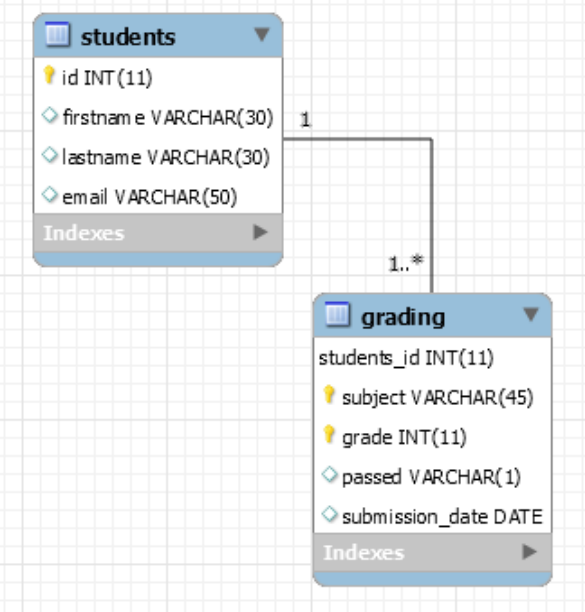
MySQL NDB Cluster Übungsaufgabe: CRUD-Operationen mit der NoSQL-API in Javascript

Voraussetzungen:

1. **MySQL NDB Cluster-Datenbank**
   Der Aufbau einer Cluster-Datenbank wird in der Hands-On Session in einer Docker-Umgebung zuvor erarbeitet.

2. **Node.js**
   Download: https://nodejs.org/de/download/
   Verwendete API: mysql-js (https://github.com/mysql/mysql-js/tree/master/database-jones)
   Installation der Module erfolgt in der Kommandozeile über den „npm“-Command:
   `npm install mysql`
   `npm install database-jones`

3. **Ausgangsbasis**
   Die CRUD-Operationen sollen anhand der Tabellen „students“ und „grading“ ausgeführt werden:



Als Ausgangsbasis für die Übungsaufgaben kann folgender Quellcode verwendet werden:

```
var nosql = require('database-jones');

var Student = function(firstname, lastname, email) {
    if (firstname) this.firstname = firstname;
    if (lastname) this.lastname = lastname;
    if (email) this.email = email;
};

var annotations = new nosql.TableMapping('students').applyToClass(Student);

var onSession = function(err, session) {
    // Implementierung der Übungsaufgaben
}
```

```
var dbProperties = {
    "implementation" : "mysql",
    "host" : "localhost",
    "database" : "kfru",
    "user" : "USERNAME",
    "password" : "PASSWORD"
};

console.log('Openning session');
// Datenbankverbindung herstellen
nosql.openSession(dbProperties, Student, onSession);
```

Übungsaufgaben:

1. **CREATE**
   - Erstelle einen Datensatz in der Tabelle „students“.
   - Erstelle einen oder mehrere Datensätze in der Tabelle „grading“ mit der gleichen ID.

2. **READ**
   - Lese den gerade angelegten Student über die Operation find() aus und gebe ihn in der Konsole aus.
   - Lese den gleichen Datensatz mithilfe der Query-API (https://github.com/mysql/mysql-js/blob/master/database-jones/API-documentation/Query) aus.
   - Nutze die APIs von TableMapping (https://github.com/mysql/mysql-js/blob/master/database-jones/API-documentation/TableMapping) und Projection (https://github.com/mysql/mysql-js/blob/master/database-jones/API-documentation/Projection) um alle Gradings des angelegten Students zu erhalten.

3. **UPDATE**
   Verändere einen/oder mehrere Werte des Datensatzes in „students“.

4. **DELETE**
   Lösche den Datensatz wieder.

Syntax für die CRUD-Operationen:

Operation	Node.js NoSQL-API (Javascript)
CREATE	session.persist(TableName, ObjectValues, [callback] …); session.persist(ObjectInstance, [callback] …);
READ	session.find(TableName, key, [callback] …); query.where([cond]); query.execute([queryParameters], [callback]…);
UPDATE	session.update(TableName, key, values, [callback] …); session.save(TableName, ObjectValues, [callback] …); session.save(ObjectInstance, [callback] …);
DELETE	session.remove(TableName, key, [callback]…); session.remove(ObjectInstance, [callback]…);

Lösungen:

1. **CREATE**

```
// INSERT
var onSession = function(err, session) {
    console.log('onSession.');
    if (err) {
    console.log('Error onSession.');
        console.log(err);
        process.exit(0);
    } else {
       var data = new Student('Thomas', 'Mueller', 'thomas.mueller@bayern.de');
       // 1. INSERT durch direkten Zugriff auf Tabelle
       session.persist('students', {'firstname' : 'Thomas', 'lastname' : 'Mueller'
, 'email' : 'thomas.mueller@bayern.de' }, onInsert, data, session);
       // 2. INSERT über TableMapping
       session.persist(data, onInsert, data, session);
    }
};
```

2. **READ**

2.1 READ über Primärschlüsselzugriff

```
var onFind = function(err, result) {
    console.log('onFind.');
    if (err) {
        console.log(err);
    } else {
        // Ausgabe des gelesenen Datensatzes
        console.log('Found: ' + JSON.stringify(result))
    }
    process.exit(0);
};

// READ
var onInsert = function(err, object, session) {
    console.log('onInsert.');
    if (err) {
        console.log(err);
    } else {
        console.log('Inserted: ' + JSON.stringify(object));
        // READ mit Zugriff über PrimaryKey
        session.find(Student, 1, onFind);
    }
};
```

2.2 READ über Query

```
nosql.openSession(dbProperties, Student).
then(function(session) {
    return session.createQuery(Student); // Tabelle übergeben
}).
then(function(query) {
    query.where(query.firstname.eq('Thomas').and(query.lastname.eq('Mueller')));
    return query.execute( { "order": 'asc' }); // Query ausführen
}).
then(console.log, console.trace).
then(nosql.closeAllOpenSessionFactories);
```

2.3 READ über TableMapping und Projection

```
var nosql = require('database-jones');

function Student() {}
function Grading() {}

var studentMapping = new nosql.TableMapping("students");
studentMapping.applyToClass(Student);

studentMapping.mapOneToMany(
    {
        fieldName: "id",
        target: Grading,
        targetField: "students_id"
    });

var gradingMapping = new nosql.TableMapping("grading");
gradingMapping.applyToClass(Grading);
gradingMapping.mapManyToOne(
     {
         fieldName: "students_id",
         target:    Student,
         targetField: "id",
         foreignKey: "fk_grading_students",
     });

var gradingProjection = new nosql.Projection(Grading);
gradingProjection.addFields(["subject", "grade", "passed", "submission_date"])
;

var studentProjection = new nosql.Projection(Student);
studentProjection.addRelationship("id", gradingProjection);
studentProjection.addFields(["firstname","lastname"]);

var dbProperties = {
    "implementation" : "mysql",
```

```
    "host" : "localhost",
    "database" : "kfru",
    "user" : "root",
    "password" : "root"
};

var key = 1;
nosql.openSession(dbProperties).
  then(function(session) {
    return session.find(studentProjection, key);
  }).
  then(console.log, console.trace).
  then(nosql.closeAllOpenSessionFactories);
```

3. **UPDATE**

```
var key = 1;

var onSession = function(err, session) {
    console.log('onSession.');
    if (err) {
    console.log('Error onSession.');
        console.log(err);
        process.exit(0);
    } else {
        // UPDATE 1 - direkter Zugriff auf Tabellenattribute
        session.update('students', key, { 'email' : 'thomasmueller@kfru.de' },
onUpdate, session);
        // UPDATE 2 - Zugriff über Tablemapping
        var changes = new Student('Thomas', 'Mueller', 'thomasmueller@kfru.de'
);
        session.update(Student, key, changes, onUpdate, session);
    }
};
```

4. **DELETE**

```
var key = 1;

var onDelete = function(err, result) {
    console.log('onDelete.');
    if (err) {
        console.log(err);
    } else {
        console.log('To be deleted: ' + JSON.stringify(result));
        console.log('Deleted entry with id=' + key);
    }
};
var onSession = function(err, session) {
    console.log('onSession.');
```

```
    if (err) {
    console.log('Error onSession.');
        console.log(err);
        process.exit(0);
    } else {
        session.find(Student, key, onDelete);
        session.remove(Student, key, onDelete);

    }
};
```