDB
    - **Ứng dụng:** Dùng cho content caching Applications
    - **Ưu điểm:** Tìm kiếm rất nhanh
    - **Nhược điểm:** Lưu dữ liệu không theo dạng (schema) nhất định.
- **Column-base:** Dữ liệu lưu dưới dạng bảng. Gần giống với mô hình RDBMS. Tuy nhiên, chúng lưu dữ liệu bởi các cột chứ không phải dòng. Nó khá thích hợp để hiển thị bằng các phần mềm quản lý kho dữ liệu.
    - **Ví dụ:** Apache Hbase, Apache Cassandra, Hypertable
    - **Ứng dụng:** Dùng cho các hệ phân tán file
    - **Ưu điểm:** Tìm kiếm nhanh, phân tán dữ liệu tốt
    - **Nhược điểm:** Hỗ trợ được với rất ít phần mềm
- **Document-base:** Dữ liệu (bán cấu trúc hay semi-structured) được lưu trữ và tổ chức dưới dạng một tập hợp các document. Các document này linh hoạt, mỗi document có một tập nhiều trường.
    - **Ví dụ:** Apache CounchDB, MongoDB
    - **Ứng dụng:** Dùng cho các ứng dụng web
    - **Ưu điểm:** Dùng được khi dữ liệu nguồn không được mô tả đầy đủ
    - **Nhược điểm:** Hiệu năng truy vấn không cao, cú pháp không rõ ràng

- **Graph-base data-stores:** Những CSDL này áp dụng lý thuyết đồ thị trong khoa học máy tính để lưu trữ và truy suất dữ liệu. Chúng tập trung vào tính rời rạc giữa các phần dữ liệu. Các phần tử đơn vị dữ liệu được biểu thị như một nút và liên kết với các thành phần khác nhau bằng cạnh.
    - **Ví dụ:** Neo4j, InfiniteGraph, DEX
    - **Ứng dụng:** mạng xã hội
    - **Ưu điểm:** Ứng dụng được các thuật toán trên đồ thị
    - **Nhược điểm:** Phải duyệt nội bộ đồ thị, không dễ để phân tán.

Phần 2. Oracle NoSQL Database

I. Giới thiệu về Oracle NoSQL Database

1. Thế nào là Oracle NoSQL Database

Hình 1: Oracle NoSQL database

- Oracle NoSQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở ban đầu được xây dựng dựa trên mô hình cơ sở dữ liệu dạng key-value, với khả năng lưu trữ lớn, thông lượng có thể mở rộng và hiệu suất tuyệt vời. Tuy nhiên đến tháng 4/2014, với phiên bản 3.0 thì Oracle NoSQL đã mở rộng mô hình dữ liệu key/value strore đến mô hình dữ liệu hướng tài liệu document-oriented.
- Oracle NoSQL hỗ trợ đa nền tảng: linux,window cho đến OS X.
- Oracle NoSQL hỗ trợ giao dịch với các tính chất Automicity – nguyên tố, Consistency – nhất quán, Isolation – tách biệt, Durability – bền vững, hỗ trợ bảng và tất nhiên như nhiều CSDL NoSQL khác, Oracle NoSQL hỗ trợ cấu trúc kiểu dữ liệu JSON.
- Oracle NoSQL tiến hành mã hóa SSL các phiên và cổng mạng bảo vệ hạn chế cho sự xâm nhập mạng và bảo vệ khỏi những truy cập trái phép.

2. Kiến trúc và các khái niệm cơ bản

2.1. Các khái niệm:

- **Storage node:** Là một bộ nhớ vật lý lưu trữ dữ liệu của Oracle NoSQL databases
- **Replication Node:** là một cơ sở dữ liệu độc lập lưu trữ bảng hoặc các cặp khóa/giá trị
- **Shard:** Là một tập hợp của các Replication Node với một Master Node và các node con
- **Replication Factor:** Số lượng node thuộc một shard
- **Partition:** lưu trữ các khóa để truy cập vào dữ liệu
- **Zone:** Nơi đặt các Storage node
- **Topology:** tập hợp các zone, storage node, replication node, shard, dịch vụ quản trị tạo nên cơ sở dữ liệu NoSQL.

2.2. Kiến trúc:

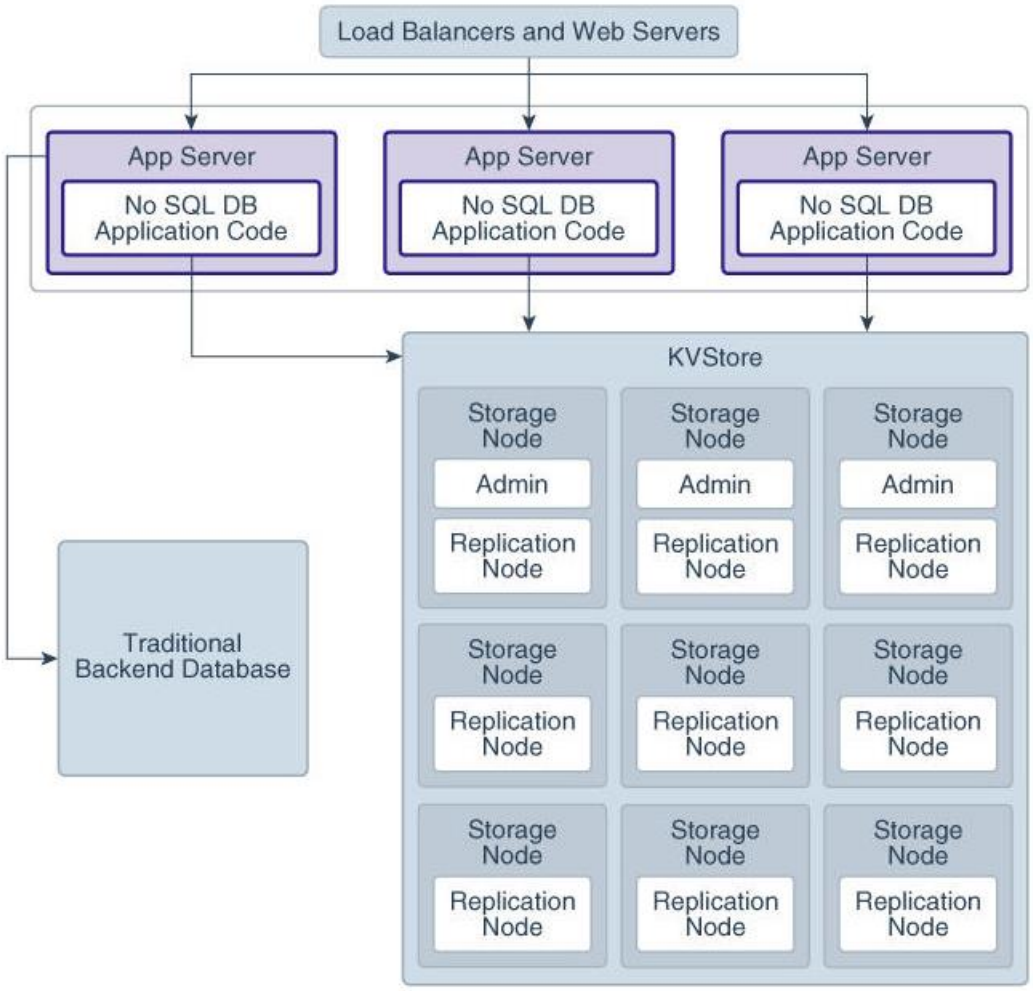
Các ứng dụng sử dụng Oracle NoSQL Database đọc và ghi dữ liệu bằng cách thực hiện các yêu cầu với một kho dữ liệu Oracle NoSQL Database, được gọi chung là KVStore. KVStore là tập hợp các Storage Nodes, mỗi một nút lưu trữ một hoặc nhiều Replication Nodes. Dữ liệu được tự động trải rộng trên các Replication Nodes bởi các cơ chế bên trong KVStore.

Với kiến trúc web ba tầng truyền thống, KVStore có thể nơi chạy back-end hoặc là chạy song song với nó. Một KVStore có thể được cài đặt dọc theo nhiều địa chỉ vật lý, mỗi vị trí gọi là một Zone.

Một Zone là một vị trí địa lý hỗ trợ kết nối mạng dung lượng cao giữa các nút lưu trữ được triển khai bên trong đó. Thông thường, mỗi vùng bao gồm nguồn cung cấp điện dự phòng, thiết bị điều khiển môi trường (ví dụ: điều hòa, bình cứu hỏa, …) và thiết bị bảo mật. Một zone có thể là một tòa nhà trung tâm dữ liệu vật

lý, một phòng, một tầng, khoang hoặc giá đỡ, tùy thuộc vào việc triển khai cụ thể. Mỗi kho gồm nhiều nút lưu trữ (Storage Nodes). Một nút lưu trữ(Storage Node) là một máy vật lý(hoặc ảo) với một bộ nhớ local riêng biệt.

Dưới đây là hình ảnh một cấu trúc điển hình của một kho dữ liệu Oracle NoSQL Database:



Hình 2: Mô hình kiến trúc điển hình của Oracle NoSQL database

Đây là mô hình top-down mô tả một mô hình đơn giản với ba phần. Phần trên cùng là cân bằng tải và quản trị của web server cho ba ứng dụng chạy trên server. Mỗi server chạy một ứng dụng NoSQL database.

Ở phần 3, bên trái là phần backend của database, bên phải là KVStore. KVStore bao gồm chín nút lưu trữ. Mỗi nút lưu trữ đặt một Replication Node (nút nhân rộng). Nút nhân rộng (Replication Node) có thể hiểu là một cơ sở dữ liệu độc lập lưu trữ bảng hoặc cặp khóa – giá trị. Một nút lưu trữ có thể chứ một hoặc nhiều nút nhân rộng, nhưng thường một nút lưu trữ chỉ chứa một nút nhân rộng.

Các Replication Node được tổ chức thành các shards (phân đoạn). Một shard chứa một master replication Node mô tả toàn bộ các hoạt động ghi trên cơ sở dữ liệu vào các replica. Có thể có một hoặc nhiều hơn replica chỉ hỗ trợ các service đọc dữ liệu.

Số lượng node thuộc một shard được gọi là Replication Factor. Replication Factor càng lớn, khả năng đọc của server càng tốt nhưng ngược lại khả năng ghi lại giảm. Một StorageNode có thể được cài đặt tại các zone khác nhau. Mỗi zone được cài đặt một Replication Factor, Oracle NoSQL database sẽ đảm bảo số lượng các nút được nhân bản là thích hợp cho mỗi shard và trong mỗi zone của Storage Node.

Tất cả dữ liệu được truy cập thông qua một hoặc nhiều khóa. Một khóa có thể là một cột trong bảng hoặc là một giá trị khóa trong cặp khóa/giá trị. Các khóa được lưu trữ trong các bộ chứa logic được gọi là partitions, mỗi shard chứa một hoặc nhiều partition. Một khi khóa được lưu trong partitions, nó không thể di chuyển sang partition khác.

Có 3 nút lưu chữ chạy quy trình quản trị.

Bằng cách thực hiện kết nối với một trong các quy trình quản trị, quản trị viên và các người dùng khác được ủy quyền có thể gọi các tiện ích, thực hiện thiết lập và các tác vụ khác.

3. Các cú pháp cơ bản

Oracle NoSQL hỗ trợ lưu trữ dữ liệu dạng bảng và dạng khóa/giá trị. Trong khuôn khổ bản báo cáo này sẽ đưa ra các cú pháp cơ bản thao tác với cơ sở dữ liệu

dạng bảng với các câu lệnh DDL statement trên Command Line và ORM Java. Các bạn muốn tìm hiểu các bộ thư viện dành cho ngôn ngữ khác có thể đi đến liên kết.

3.1. Thao tác với bảng table:

3.1.1 Tạo bảng

Việc tạo bảng trên Oracle NoSQL được xử lý thông qua các câu lệnh gần giống với SQL:

```
CREATE TABLE myTable (
  item STRING,
  description STRING,
  count INTEGER,
  percentage DOUBLE,
  PRIMARY KEY (item) // Every table must have a primary key
)
```

Hình 3: Câu lệnh tạo bảng

Khi chạy trên Command Line, ta đưa những câu lệnh trên vào trong dấu nháy đơn và đặt trong hàm **executed(“”)**

```
kv-> execute "CREATE TABLE myTable (
> item STRING,
> description STRING,
> count INTEGER,
> percentage DOUBLE,
> PRIMARY KEY (item))"
Statement completed successfully
kv->
```

Hình 4: Cú pháp trên CMD

Oracle NoSQL Database hỗ trợ các kiểu dữ liệu: Array(chuỗi giá trị cùng kiểu dữ liệu), Binary, Boolean, Double, Enum, Fixed Binary, Float, Integer, Json, Long, Number, Map, Records, String, Timestamp,…Ngoài ra, Oracle NoSQL Database còn hỗ trợ tạo bảng con (Child table) bên trong một bảng chính để lưu trữ các

thông tin bổ sung. Nếu chỉ cần các yêu cầu đơn giản cho dữ liệu, ta có thể dùng các record thay vì bảng con. Ví dụ:

+, Tạo bảng với record:

```
CREATE TABLE myContactsTable (
    uid STRING,
    surname STRING,
    familiarName STRING,
    homePhone STRING,
    workPhone STRING,
    homeAddress RECORD (street STRING, city STRING, state STRING,
                zip INTEGER),
    workAddress RECORD (street STRING, city STRING, state STRING,
                zip INTEGER),
    PRIMARY KEY(uid))
```

Hình 5: Câu lệnh tạo bảng với record

+ Tạo bảng con:

```
CREATE TABLE myInventory.itemDetails (
    itemSKU STRING,
    itemDescription STRING,
    price FLOAT,
    inventoryCount INTEGER,
    PRIMARY KEY (itemSKU)
)
```

Hình 6: Câu lệnh tạo bảng con

Với itemDetails là bảng con của bảng myInventory.

Mỗi bảng đều phải có ít nhất một trường được đặt làm khóa chính khi khởi tạo và không thể thay đổi sau đó. Mỗi giá trị của khóa là duy nhất và không trùng lặp.

```
CREATE TABLE myProducts (
    productName STRING,
    productType STRING,
    productClass STRING,
    color ENUM (blue,green,red),
    size ENUM (small,medium,large),
    inventoryCount INTEGER,
    // Define the primary key. Every table must have one.
    PRIMARY KEY (productName, productType, productClass)
)
```

Hình 7: Câu lệnh tạo bảng với nhiều khóa

Trong trường hợp này, cả ba trường productName, productType và productClass đều là khóa của bảng.

Ngoài khóa chính của bảng, Oracle còn cung cấp khóa chính của shard. Khóa của shard định nghĩa trường có ý nghĩa nhất trong toàn bộ kho dữ liệu bên trong shard.

```
PRIMARY KEY (SHARD(productType, productName), productClass)
```

Hình 8: Câu lệnh tạo shard key

3.1.2 Update bảng

Thao tác update bảng của Oracle NoSQL khi nó đã được chạy cùng với ứng dụng cũng gần giống với MySQL

```
ALTER TABLE myInventory.itemDetails (ADD salePrice FLOAT)
```

Hình 9: Câu lệnh thêm cột vào bảng

```
ALTER TABLE myInventory.itemDetails (DROP inventoryCount)
```

Hình 10: Câu lệnh xóa cột khỏi bảng

3.2 Tạo Index

Oracle NoSQL database cũng hỗ trợ index cho phép duyệt bảng nhanh hơn. Cú pháp khi tạo một index cho một cột trong bảng:

```
CREATE INDEX inventoryIdx on myInventory.itemDetails(inventoryCount)
```

Hình 11: Câu lệnh tạo index cho bảng

Với inventoryIdx là tên cột index, myInventory.itemDetails là bảng con vừa tạo ở trên, inventoryCount là cột tạo index.

Tương tự, việc xóa index trong bảng cũng rất đơn giản:

```
DROP INDEX inventoryIdx on myInventory.itemDetails
```

Hình 12: Câu lệnh xóa index khỏi bảng

3.3 Chèn row vào bảng

Oracle NoSQL database chỉ hỗ trợ chèn row vào bảng thông qua ORM hoặc import JSON file chứa các dòng của bảng.

Với chèn row:

- Khởi tạo một lớp gọi bảng ghi dữ liệu bằng cách gọi câu lệnh *KVStore.getTableAPI()* và tạo đối tượng bảng bằng câu lệnh *TableAPI.getTable()*
- Dùng câu lệnh *Table.createRow()* để tạo một đối tượng hàng
- Dùng lệnh *Row.put()* để viết từng trường vào hàng
- Ghi nhận dữ liệu dòng mới vào kho dữ liệu dùng lệnh *TableAPI.put()*.

Ví dụ về việc ghi hàng:

```
// Get a Row instance
Row row = myTable.createRow();

// Use row.put to put all of the cells into the row.
// This does NOT actually write the data to the store.

row.put("item", "Bolts");
row.put("description", "Hex head, stainless");
row.put("count", 5);
row.put("percentage", 0.2173913);

// Now write the table to the store.
// "item" is the row's primary key. If we had not set that key and its value,
// this operation will result in an IllegalArgumentException.
tableH.put(row, null, null);
```

Hình 13: Câu lệnh ghi hàng

3.4 Xóa hàng

Ta chỉ cần dùng câu lệnh *TableAPI.delete()*. Tuy nhiên, việc xóa hàng phải dựa trên khóa chính, ta cần phải dùng lệnh *Table.createPrimaryKey()* để lấy ra khóa và xóa cùng với hàng.

```
// Get the primary key for the row that we want to delete
PrimaryKey primaryKey = myTable.createPrimaryKey();
primaryKey.put("item", "Bolts");

// Delete the row
// This performs a store write operation
tableH.delete(primaryKey, null, null);
```

Hình 14: Câu lệnh xóa hàng

Ta cũng có thể xóa nhiều hàng cùng một lúc với cú pháp tương tự.

```
// Get the primary key for the row that we want to delete
PrimaryKey primaryKey = myTable.createPrimaryKey();
primaryKey.put("itemType", "Hats");
primaryKey.put("itemCategory", "baseball");
primaryKey.put("itemClass", "longbill");

// Exception handling omitted
tableH.multiDelete(primaryKey, null, null);
```

Hình 15: Câu lệnh xóa nhiều hàng

3.5 Đọc dữ liệu trên từng dòng

Việc đọc dữ liệu cũng thông qua việc sử dụng câu lệnh *TableAPI.get()*

```
PrimaryKey key = myTable.createPrimaryKey();
key.put("item", "Bolts");

// Retrieve the row. This performs a store read operation.
// Exception handling is skipped for this trivial example.
Row row = tableH.get(key, null);

// Now retrieve the individual fields from the row.
String item = row.get("item").asString().get();
String description = row.get("description").asString().get();
Integer count = row.get("count").asInteger().get();
Double percentage = row.get("percentage").asDouble().get();
```

Hình 16: Câu lệnh lấy dữ liệu từng hàng

Lấy dữ liệu nhiều hàng:

```
PrimaryKey key = myTable.createPrimaryKey();
key.put("itemType", "Hats");
key.put("itemCategory", "baseball");
key.put("itemClass", "longbill");

List<Row> myRows = null;

try {
    myRows = tableH.multiGet(key, null, null);
} catch (ConsistencyException ce) {
    // The consistency guarantee was not met
} catch (RequestTimeoutException re) {
    // The operation was not completed within the
    // timeout value
}
```

Hình 17: Câu lệnh lấy dữ liệu nhiều hàng

3.6 Đọc dữ liệu thông qua Indexes

Việc duyệt thẻ chứa index cũng tương tự như việc duyệt thẻ thường.

```
// Construct the IndexKey. The name we gave our index when
// we created it was 'DoB'.
Index dobIdx = myTable.getIndex("DoB");
IndexKey dobIdxKey = dobIdx.createIndexKey();

// Exception handling is omitted, but in production code
// ConsistencyException, RequestTimeException, and FaultException
// would have to be handled.
TableIterator<Row> iter = tableH.tableIterator(dobIdxKey, null, null);
try {
    while (iter.hasNext()) {
        Row row = iter.next();
        // Examine your row's fields here
    }
} finally {
    if (iter != null) {
        iter.close();
    }
}
```

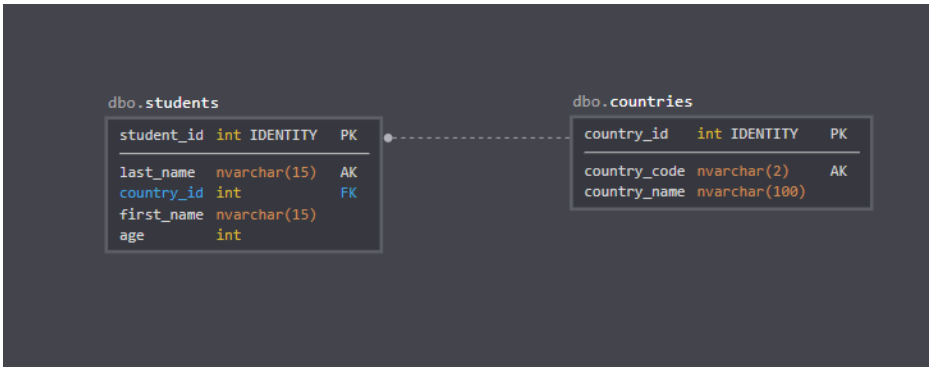
Hình 18: Câu lệnh lấy dữ liệu thông qua index

II. So sánh hiệu năng của Oracle NoSQL database so với DBMS truyền thống (MySQL) thông qua thực nghiệm

1. Mô hình cơ sở dữ liệu áp dụng trên MySQL và trên Oracle

Kết quả thu được sau khi test 20.000.000 bản ghi, trên Ubuntu server 4 core 16G Ram

- Mô hình dữ liệu test trên MySQL:



Hình 19: Mô hình dữ liệu test trên MySQL

1.1 Insert

Chèn dữ liệu vào trong bảng:

Bảng 2: Kết quả chèn dữ liệu trên MySQL và Oracle NoSQL

	MySQL	Oracle NoSQL
1 record	0,001s	0,001s
1.000 record	0,13s	0,34s
1.000.000 record	115s	178s
100.000.000 record	Khoảng 513p	Khoảng 400p

1.2 Delete

Xóa dữ liệu từ bảng:

Bảng 3: Kết quả xóa dữ liệu trên MySQL và Oracle NoSQL

NoSQL Query: delete from countries.students where student_id = 1 and country_id = 19

MySQL Query: delete from students where student_id = 10

MySQL	Oracle NoSQL
0,0001s	0,0005

1.3 Find

Đọc dữ liệu từ bảng:

Bảng 4: Kết quả đọc dữ liệu trên một bảng MySQL và Oracle NoSQL

*NoSQL Query: select * from countries.students where student_id = 1*

*MySQL Query: select * from students s where student_id = 1*

MySQL	Oracle NoSQL
0,000s	14s

Đọc dữ liệu sau khi join 2 bảng:

Bảng 5: Kết quả duyệt dữ liệu join 2 bảng trên MySQL và Oracle NoSQL

*NoSQL Query: select * from nested tables (countries.students s ancestors (countries c)) where s.student_id = 1*

MySQL Query: select s., c.* from students s inner join countries c on s.country_id = c.country_id and student_id = 1*

MySQL	Oracle NoSQL
0,001	15s

1.4 Update

Bảng 6: Kết quả thay đổi dữ liệu trên MySQL và Oracle NoSQL

NoSQL Query: update countries.students set age = 20 where student_id = 1 and country_id = 19;

MySQL Query: update students set age = 20 where student_id = 5

MySQL	Oracle NoSQL
0,001s	5s

C:\Users\Admin\Downloads\III.docx

Oracle NoSQL database là cơ sở dữ liệu sử dụng trên các hệ quản trị lớn có nhiều dữ liệu như hệ thống ngân hàng. Ngoài ra nhờ bộ mã nguồn hỗ trợ nhiều ngôn ngữ và bộ tài liệu chi tiết giúp cho việc triển khai vào các ứng dụng thực tế trở nên dễ dàng.

Phần 4. Tài liệu tham khảo

Getting Started with NoSQL Database Table Java Driver, *https://docs.oracle.com/en/database/other-databases/nosql-database/18.1/java-driver-table/preface.html#GUID-2C12DC44-6459-4190-93CB-5DC6C3A7C6DA*