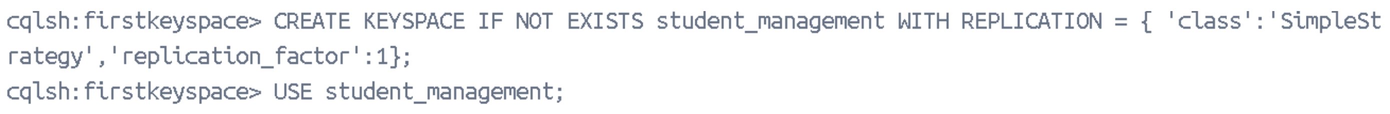
CSA3423-NoSQL
Implement a Student Management System Database in Cassandra
Cassandra Exp.3

Student Management System in Cassandra

1. Create Keyspace

```
CREATE KEYSPACE IF NOT EXISTS student_management

WITH REPLICATION = {

   'class': 'SimpleStrategy',

   'replication_factor': 1

};

USE student_management;
```



```
cqlsh:firstkeyspace> CREATE KEYSPACE IF NOT EXISTS student_management WITH REPLICATION = { 'class':'SimpleSt
rategy','replication_factor':1};
cqlsh:firstkeyspace> USE student_management;
```

2. Table Design

Cassandra is query-driven, so tables are created based on access patterns.
A typical student management system might need:

1. Store student details
2. Store course details
3. Store enrollment records
4. Retrieve students by course
5. Retrieve courses by student

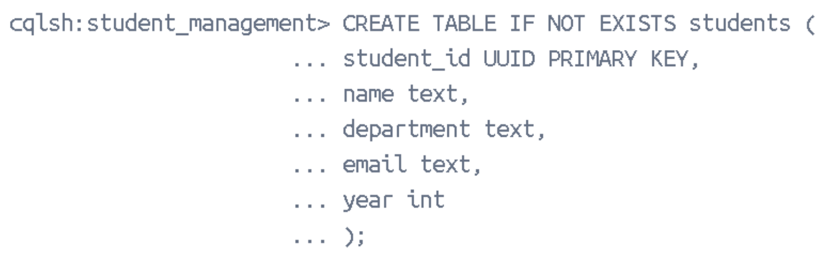
Below are the tables structured for these purposes.

3. Create Tables

3.1 Students Table

Stores basic student information.

```
CREATE TABLE IF NOT EXISTS students (

   student_id UUID PRIMARY KEY,

   name text,

   department text,

   email text,

   year int

);
```



```
cqlsh:student_management> CREATE TABLE IF NOT EXISTS students (
                      ... student_id UUID PRIMARY KEY,
                      ... name text,
                      ... department text,
                      ... email text,
                      ... year int
                      ... );
```

3.2 Courses Table

Stores course information.

CREATE TABLE IF NOT EXISTS courses (

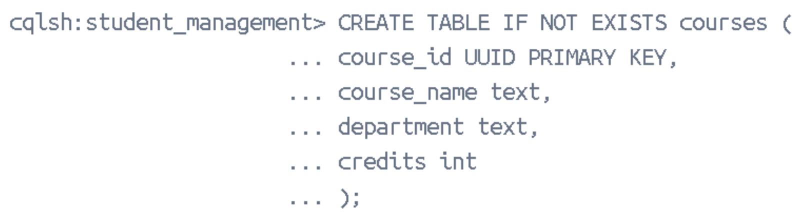
course_id UUID PRIMARY KEY,

course_name text,

department text,

credits int

);



```
cqlsh:student_management> CREATE TABLE IF NOT EXISTS courses (
                      ... course_id UUID PRIMARY KEY,
                      ... course_name text,
                      ... department text,
                      ... credits int
                      ... );
```

3.3 Enrollments by Student

Allows fetching all courses taken by a specific student.

CREATE TABLE IF NOT EXISTS enrollments_by_student (

student_id UUID,

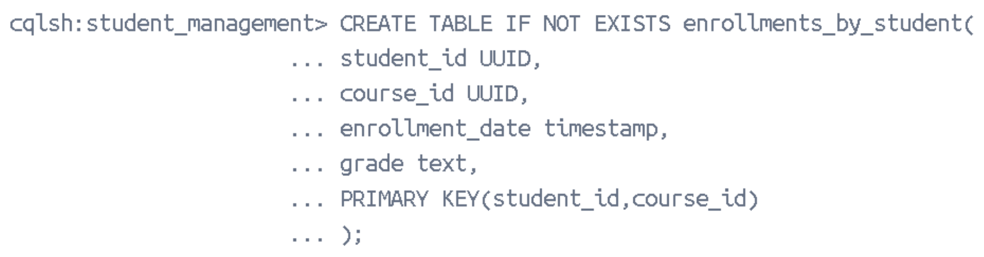
course_id UUID,

enrollment_date timestamp,

grade text,

PRIMARY KEY (student_id, course_id)

);



```
cqlsh:student_management> CREATE TABLE IF NOT EXISTS enrollments_by_student(
                      ... student_id UUID,
                      ... course_id UUID,
                      ... enrollment_date timestamp,
                      ... grade text,
                      ... PRIMARY KEY(student_id,course_id)
                      ... );
```

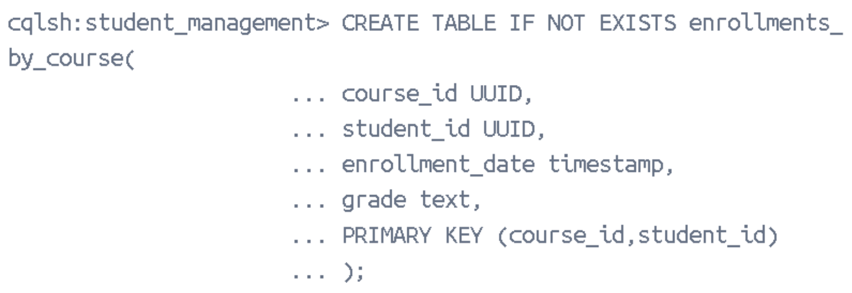
3.4 Enrollments by Course

Allows fetching all students enrolled in a specific course.

CREATE TABLE IF NOT EXISTS enrollments_by_course (

course_id UUID,

```
    student_id UUID,

    enrollment_date timestamp,

    grade text,

    PRIMARY KEY (course_id, student_id)

);
```

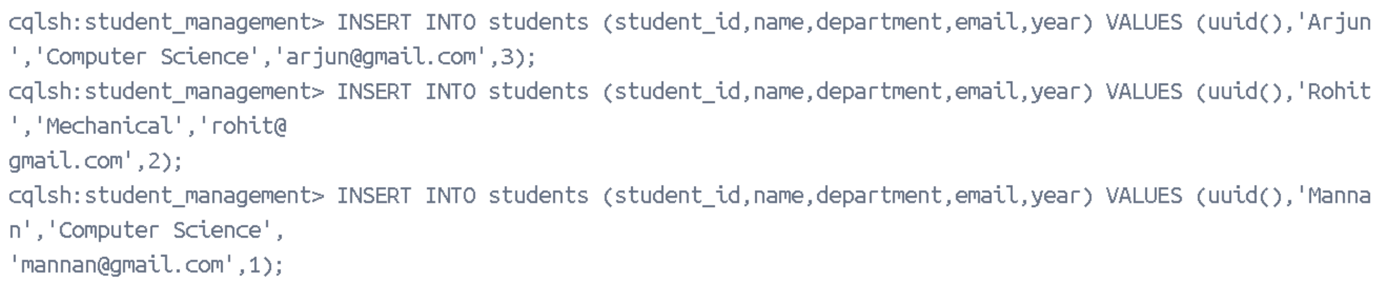


```
cqlsh:student_management> CREATE TABLE IF NOT EXISTS enrollments_
by_course(
                      ... course_id UUID,
                      ... student_id UUID,
                      ... enrollment_date timestamp,
                      ... grade text,
                      ... PRIMARY KEY (course_id,student_id)
                      ... );
```

4. Insert Sample Data

4.1 Insert Students

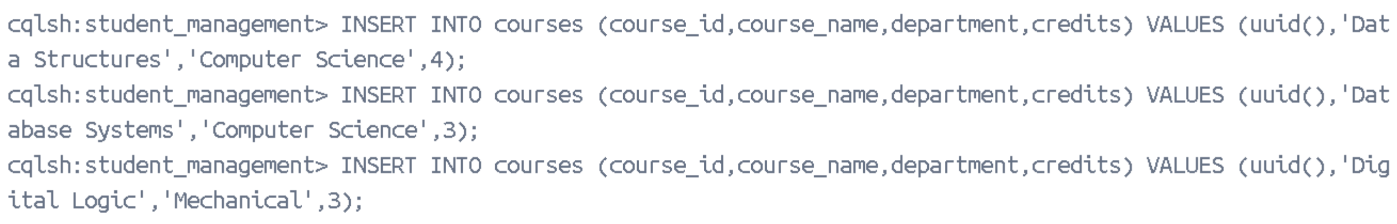
```
INSERT INTO students (student_id, name, department, email, year)

VALUES (uuid(), 'Arjun', 'Computer Science', 'arjun@example.com', 3);


INSERT INTO students (student_id, name, department, email, year)

VALUES (uuid(), 'Rohit', 'Mechanical', 'rohit@example.com', 2);


INSERT INTO students (student_id, name, department, email, year)

VALUES (uuid(), 'Mannan', 'Computer Science', 'mannan@example.com', 1);
```



```
cqlsh:student_management> INSERT INTO students (student_id,name,department,email,year) VALUES (uuid(),'Arjun
','Computer Science','arjun@gmail.com',3);
cqlsh:student_management> INSERT INTO students (student_id,name,department,email,year) VALUES (uuid(),'Rohit
','Mechanical','rohit@
gmail.com',2);
cqlsh:student_management> INSERT INTO students (student_id,name,department,email,year) VALUES (uuid(),'Manna
n','Computer Science',
'mannan@gmail.com',1);
```

4.2 Insert Courses

```
INSERT INTO courses (course_id, course_name, department, credits)

VALUES (uuid(), 'Data Structures', 'Computer Science', 4);


INSERT INTO courses (course_id, course_name, department, credits)

VALUES (uuid(), 'Database Systems', 'Computer Science', 3);


INSERT INTO courses (course_id, course_name, department, credits)

VALUES (uuid(), 'Digital Logic', 'Electronics', 4);
```


```
cqlsh:student_management> INSERT INTO courses (course_id,course_name,department,credits) VALUES (uuid(),'Dat
a Structures','Computer Science',4);
cqlsh:student_management> INSERT INTO courses (course_id,course_name,department,credits) VALUES (uuid(),'Dat
abase Systems','Computer Science',3);
cqlsh:student_management> INSERT INTO courses (course_id,course_name,department,credits) VALUES (uuid(),'Dig
ital Logic','Mechanical',3);
```

4.3 Insert Enrollment Records

Assume sample UUIDs will be used in a real setup.

INSERT INTO enrollments_by_student (student_id, course_id, enrollment_date, grade)

VALUES (student_uuid_1, course_uuid_1, toTimestamp(now()), 'A');

INSERT INTO enrollments_by_student (student_id, course_id, enrollment_date, grade)

VALUES (student_uuid_2, course_uuid_2, toTimestamp(now()), 'B');

INSERT INTO enrollments_by_course (course_id, student_id, enrollment_date, grade)

VALUES (course_uuid_1, student_uuid_1, toTimestamp(now()), 'A');

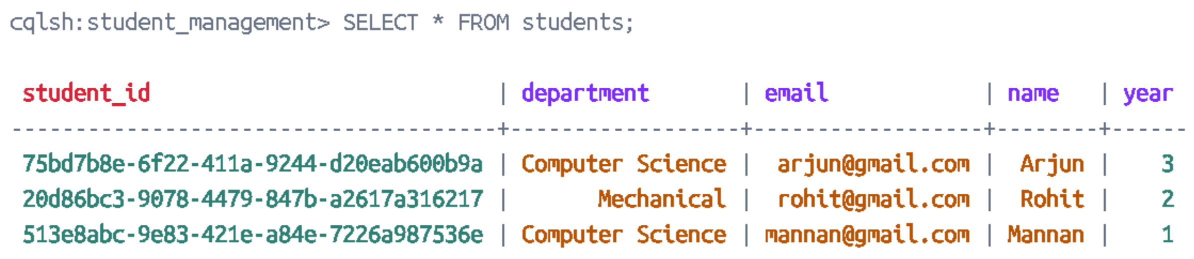
INSERT INTO enrollments_by_course (course_id, student_id, enrollment_date, grade)

VALUES (course_uuid_2, student_uuid_2, toTimestamp(now()), 'B');

5. Query Data

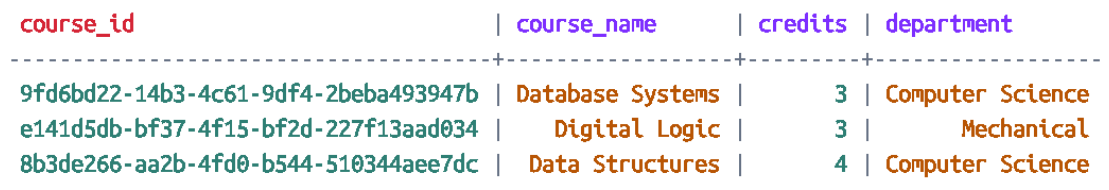
5.1 Get All Students

SELECT * FROM students;


```
cqlsh:student_management> SELECT * FROM students;

 student_id                           | department       | email             | name   | year
--------------------------------------+------------------+-------------------+--------+------
 75bd7b8e-6f22-411a-9244-d20eab600b9a | Computer Science |  arjun@gmail.com |  Arjun |    3
 20d86bc3-9078-4479-847b-a2617a316217 |       Mechanical |  rohit@gmail.com |  Rohit |    2
 513e8abc-9e83-421e-a84e-7226a987536e | Computer Science | mannan@gmail.com | Mannan |    1
```

5.2 Get All Courses

SELECT * FROM courses;


```
 course_id                            | course_name      | credits | department
--------------------------------------+------------------+---------+------------------
 9fd6bd22-14b3-4c61-9df4-2beba493947b | Database Systems |       3 | Computer Science
 e141d5db-bf37-4f15-bf2d-227f13aad034 |    Digital Logic |       3 |       Mechanical
 8b3de266-aa2b-4fd0-b544-510344aee7dc |  Data Structures |       4 | Computer Science
```

5.3 Get Courses Taken by a Student

(Uses enrollments_by_student)

```
SELECT * FROM enrollments_by_student

WHERE student_id = student_uuid_1;
```

5.4 Get Students Enrolled in a Course

(Uses enrollments_by_course)

```
SELECT * FROM enrollments_by_course

WHERE course_id = course_uuid_1;
```