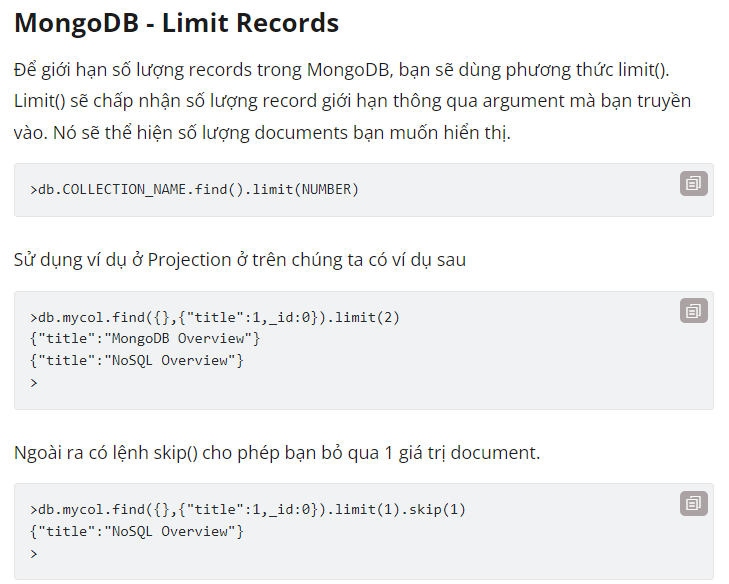
**TRUY VẤN TRONG MONGODB**

**Graphical user interface, text, application

Description automatically generated**

**Table

Description automatically generated**

****

**Graphical user interface, text

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

Text search Query:

https://www.geeksforgeeks.org/mongodb-regex/

* Cách sử dụng $unwind trong aggregate:
  + Unwind Array:

Giả sử ta có collection có tên inventory, chứa một doc:

**"\_id"** : 1, **"item"** : **"ABC1"**, sizes: [ **"S"**, **"M"**, **"L"**] }

Khi sử dụng câu truy vấn:

db.inventory.aggregate( [ { $unwind : **"$sizes"** } ] )

Kết quả trả về:

|  |
| --- |
| { **"\_id"** : 1, **"item"** : **"ABC1"**, **"sizes"** : **"S"** } |
| { **"\_id"** : 1, **"item"** : **"ABC1"**, **"sizes"** : **"M"** } |
| { **"\_id"** : 1, **"item"** : **"ABC1"**, **"sizes"** : **"L"** } |

Mỗi doc giống doc ban đầu ngoại trừ giá trị của trường sizes giữ một giá trị của mảng sizes ban đầu

Missing or Non-array Values:

Giả sử ta có collection clothing chứa các doc:

|  |
| --- |
| { **"\_id"** : 1, **"item"** : **"Shirt"**, **"sizes"**: [ **"S"**, **"M"**, **"L"**] }, |
| { **"\_id"** : 2, **"item"** : **"Shorts"**, **"sizes"** : [ ] }, |
| { **"\_id"** : 3, **"item"** : **"Hat"**, **"sizes"**: **"M"** }, |
| { **"\_id"** : 4, **"item"** : **"Gloves"** }, |
| { **"\_id"** : 5, **"item"** : **"Scarf"**, **"sizes"** : null } |

Thực hiện $unwind mảng sizes:

db.clothing.aggregate( [ { $unwind: { path: **"$sizes"** } } ] )

Kết quả trả về:

|  |
| --- |
| { \_id: 1, item: **'Shirt'**, sizes: **'S'** }, |
| { \_id: 1, item: **'Shirt'**, sizes: **'M'** }, |
| { \_id: 1, item: **'Shirt'**, sizes: **'L'** }, |
| { \_id: 3, item: **'Hat'**, sizes: **'M'** } |

Giải thích:

Trong doc \_id: 1, mảng sizes là một populated. $unwind trả về mỗi giá trị trong mảng

Tương tự với \_id: 3

Nhưng đối với \_id: 2, 4, 5 không trả về bất cứ giá trị nào vì mảng sizes không thể tách ra thanh các giá trị đơn (\_id: 2 có sizes rỗng, \_id: 4 không có sizes, \_id:5 sizes = null)

NOTE:

{path: <FIELD>} là cú pháp tùy chọn, 2 câu truy vấn đưới đây là như nhau:

|  |
| --- |
| db.clothing.aggregate( [ { $unwind: **"$sizes"** } ] ) |
| db.clothing.aggregate( [ { $unwind: { path: **"$sizes"** } } ] ) |

$addToSet:

* Thêm một giá trị vào một mảng trừ khi giá trị đó đã có sẵn
* Form:

{ $addToSet: { <field1>: <value1>, ... } }

* + <field> có thể là 1 trường trong doc, hoặc trong array, sử dụng dot notation (toán tử .)
* Behavior:
  + $addToSet chỉ đảm bảo rằng không có mục trùng lặp nào được thêm vào tập hợp và không ảnh hưởng đến các phần tử trùng lặp hiện có.
  + $addToSet không đảm bảo thứ tự cụ thể của các phần tử trong tập hợp đã sửa đổi
* Missing Field: Nếu sử dụng $addToSet trên một trường không có tài liệu để cập nhật, $addToSet sẽ tạo trường mảng với giá trị được chỉ định là phần tử của nó
* Nếu sử dụng $addToSet cho một trường không phải MẢNG, toán tử này sẽ lỗi
  + Ví dụ: ta có collection pigments chứa các giá trị:

\_id: 1, colors: **"blue, green, red"**

Colors không phải là một mảng, nếu dùng $addToSet sẽ bị lỗi

|  |
| --- |
| db.pigments.updateOne( |
| { \_id: 1 }, |
| { $addToSet: { colors: **"mauve"** } } |
| ) |

Giá trị khi được thêm vào mảng:

Ta có collection alphabet với 1 doc:

\_id: 1, letters: [**"a"**, **"b"**] }

Thực hiện update mảng letters:

A picture containing graphical user interface

Description automatically generated

Mảng letters sau khi được update sẽ trở thanh:



Tip: để add từng giá trị vào mảng, sử dụng $each chỉnh sửa trong $addToSet



Dot Notation:

MongoDB sử dụng dot notation để truy cập tới các phần tử của mảng và truy suât tới các trường được nhứng trong doc

*https://www.mongodb.com/docs/manual/core/document/#std-label-document-dot-notation*

Xóa một mảng trong một mảng:

Text

Description automatically generated

Các cách xóa element from array:

[***https://www.delftstack.com/howto/mongodb/mongodb-remove-element-from-array/***](https://www.delftstack.com/howto/mongodb/mongodb-remove-element-from-array/)

* Xóa mảng trong mảng dùng $pullAll:

**Text

Description automatically generated**

* Xóa mảng trong mảng dùng $pop:
  + Dùng $pop nếu muốn xóa phần tử đầu tiên hoặc cuối cùng khỏi một mảng. Set 1 hoặc -1 để xóa phần tử cuối mảng hoặc đầu mảng

**Text

Description automatically generated**

* Xóa phần tử trong mảng dùng $unset:
  + Dùng toán tử này auto xóa trường chứa phần tử muốn xóa

**→** không dùng được trong trường hợp muốn xóa phần tử trong mảng

Text

Description automatically generated

multi: true 🡪 cập nhật hoặc áp dụng truy vấn cho nhiều bản ghi cùng 1 lúc

ví dụ:

[***https://www.tutorialspoint.com/how-to-update-multiple-documents-in-mongodb***](https://www.tutorialspoint.com/how-to-update-multiple-documents-in-mongodb)

$skip (aggregation):

Bỏ qua số lượng tài liệu được chỉ định chuyển đến stage và chuyển các docs còn lại sang giai đoạn tiếp theo trong pipeline



$skip nhận vào một số nguyên dương xác định số lượng tài liệu tối đa cần bỏ qua  
$skip không ảnh hưởng tới nội dung docs mà nó chuyển qua pipelin

$toUpper, $toLower (aggregation)

Text

Description automatically generated

$lookup (aggregation)

Giống join left trong sql

Graphical user interface, text, application, email

Description automatically generated

* From: collection trong cùng database muốn join với bảng .aggregate hiện tại
* localField: trường trong docs của collection .aggregate
* foreignField: trường ứng với trường trong docs của collection .aggregate muốn match giá trị
* as: tên của mảng trường mới được thêm vào input docs. Mảng trường mới này chứa các docs matching với collection form. Nếu tên đã tồn tại trong input docs rồi thì trường này sẽ overwriten (ghi đè)

Join có điều kiện và truy vấn trên collection sau khi join

Để thực hiện các truy vấn con có tương quan và không tương quan với 2 collection và thực hiện các điều kiện két hợp khác ngoai một kết hợp binh đẳng duy nhất, dãy sử dụng cú pháp $lookup sau:

Graphical user interface, text, application, email

Description automatically generated

* from: collection trong cùng một database muốn thực hiện join
  + từ mongoDB 5.1, from collection có thể được chia nhỏ
* let: biến tùy chọn, có thể được truy cập bằng các stage trong pipeline, bao gồm các $lookup stages bổ sung được lồng trong pipeline
  + Để tham chiếu đén các biesn trong pipeline stage, hãy sử dụng cú pháp “$$<variable>”
  + Stage $match yêu cầu sử dụng toán tử $expr để truy cập các biến
    - Toán tử $expr cho phép sử dụng cá biểu thức tổng hợp bên trong cú pháp $match
    - Bắt đầu từ Mongodb 5.0 các toán tử so sanh $eq, $lt, $lte, $gt và gte được đặt trong một toán tử $expr có thể sử dụng một index trên from collection tham chiếu trong stage $lookup. Hạn chế:
      * Multikey indexes không được sử dụng
      * Các index không được sử dụng để so sanh khi toán hạng là một mảng hoặc một toán hoạn không được xác định
      * Các index không được sử dụng để so sanh với nhiều hơn 1 trường trong đường dẫn toán hạng
  + Các stage khác (không khớp với $) trong quy trinh không yêu cầu toán tử $expr để truy cập các biến
* Pipeline:
  + Chỉ định đường dẫ để chạy trên collection đã join. Pipeline xác định các tài liệu từ colelction được join. Để trả lại tất cả docs, hãy chỉ định một pipeline trống []
  + Pipeline không thể bao gồm $out stage hoặc $merge stage. Bắt đầu từ phiên bản 6.0, pipeline có thể chứa Atlas Search $saearch stage như một stage đầu tiên bên trong pipeline. Để hiểu hơn, xem Atlas Search Support
  + Pipeline không thể truy nhập trực tieps vào các trường docs đã tham gia. Thay vào đó, hãy xác định các biến cho các trường tài liệu được két hợp bằn cách sử dụng tùy chọn let và sau đó tham chiesu đén các stage của pipeline

Embedded docs in mongodb:

* Trong CSDL quan hệ, chúng ta lưu trữ từng thực thể riêng lẻ ở trong mỗi bảng riêng, và kết nối chúng thông qua khoa ngoại. Trong khi đó MongoDB hỗ trợ tham chiếu từ docs này sang docs khác và thậm chí join nhiều docs với nhau.
* Khi nào thì sử dụng embedded hơn là ref:
  + Docs được embedded là một cách hiệu quả và rõ ràng để lưu trữ dữ liệu liên quan, đặc biệt là dữ liệu thường xuyên được truy cập cùng nhau. Nói chung, khi thiết kế lược dododf cho MongoDB, cta thường nhúng theo mặc định và chỉ khi sử dụng tham chiếu và kết nối phía ứng dụng hoặc phía CSDL. Một khối lượng công việc nhất định có thể truy xuất một docs duy nhất và có tất cả dữ liệu cần thiết thường xuyên hơn, thì ứng dụng của cta sẽ có hiệu suất cao nhất quán.

Embedded doc Pattern

* Embedded subset pattern:
* Extend reference pattern:
* Trong mối quan hệ 1:1, cách tốt nhất để design Schema là embedded
* Một collection trong mongodb không thể phát triển hơn 16Mb

Mô hình tinh Bucket pattern

[***https://viblo.asia/p/mongodb-co-ban-phan-1-l5XRBVN3RqPe***](https://viblo.asia/p/mongodb-co-ban-phan-1-l5XRBVN3RqPe)

Cách sử dụng $regex:

<https://www.statology.org/mongodb-like-query/>

* Seasons: Thông tin các mùa giải
  + Name: tên mùa giải
  + Year: năm diễn ra
  + conpetitionCategory: chứa thông tin các hạng mục thi đấu
  + seasonGroups: nhóm mùa giải
* players: Thông tin các cầu thủ
  + clubs: [] - một cầu thủ có thể thuộc nhiều clb
  + isDeleted: đã bị xóa – boolean
  + firstName: họ
  + lastName: tên đệm
  + dob: ngày tháng năm sinh
  + academicScores: các điểm học vấn
    - GPA
    - PSAT
    - SAT
    - ACT
  + graduationYear: {} – năm tốt nghiệp
  + nationaslities: một cầu thủ có thể đến từ nhiều quốc gia []
    - name: tên quốc gia
    - flag: cờ - scr image
  + picture: thông tin ảnh
  + position: vị trí chơi
  + squadNumber: số đội chơi cho
  + team: thông tin đội bóng
  + bio: mô tả thông tin cá nhân

→ Hầu hết các documents không chứa đầy đủ các thông tin như trên, các trường có thể tính toán mà có ở tât cả cacs doc là dob

***→ Thống kê tuổi tới hiện tại của mỗi cầu thủ***

***→ Liệt kê cách cầu thủ có tuổi >= 15***

* teams: chứa thông tin các đội bóng
  + name: tên đội bóng
  + category: chứa thông tin thể loại thi đấu
  + club: thông tin clb
  + seasons: [] thông tin các mùa giải
  + competition: [] thông tin các cuộc thi

***→ truy vấn các đội bóng thuộc clb SC Wave***

* categories: chứa thông tin các thể loại thi đấu
  + name: tên thể loại
* Club\_private\_signup\_links:
* Clubs: chứa thông tin của các clb:
  + Name: tên clb
  + Logo: src logo của clb
  + Conference: [] hội nghị
  + Team: [] chứa thông tin các đội bóng thuộc clb
  + Organisations: [] mảng chứa danh sách các tổ chức
  + homeColor: màu sân nhà
  + awayColor: …
  + homeVenueName: tên sân nhà
* competitioncategories: chứa thông tin các hạng mục thi đấu
  + name: tên hạng mục thi đấu
  + competition: chứa thông tin cuộc thi
  + seasons: [] chứa thông tin các mùa giải
  + category: chứa thông tin hạng mục thi đấu
* competitions: chứa thông tin các cuộc thi
  + name: tên cuộc thi
  + competititonType: loại hình thi đấu
  + organization: thông tin tổ chức
  + conferences: []
  + competitonCategories: [] thông tin các hạng mục tranh đấu
* conferences: chứa thông tin đơn vị tổ chức
  + name: tên tổ chức
  + organization: chứa thông tin tổ chức
* defaulrostermembers: chứa thông tin thành viên thuộc danh sách mặc định
  + isDeleted:
  + player: thông tin cầu thủ
  + squadNumber:
  + team: thông tin đội bóng
* fixtures: chứa thông tin đồ đạc
  + category: thể loại
  + fixtureDate: thời gian cấp đồ

Mối quan hệ giữa các collection:

* Một club có nhiều team, một team chỉ thuộc 1 club

→ Mối quan hệ giữa club và team là 1:n

* Quan hệ giữa teams và seasons là 1:n
* Quan hệ giữa seasons và competitioncategories: 1:n
* Quan hệ giữa teams và defaulrostermember: 1: n
* Tại sao dùng ref: model mà không dùng ref: “tên model”

Text

Description automatically generated

* Tại sao dùng $addFileds như trên lại bị lỗi, mà cho vào $project thì k?
* Có cách nào populate trong aggregate k?