**Appendix P**

**Working with MongoDB**

|  |
| --- |
| **NOTE**  Most of this appendix, and all of the end of appendix problems, require the use of the Ch14\_FACT.json file. The Appendix provides instructions on how to import this file into a MongoDB database as a collection. The documents in this file is a reduced version of the data from the Ch07\_FACT database used in Chapter 7.  It can be helpful to draw to students’ attention that this is a reduced data set compared to Chapter 7. The reason this is a reduced set is because it is limited to a specific intended application. Refer students back to Chapter 14, Big Data and NoSQL, to remind them that document databases like MongoDB are aggregate aware. Therefore, the data are organized into documents with a great deal of redundancy across documents, but in a manner that reduces the number of documents that need to be accessed during the processing of a transaction. |

**Answers to Review Questions**

**1. What is the difference between a replacement update and an operator update in MongoDB?**

In MongoDB a replacement update will replace the entire document being updated. If the existing document has key:value pairs that are not included in the update command, then those pairs are lost. Only the pairs specified in the update command will exist in the replaced version of the document. With an operator update, the existing document is unchanged except for the changes specified in the update command. Pairs not included in the update command are not affected.

**2. Explain what an upsert does.**

Upsert is a combination insert / update. If an existing document is found that matches the criteria given, then an update is performed on that document using the key:value pairs specified in the command. If an existing document is not found that matches the criteria given, then an insert is performed to create a document with the key:value pairs specified in the command.

**3. Describe the difference between using $push and $addToSet in MongoDB.**

Both commands are used to add a value to an array. The $push command will always add the value to the array, even it results in duplicate values in the array. The $addToSet command will only add the value to the array if adding it does not result in duplicate values in the array.

**4. Explain the functions used to enable pagination of results in MongoDB.**

Results can be provided in pages of information by using limit() and skip() functions. The limit() function specifies how many results to return. The skip() function allows the programmer to provide an offset of documents before the limit is applied.

**5. Explain the difference in processing when using an explicit *and* and an implicit *and* with MongoDB.**

With both forms of logical *and* the DBMS must apply criteria to a document to determine if the document should be included in the results. An explicit *and*, using the $and operator, will determine that a document should not be included and stop applying criteria to that document as soon as one of the criteria evaluates to FALSE for that document. An implicit *and* will apply all criteria to the document before determining if the document should be included or not in the results. As a result, explicit *and* tends to perform better in most cases.

**Problem Solutions**

**For the following set of problems, use the *fact* database and *patron* collection created in the text for use with MongoDB.**

**1. Create a new document in the patron collection. The document should satisfy the following requirements:**

**First name is “Rachel”**

**Last name is “Cunningham”**

**Display name is “Rachel Cunningham”**

**Patron type is student**

**Rachel is 24 years old**

**Rachel has never checked out a book**

**Be certain to use the same keys as already exist in the collection. Be certain capitalization is consistent with the documents already in the collection. Do not store any keys that do not have a value (in other words, no NULLs).**

db.patron.insert(

{ fname: "Rachel",

lname: "Cunningham",

display: "Rachel Cunningham",

type: "student",

age: 24

}

);

**2. Modify the document entered in the previous question with the following data. Do not replace the current document.**

**Rachel has checked out two books on January 25, 2018.**

**The id of the first checkout is “95000”**

**The first book checked out was book number 5237**

**Book 5237 is titled “Mastering the database environment”**

**Book 5237 was published in 2015 and is in the “database” subject**

**The id of the second checkout is “95001”**

**The second book checked out was book number 5240**

**Book 5240 is titled “iOS Programming”**

**Book 5240 was published in 2015 and is in the “programming” subject**

**Use the same keys as already exist within the collection. Conform to the existing documents in terms of capitalization.**

db.patron.update(

{ "\_id": ObjectId("5a45c23f395ff183e78d9c17")},

{ $set: {checkouts: [

{ “id”: "95000",

“year”: "2018",

“month”: "1",

“day”: "25",

“book": 5237",

"title": "Mastering the database environment",

“pubyear”: "2015",

“subject”: "database"

},

{ "id": "95001",

"year": "2018",

"month": "1",

"day": "25",

“book”: "5240",

“title”: "iOS Programming",

“pubyear”: "2015",

“subject”: "Programming"

}

]

}

}

)

**3. Write a query to retrieve the \_id, display name and age of students that have checked out a book in the cloud subject.**

db.patron.find({"checkouts.subject":"cloud"},

{display:1, age:1})

**4. Write a query to retrieve only the first name, last name, and type of faculty patrons that have checked out at least one book with the subject “programming”.**

db.patron.find({type: "faculty", "checkouts.subject":"programming"},

{fname:1, lname:1, type:1, \_id:0})

**5. Write a query to retrieve the documents of patrons that are faculty and checked out book 5235, or that are students under the age of 30 that have checked out book 5240. Display the documents in a readable format.**

db.patron.find({$or: [

{type: "faculty", "checkouts.book":"5235"},

{type: "student", "checkouts.book":5240, age: {$lt:30}}

]

}

).pretty()

**6. Write a query to display the only the first name, last name, and age of students that are between the ages of 22 and 26.**

db.patron.find({ type:"student",

$and: [{age: {$gte:22}},

{age: {$lte:26}}

]

},

{fname:1, lname:1, age:1, \_id:0}

)