# CS 430: Assignment 6

Brandon Ingli

## Question 1

### Part a: University Database

#### Use Case 1: Course Review

**Use Case**: The CS department is reviewing the curriculum for their introduction course, CS 101. The department head wishes to retrieve the grades and majors of all students who took the class and their instructors since the last review, where the new curriculum took effect in 2017. The data will be used by another program to calculate a variety of statistics.

```
SELECT Grade_ltr, Major, Section_inst
FROM Student
JOIN Grade ON Student.SSN=Grade.SSN
WHERE Course_no="CS 101" AND Section_yr >= 2017;
```

#### Use Case 2: Transcript

**Use Case**: The University would like to allow a student to see their transcript via the secure web portal. A transcript includes a list of courses they've taken, when, their letter grade, and the number of quality points earned for the class. The SSN will be inserted into the query when the student logs in, and will be represented here as `${SSN}`.

```
SELECT Course_no, Section_sem, Section_yr, Grade_ltr,
    Credit_hours*Grade_no AS Quality_points
FROM Grade
JOIN Grade_Values on Grade.Grade_ltr=Grade_Values.Grade_ltr
JOIN Course ON Grade.Course_no=Course.Course_no
WHERE Grade.SSN="${SSN}"
ORDER BY Section_yr, Section_sem ASC;
```

#### Use Case 3: How Many Majors and Minors?

**Use Case**: The University would like a count of how many majors and minors each department is offering so that the figures can be displayed on the admissions department website.

```sql
SELECT Dept_name, COUNT(DISTINCT Major) AS No_majors, COUNT(DISTINCT Minor) AS No_minors
FROM Departments
JOIN Majors ON Majors.Dept_code=Departments.Dept_code
JOIN Minors ON Minors.Dept_code=Departments.Dept_code
GROUP BY Departments.Dept_code
ORDER BY No_majors, No_minors DESC;
```

## Part b: MLB Database

### Use Case 1: Tommy John Stats

**Use Case**: MLB would like a report on how many players per team have been treated with Tommy John Surgery, a common procedure for pitchers to repair the ligaments in their arm, in their history.

```sql
SELECT Player_team_name, COUNT(DISTINCT Player_SSN) As No_players
FROM Player
JOIN Injury ON Player.Player_SSN=Injury.Injured_player_SSN
WHERE Treatment="Tommy John Surgery"
GROUP BY Player_team_name;
```

### Use Case 2: Who Played on July 4?

**Use Case**: The MLB would like to give a bonus to the already overpayed players who played on the Fourth of July holiday this year, but they need a report of who to pay. They require the player's SSN, name and team name.

```sql
SELECT DISTINCT Player.Player_SSN, Player_name, Player_team_name
FROM Player
JOIN Player_in_game ON Player.Player_SSN=Player_in_game.Player_SSN
JOIN Game ON Game.Game_ID=Player_in_game.Game_ID
WHERE DATE(Game_timestamp)='2019-07-04';
```

### Use Case 3: Pay Per Game

**Use Case**: The MLB has now decided to pay players on a per-game basis instead of on a salary. They need to calculate how many games each player played in last year. The player name, SSN, team name, and number of games is required.

```sql
SELECT Player_name, Player.Player_SSN, Player_team_name, COUNT(DISTINCT Game_id) AS No_games
FROM Player
JOIN Player_in_game ON Player.Player_SSN=Player_in_game.Player_SSN
JOIN Game ON Game.Game_ID=Player_in_game.Game_ID
WHERE YEAR(Game_timestamp)=2018
GROUP BY Player.Player_SSN;
```

# Question 2

## Part a: University Database

This database will be comprised of 2 collections, one to hold student information and one to hold department information. Documents will be structured as follows.

**Student**

```
{
    ssn: "123456789",
    name: {
        first: "John",
        middle: "Q",
        last: "Smith"
    },
    sex: "M",
    birthday: "1970-01-01",
    class: 3,
    degree: "BS",
    addresses: [
        {
            street: "123 Main",
            city: "Kirksville",
            state: "MO",
            zip: "63501",
            type: "permanent"
        },
        ...
    ],
    phones: [
        {
            number: "1234567890",
            type: "mobile"
        },
        ...
    ],
    major: "Computer Science",
    minors: [
        "Data Science",
        "Photography"
    ]
    grades: [
        {
            course_no: "CS 101",
            course_name: "Intro to Computer Science",
            course_desc: "Learning to program stuff",
            course_level: 1,
```

```
            semester: "Fall",
            year: 2017,
            section_no: 2,
            instructor: "Doe",
            grade: "A",
            grade_no: 4,
            credits: 3
        },
        ...
    ]

}
```

**Department**

```
{
    code: "134",
    name: "Computer Science",
    office_no: "VH 2000",
    phone: "1123345566",
    college: "Science and Mathematics",
    majors: [
        "Computer Science"
    ],
    minors: [
        "Computer Science",
        "Cognitive Science",
        "Information Systems",
        "Mathematical Biology"
    ],
    courses: [
        {
            course_no: "CS 101",
            course_name: "Intro to Computer Science",
            course_desc: "Learning to program stuff",
            course_level: 1,
            credits: 3,
            sections: [
                {
                    semester: "Fall",
                    year: 2017,
                    section_no: 2,
                    instructor: "Doe"
                },
                ...
            ]
        }
    ]
}
```

This is a good fit for the use cases above since all relevant information is grouped together, reducing the number of database accesses. All information needed in these use cases is on a per-student or per-department basis, so grouping the data into these two collections makes sense. There is a small amount of duplicate information regarding course and section information, but that is OK in MongoDB, and may actually help in this case. A course will retain its original info within a student's grade even if the University adopts a new catalog and changes information such as credit hours or description.

#### Use Case 1: Course Review

```
db.students.find({grades.course_no: "CS 101", grades.year: {$gte: 2017}},
        {grades.grade: true, grades.instructor: true, major: true, _id: false});
```

#### Use Case 2: Transcript

```
db.students.find({ssn: "${SSN}"},
        {grades: true, _id: false}).sort({grades.year: 1});
```

#### Use Case 3: How Many Majors and Minors?

```
db.departments.aggregate([
    {
        $project: {
            name: true,
            no_majors: {$size: "$majors"},
            no_minors: {$size: "$minors"}
        }
    }
]).sort({no_majors: -1});
```

### Part b: MLB Database

This database will be comprised of one collection. Documents will take the following form.

**mlb**

```
{
    name: "Yadier Molina",
    ssn: "123456789",
    dob: "1982-07-13",
    number: 4,
    type: "player",
    positions: [
        "Catcher"
    ],
    team: {
```

```
        city: "St. Louis",
        name: "Cardinals"
    },
    injuries: [
        {
            date_on: "2019-05-15",
            date_off: "2019-05-25",
            desc: "left thumb fracture",
            treatment: "ice"
        },
        ...
    ],
    games_played: [
        {
            start_date: "2019-07-04 14:05",
            home: {
                name: "Mariners",
                score: 4
            },
            away: {
                name: "Cardinals",
                score: 5
            },
            innings: 9
        }
    ]
}
```

This is a good fit for the use cases since all relevant information is grouped together in the same document in a collection. Most information is on a per-player basis, so grouping all relevant information with the player information makes sense. There is redundant information about game information, but for our use case, that is ok, and shouldn't be an update issue since game information does not change once inserted. Additionally, we do not need to list all games from a team, for example, so having the game information spread out doesn't affect our use cases. Coaches and players share the same structure, so there is no use putting them in separate collections or to use separate document structures.

### Use Case 1: Tommy John Stats

```
db.mlb.find({injuries.treatment: "Tommy John Surgery", type: "player"},
        {_id: false, team.name: true, name: true}).aggregate([
            {$sortByCount: "$team.name"}
        ])
```

### Use Case 2: Who Played on July 4?

```
db.mlb.find({games_played.start_date: {$regex: "2019-07-04"}}.
            {_id: false, ssn: true, name: true, team.name: true});
```

### Use Case 3: Pay Per Game

```
db.mlb.aggregate([
    $project: {
        _id: false,
        name: true,
        ssn: true,
        team.name: true,
        no_games: {
            $size: {
                $filter: {
                    input: "$games_played",
                    as: "gp",
                    cond: {$regex: "2018"}
                }
            }
        }
    }
]);
```