**COMP 4332 Project – 2018 Spring**

[ Phase 2 ]

**NoSQL Database Design Report**

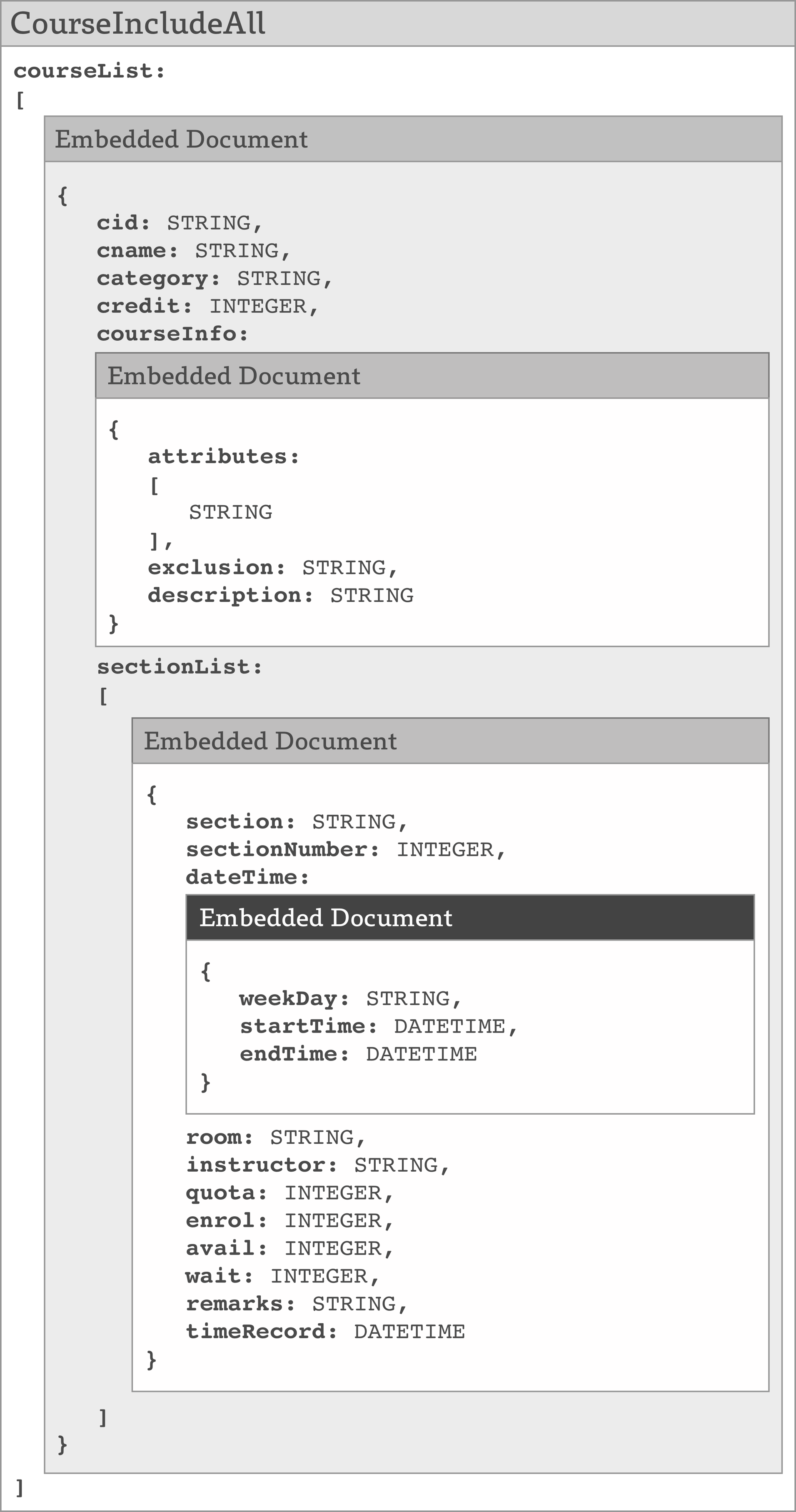
with Insertion and Query Explanation

Group 29

**Kim Ziwon (20216497)**

**Hu Yao-Chieh (20216239)**

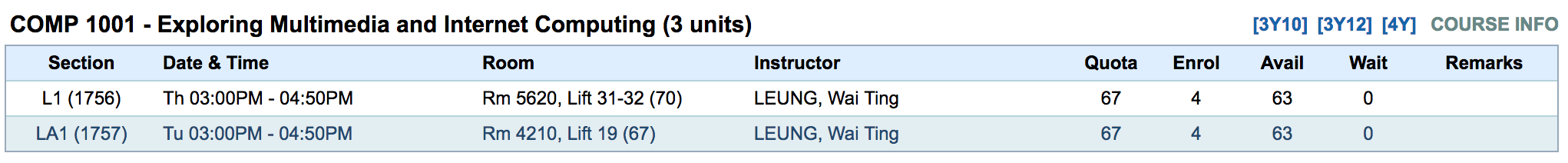
**Database Format and Design**



**Explanation on NoSQL Insertion Script**

The sample insertion is to insert information from the course webpage into the designed NoSQL database. The selected course is **COMP1001**:

Link: <http://comp4332.com/realistic/2017/Spring/01/25/09/00/subjects/COMP.html>



The script is shown below with detailed explanation in comments.

// Retrieve "university" database without modifying the db variable in shell

db = db.getSiblingDB("university")

// ################ Insertion ################ //

db.course.insert(

{

cid: "COMP 1001", // Course ID (STRING)

cname: "Exploring Multimedia and Internet Computing", // Course Name (STRING)

category: "COMP", // Course Category (STRING): To speed up categorized query in future usage

credit: 3, // Credit Unit (INTEGER)

courseInfo: // Course Information (Embedded Document): Stratified multiple items

{

attributes: // Common Core Attributes (Array of STRING): Can be empty

[

"Common Core (S&T) for 2010 & 2011 3Y programs",

"Common Core (S&T) for 2012 3Y programs",

"Common Core (S&T) for 4Y programs"

],

exclusion: "ISOM 2010, any COMP courses of 2000-level or above", // Exclusion and Description (STRING)

description: "This course is an introduction to computers and computing tools. It introduces the organization and basic working mechanism of a computer system, including the development of the trend of modern computer system. It covers the fundamentals of computer hardware design and software application development. The course emphasizes the application of the state-of-the-art software tools to solve problems and present solutions via a range of skills related to multimedia and internet computing tools such as internet, e-mail, WWW, webpage design, computer animation, spread sheet charts/figures, presentations with graphics and animations, etc. The course also covers business, accessibility, and relevant security issues in the use of computers and Internet."

},

sectionList: // Sections belonged to the course (Array of Embedded Document): Each section can have different section title (e.g. "L1", "LA1") and different timeSlot.

[

{

section: "L1", // Section Title (STRING)

sectionNumber: 1756, // Section Number (INTEGER)

dateTime: // Day and Time (Embedded Document): Contains info about weekly schedule

{

weekDay: "Thursday", // Weekday (STRING)

startTime: "03:00PM", // Start Time (STRING)

endTime: "04:50PM", // End TIme (STRING)

},

room: "Rm 5620, Lift 31-32 (70)", // Venue (STRING)

instructor: "LEUNG, Wai Ting", // Instructor (STRING): Can vary among sections within one course

quota: 67, // Quota Number (INTEGER)

enrol: 4, // Enroll Number (INTEGER)

avail: 63, // Available Number (INTEGER)

wait: 0, // Waitlist Number (INTEGER)

remarks: "", // Additional Remarks about enrollment (STRING)

timeSlot: new Date("2018-01-25T09:00:00") // Timeslot of the snapshot (ISODATE): In the format of YYYY-mm-ddTHH:MM:ss

},

{

section: "LA1",

sectionNumber: 1757,

dateTime:

{

weekDay: "Tuesday",

startTime: "03:00PM",

endTime: "04:50PM",

},

room: "Rm 4210, Lift 19 (67)",

instructor: "LEUNG, Wai Ting",

quota: 67,

enrol: 4,

avail: 63,

wait: 0,

remarks: "",

timeSlot: new Date("2018-01-25T09:00:00")

}

]

});

**Explanation on NoSQL Query Script**

The sample query has two major versions and few minor branches of implementation options that have been provided in the script (bolded ones are the chosen implementation):

**1. Query by Keyword**

1. Index all fields

**2. Index “cname” field**

**1. Index-based Search**

3. Regex-based Search

**2. Query by Waitlist Size**

**1) Query by Keyword**

// Retrieve "university" database without modifying the db variable in the shell.

db = db.getSiblingDB("university")

// ------------------ Query by Keyword --------------------- //

print(" -------------- Query by Keyword --------------- ")

// Creating a text index for all fields.

// db.course.createIndex({"$\*\*": "text"})

// Creating a text index for "cname" field. It can speed up keyword search.

db.course.createIndex({"cname": "text"})

// INDEX BASED SEARCH

indexAggregateSortedResult = db.course.aggregate( // Aggregation Pipeline that handles multiple mongo executions and returns a single result

[

{ $match: {$text: {$search: 'Exploring something'}}}, // MATCH: Filter out non-satisfied records and keep the records that contain the searched word (e.g. either "Exploring" or "something")

{ $unwind: '$sectionList'}, // UNWIND: Based on sectionList, unroll all the elements inside the sectionList as stand-alone records with all the corresponding fields and values as originals

{ $sort: {"cid": -1, "sectionList.section": 1}}, // SORT: Order the records by field "cid" decendingly first, and by field "sectionList.section" ascendingly afterward

{ $project : { \_id: 0, cid: 1, cname: 1, credit: 1, "sectionList.section": 1, "sectionList.dateTime": 1, "sectionList.quota": 1, "sectionList.enrol": 1, "sectionList.avail": 1, "sectionList.wait": 1 } }, // PROJECT: Discard \_id and keep all the following fields shown, if denoted as 1

{ $group: { // GROUP: Group the records by cid, and reorganize the fields as following

"\_id": "$cid", // cid is the pivot for grouping

"Course Code": {"$last": "$cid"}, // Find the latest

"Course Title": {"$last": "$cname"}, // Find the latest

"Number of Units/Credits": {"$last": "$credit"}, // Find the latest

"Section\_List": {"$push" :"$sectionList"}, // Push all the records to sectionList that turns out to be an array

}},

{$project: {\_id: 0}} // Discard \_id

]

);

// REGEX BASED SEARCH

// indexResult = db.course.find({"cname": {$regex: /Multimedi/}})

printjson(indexAggregateSortedResult.toArray()); // Show the query result in the format of JSON

**2) Query by Waitlist Size**

// -------------- Query by Waitlist Size --------------- //

print("\n\n\n -------------- Query by Waitlist Size --------------- ")

num\_f = 2; // The number 'f' as requested

start\_ts = ISODate("2018-01-24T00:00:00.000Z"); // The starting timestamp

end\_ts = ISODate("2018-01-29T00:00:00.000Z"); // The ending timestamp

latestTimeSlotResult = db.course.aggregate([

{ $unwind: '$sectionList'}, // UNWIND: unroll all the elements in sectionList, and each of which become a stand-alone records with all the corresponding fields and values as originals

{ $match: { // MATCH: Filter out all the unsatisfied records and keep the records that locate within the given time span ranging from start\_ts to end\_ts

"sectionList.timeSlot": {

$gte: start\_ts,

$lt: end\_ts

}

}},

// The filter that keeps the records that fulfill the condition that (waitlist size >= enrolment \* f)

// { $match: {$expr: {$gt: ["$sectionList.wait", {$multiply: ["$sectionList.enrol", num\_f]}]}}},

{ $group: { // GROUP: using cid as the pivot, find out the latest timestamp of each course

"\_id": "$cid",

"LatestTimeSlot": {

"$last": "$sectionList.timeSlot"

}

}}

]);

// Collect all the latest time slot for proceeding executions

// printjson(latestTimeSlotResult.toArray())

latestTimeSlot = latestTimeSlotResult.toArray()[0].LatestTimeSlot

result = db.course.aggregate([

{ $unwind: '$sectionList'}, // UNWIND: unroll all the elements in sectionList, and each of which become a stand-alone records with all the corresponding fields and values as originals

{ $match: {"sectionList.timeSlot": { // MATCH: Filter out all the unsatisfied records and keep the records that locate within the given time span ranging from start\_ts to end\_ts

$gte: start\_ts,

$lt: end\_ts

}}},

// The filter that keeps the records that fulfill the condition that (waitlist size >= enrolment \* f)

// { $match: {$expr: {$gt: ["$sectionList.wait", {$multiply: ["$sectionList.enrol", num\_f]}]}}},

{ $project : { // Discard \_id and keep the following fields if denoted as 1

\_id: 0,

cid: 1,

cname: 1,

credit: 1,

// The conditional statement to generate the value "Satisfied" for each section (not included in phase 2)

// "sectionList.Satisfied": {

// $cond: {

// if: {$gt: ["$sectionList.wait", {$multiply: ["$sectionList.enrol", num\_f]}]},

// then: "Yes",

// else: "No"

// }

// },

"sectionList.section": 1,

"sectionList.dateTime": 1,

"sectionList.quota": 1,

"sectionList.enrol": 1,

"sectionList.avail": 1,

"sectionList.wait": 1,

"sectionList.timeSlot": 1,

}

},

{ $sort: {"cid": -1, "sectionList.section": 1}}, // SORT: Order the cid decendingly and then section title ascendingly

{ $match: { "sectionList.timeSlot": latestTimeSlot }}, // MATCH: Filter that capture the latest timeslot for each section

{ $group: { // GROUP: Use \_id as pivot to group each course and show the additional fields as following

"\_id": "$cid",

"Course Code": {"$last": "$cid"}, // Find the latest

"Course Title": {"$last": "$cname"}, // Find the latest

"Number of Units/Credits": {"$last": "$credit"}, // Find the latest

"Matched Time Slot": {"$last": latestTimeSlot}, // Find the latest

"Section\_List": {"$push" :"$sectionList"}, // Push all the records to sectionList that turns out to be an array

}},

{$project: {\_id: 0}} // Discard \_id

]

);

printjson(result.toArray()); // Show the query result in the format of JSON

- This is the end of the report, thanks -