Sophia Thomas
Darrell Clark
Lucas Gonzalez
Sean Hia

MongoDB Design

Database Design Example | Rubric

Parent Table Name	Child Table Name	Multiplicity	Direction	Design Pattern	Comments	Rationale
departments	courses	1..1 to 0..*	Parent to Child	One-to-Many	Create an array of the courses number and name in departments	We redundantly store the name and number of the course into departments because the name comes up frequently in queries and the number is the surrogate key
departments	courses	1..1 to 0..*	Child to Parent	Two way referencing	In the Courses collection, embed the abbreviation of departments	Abbreviation is the column that is most frequently queried in courses due to the combination of course number and department abbreviation uniquely identifying a course; abbreviation isn't likely to change
departments	majors	1..1 to 1..*	Parent to Child	One-to-Many	Create an array of major names in departments	One to few would prohibit us from utilizing Major as a stand alone entity. Which is necessary for the student major
departments	majors	1..1 to 1..*	Child to Parent	Two way referencing	Majors collection embeds department abbreviation from departments	The department abbreviation is the column that comes up in queries for majors, thus redundantly storing the abbreviation will save time
courses	sections	1..1 to 0..*	Parent to Child	One-to-Many	Courses has multiple	Abbreviation and

					references to sections; stores array of sectionID or sectionNumber	course_number directly relate to a specific course, which can link to a specific section
courses	sections	1..1 to 0..*	Child to Parent	Two way referencing	Sections embeds to the course number and department abbreviation from courses	In order to uniquely identify the section we need the course number and department abbreviation fields; resulting in more efficient queries since we won't have to search for the information
students	studentmajor	1..1 to 0..*	Parent to Child	One-to-Few	Students contains an array of student major embedded documents	Students are limited to/will only have at most 4 majors and even that's extreme
majors	studentmajor	1..1 to 0..*	Child to Parent	Denormalization from One to Many	Each element in the student major array within students has the major name	Major name frequently comes up in queries; storing the name will save time
majors	studentmajor	1..1 to 0..*	Parent to Child	One-to-Squillion	Majors has no details of the students that are declared in that major; so studentmajor contains a reference to the major(s) a student has declared	At least a thousand students are declared for any given major. The array of student majors would exceed the bound of the document size
students	sections	0..* to 0..*	Student to Section	Denormalization Many to One	Students has an array called "sections" that contains the sectionID, department abbreviation, course, semester, section number, and year columns	These fields are frequently queried to ensure uniqueness the sections that students are enrolled in; although this will redundantly store data in students it will speed up lookups
students	enrollment	1..1 to 0..*	Parent to Child	Denormalization from Many to One	The array "sections" within Students embeds the enrollment category (passfail/ lettergrade) and their attribute	The enrollment category is often queried when searching for the section a student is enrolled in thus this will save time at the

						cost of redundant data
sections	enrollment	1..1 to 0..*	Parent to Child	One to Few	Array of enrollments will be embedded into sections	Each section will have at most hundreds of students enrolled, which will keep the sections document within the size limit of 16MB
enrollment	passfail	1..1 to 0..1	Parent to Child	One to Few oneOf	Passfail is embedded into the enrollment document; using the keyword oneOf to ensure an enrollment can either be passfail or lettergrade, but not both	Each enrollment will contain zero or one instance of passfail, therefore embedding it into enrollment is the most optimal approach in terms of queries and updates
enrollment	lettergrade	1..1 to 0..1	Parent to Child	One to Few oneOf	Lettergrade is embedded into the enrollment document; using the keyword oneOf	Just like passfail, Each enrollment contains zero or one instance of lettergrade, therefore embedding it into enrollment is the most optimal approach