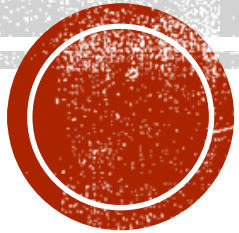
NOSQL UNIT-5

Dr. SELVA KUMAR S

B.M.S COLLEGE OF ENGINEERING



AGENDA

- Php and MongoDB
- Python and MongoDB
- Creating Blog application with PHP
- NoSQL Database Administration

PHP AND MONGODB

- PHP

JSON

```
$contact = array(
"First Name" => "Philip",
"Last Name" => "Moran",
"Address" => array(
"Street" => "681 Hinkle Lake Road",
"Place" => "Newton",
"Postal Code" => "MA 02160",
"Country" => "USA"
)
,
"E-Mail" => array(
"pm@example.com",  "pm@office.com" ,"philip@example.com",
"philip@office.com", "moran@example.com", "moran@office.com",
"pmoran@example.com", "pmoran@office.com"
),
"Phone" => "617-546-8428",
"Age" => 60
);
```

```
contact = ( {
"First Name" : "Philip",
"Last Name" : "Moran",
"Address" : [
{
"Street" : "681 Hinkle Lake Road",
"Place" : "Newton",
"Postal Code" : "MA 02160",
"Country" : "USA"
}
],
"E-Mail" : [
"pm@example.com",
"pm@office.com",
"philip@example.com",
"philip@office.com",
"moran@example.com",
"moran@office.com",
"pmoran@example.com",
"pmoran@office.com"
],
"Phone" : "617-546-8428",
"Age" : 60
})
```

MONGODB CLASSES

- The PHP driver for MongoDB contains four core classes:
- Mongo:
  - Connect()
  - Close()
  - listDBs()
  - selectDBs()
  - selectCollection()

MONGODB CLASSES

- MongoDB:
  - createCollection()
  - selectCollection()
  - createDBRef()
  - getDBRef()
  - drop()and
  - getGridFS().

MONGODB CLASSES

- MongoCollection:
  - count()
  - find()
  - findOne()
  - insert()
  - remove()
  - save(), and
  - update().

MONGODB CLASSES

- MongoCursor:
  - getNext()
  - count()
  - hint()
  - limit()
  - skip(), and
  - sort().

CONNECTING AND DISCONNECTING

- // Connect to the database
- $c = new Mongo();
- // Select the database you want to connect to, e.g *contacts*
- $c->contacts;

- // Connect to the database
- $c = new Mongo();
- // Select the database you want to connect to, e.g. *contacts*
- $c->selectDB("contacts");

CONNECTING AND DISCONNECTING

- // Connect to the database
- $c = new Mongo();
- // Selecting the database ('contacts') and collection ('people') you want
- // to connect to
- $c->contacts->people;

- // Connect to the database
- $c = new Mongo();
- // Selecting the database (‘contacts’) and collection (‘people’) you want
- // to connect to
- $c-> selectDB("contacts")->selectCollection("people ");

CONNECTING AND DISCONNECTING

- // Connecting to the database
- $c = new Mongo();
- // Listing the available databases
- print_r($c->listDBs());

- // Connecting to the database
- $c = new Mongo();
- // Listing the available collections within the 'contacts' database
- print_r($c->contacts->listCollections());

INSERTING DATA

```
$contact = array(
"First Name" => "Philip",
"Last Name" => "Moran",
"Address" => array(
"Street" => "681 Hinkle Lake Road",
"Place" => "Newton",
"Postal Code" => "MA 02160",
"Country" => "USA"
)
,
"E-Mail" => array(
"pm@example.com",
"pm@office.com",
"philip@example.com",
"philip@office.com"
),
"Phone" => "617-546-8428",
"Age" => 60
);
```

```
// Connect to the database
$c = new Mongo();
// Select the collection 'people'
$collection = $c->contacts->people;
// Insert the document '$contact' into the people collection '$collection'
$collection->insert($contact);
```

LISTING DATA

- Returning a Single Document
- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Find the very first document within the collection, and print it out
- // using print_r
- print_r($collection->findOne());

LISTING DATA

- Returning all Document
- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find();
- // For each document it finds within the collection, print its contents
- while ($document = $cursor->getNext())
- {
- print_r($document);
- }

QUERYING

- $c = new Mongo();
- // Select the collection ‘people’ from the database 'contacts'
- $collection = $c->contacts->people;
- // Use dot notation to search for a document in which the place is set to "Newton"
- $address = array("Address.Place" => "Newton");
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find($address);
- // For each document it finds within the collection, print the ID and its contents
- while ($document = $cursor->getNext())
- {
- print_r($document);
- }

QUERYING

- // Connect to the database
- $c = new Mongo();
- // Select the collection ‘people’ from the database 'contacts'
- $collection = $c->contacts->people;
- // Define the e-mail address you want to search for under $email
- $email = array("E-Mail" => "vw@example.com");
- // Find the very first person in the collection matching the e-mail address
- print_r($collection->findOne($email));

SORT

- // Connect to the database
- $c = new Mongo();
- // Select the collection ‘people’ from the database 'contacts'
- $collection = $c->contacts->people;
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find();
- // Use the sort command to sort all results in $cursor, based on their age
- $cursor->sort(array('Age' => 1));
- // Print the results
- while($document = $cursor->getNext())
- {
- print_r($document);
- }

LIMIT

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find();
- // Use the limit function to limit the number of results to 1
- $cursor->limit(1);
- //Print the result
- while($document = $cursor->getNext())
- {
- print_r($document);
- }

SKIP

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find();
- // Use the skip function to skip the first result found
- $cursor->skip(1);
- // Print the result
- while($document = $cursor->getNext())
- {
- print_r($document);
- }

COUNT

- // Connect to the database
- $c = new Mongo();
- // Select the collection ‘people’ from the database ‘contacts’
- $collection = $c->contacts->people;
- // Specify the search parameters
- $country = array("Address.Country" => "USA");
- // Execute the query and store under the $cursor variable for further processing
- $cursor = $collection->find($country);
- // Count the results and return the value
- print_r($cursor->count());

MAP REDUCE

- // Connect to the database
- $c = new Mongo();
- // Specify the database in which to work
- $db = $c->contacts;
- // Execute the mapreduce function via the command() function
- $db->command(array(
- "mapreduce" => "people",
- "map" => $map,
- "reduce" => $reduce,
- "out" => "countries"
-));

SPECIFYING INDEX WITH HINT

```
// Connect to the database
$c = new Mongo();
// Select the collection 'people' from the database 'contacts'
$collection = $c->contacts->people;
// Execute the query and store it under the $cursor variable
$cursor = $collection->find(array("Last Name" => "Moran"));
// Use the hint function to specify which index to use
$cursor->hint(array("Last Name" => -1));
//Print the result
while($document = $cursor->getNext())
{
print_r($document);
}
```

QUERIES WITH CONDITIONAL OPERATOR

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the conditional operator
- $cond = array('Age' => array('$lt' => 30));
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find($cond);
- //Print the results
- while($document = $cursor->getNext())
- {
- print_r($document);
- }

$LTE

- // Connect to the database
- $c = new Mongo();
- // Select the collection ‘people’ from the database ‘contacts’
- $collection = $c->contacts->people;
- // Specify the conditional operator
- $cond = array('Age' => array('$lte' => 60));
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find($cond);
- //Print the results
- while($document = $cursor->getNext())
- {
- print_r($document);
- }

$NE

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the conditional operator
- $cond = array('Age' => array('$ne' => 28));
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find($cond);
- //Print the results
- while($document = $cursor->getNext())
- {
- print_r($document);
- }

$IN

- // Connect to the database
- $c = new Mongo();
- // Select the collection ‘people’ from the database ‘contacts’
- $collection = $c->contacts->people;
- // Specify the conditional operator
- $cond = array('Address.Country' => array('$in' => array("USA","UK")));
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find($cond);
- //Print the results
- while($document = $cursor->getNext())
- {
- print_r($document);
- }

$ALL

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the conditional operator
- $cond = array('E-Mail' => array('$all' => array("vw@example.com","vw@office.com")));
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find($cond);
- //Print the results
- while($document = $cursor->getNext())
- {
- print_r($document);
- }

GET LAST THREE RECORD

```
// Connect to the database
$c = new Mongo();
// Select the collection 'people' from the database 'contacts'
$collection = $c->contacts->people;
// Specify our search operator
$query = array("Last Name" => "Moran");
// Specify the conditional operator
$cond = (object)array('E-Mail' => array('$slice' => -3));
// Execute the query and store it under the $cursor variable
$cursor = $collection->find($query, $cond);
// For each document it finds within the collection, print its contents
while ($document = $cursor->getNext())
{
print_r($document);
}
```

SKIP FIRST 2 RECORD AND LIMIT 3 RESULTS

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify our search operator
- $query = array("Last Name" => "Moran");
- // Specify the conditional operator
- $cond = (object)array('E-Mail' => array('$slice' => [2, 3]));
- // Execute the query and store it under the $cursor variable
- $cursor = $collection->find($query, $cond);
- // For each document it finds within the collection, print its contents
- while ($document = $cursor->getNext())
- {
- print_r($document);
- }

RETURN CONTACT DO NOT HAVE AGE FIELD

```
// Connect to the database
$c = new Mongo();
// Select the collection 'people' from the database 'contacts'
$collection = $c->contacts->people;
// Specify the conditional operator
$cond = array('Age' => array('$exists' => false));
// Execute the query and store it under the $cursor variable
$cursor = $collection->find($cond);
//Print the results
while($document = $cursor->getNext())
{
print_r($document);
}
```

```
// Connect to the database
$c = new Mongo();
// Select the collection 'people' from the database 'contacts'
$collection = $c->contacts->people;
// Specify the conditional operator
$cond = array(
"Address.Country" => "USA",
'$or' => array(
array("Last Name" => "Moran"),
array("E-Mail" => "vw@example.com")
)
);
// Execute the query and store it under the $cursor variable
$cursor = $collection->find($cond);
//Print the results
while($document = $cursor->getNext())
{
print_r($document);
}
```

REGULAR EXPRESSION

```
// Connect to the database
$c = new Mongo();
// Select the collection 'people' from the database 'contacts'
$collection = $c->contacts->people;
// Specify the Regular Expression
$regex = new MongoRegex("/stradgynl/i");
// Execute the query and store it under the $cursor variable
$cursor = $collection->find(array("Address.Place" => $regex));
//Print the results
while($document = $cursor->getNext())
{   print_r($document);    }
```

MODIFYING DATA WITH PHP

- Update():
- *upsert*: If set to true, this Boolean option causes a new document to be created if the search criteria is not matched.
- *multiple*: If set to true, this Boolean option causes all documents matching the search criteria to be updated.
- *safe*: If set to true, this option instructs the application to wait for a database response to confirm whether an update was successful. If it wasn't successful, an exception is thrown. If the safe option is set to an integer *n*, then it will replicate the update to *n* number of machines before returning success.
- *fsync*: If set to true, this Boolean option causes the data to be synced to disk before returning a success. If this option is set to true, then it's implied that safe is also set to true, even if it's set to false.

UPDATE EXAMPE

```
// Connect to the database
$c = new Mongo();
// Select the collection 'people' from the database 'contacts'
$collection = $c->contacts->people;
// Specify the search criteria
$criteria = array("Last Name" => "Wood");
// Specify the information to be changed
$update = array( "First Name" => "Vicky", "Last Name" => "Wood",
"Address" => array( "Street" => "50 Ash lane", "Place" => "Ystradgynlais",
"Postal Code" => "SA9 6XS", "Country" => "UK" ) ,
"E-Mail" => array( "vw@example.com", "vw@office.com"),
"Phone" => "078-8727-8049",
"Age" => 28
);
// Options
$options = array("upsert" => true);
// Perform the update
$collection->update($criteria,$update,$options);
// Show the result
print_r($collection->findOne($criteria));
```

INCREASING THE VALUE OF A SPECIFIC KEY WITH $INC

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Search for anyone that’s younger than 40
- $criteria = array("Age" => array('$lt' => 40));
- // Use $inc to increase their age by 3 years
- $update = array('$inc' => array('Age' => 3));
- // Options
- $options = array("upsert" => true);
- // Perform the update
- $collection->update($criteria,$update,$options);

CHANGING THE VALUE OF A KEY WITH $SET

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the search criteria
- $criteria = array("Last Name" => "Wood");
- // Specify the information to be changed
- $update = array('$set' => array("First Name" => "Vicky"));
- // Options
- $options = array("upsert" => true);
- // Perform the update
- $collection->update($criteria,$update,$options);

$SET TO ADD FIELD TO ALL THE OCCURRENCES

- // Connect to the database
- $c = new Mongo();
- // Select the collection ‘people’ from the database ‘contacts’
- $collection = $c->contacts->people;
- // Specify the search criteria using Regular Expressions
- $criteria = array("E-Mail" => new MongoRegex("/@office.com/i"));
- // Add “Category => Work” into every occurrence found
- $update = array('$set' => array('Category' => 'Work'));
- // Options
- $options = array('upsert' => true, 'multi' => true);
- // Perform the upsert via save()
- $collection->update($criteria,$update,$options);

DELETING A FIELD WITH $UNSET

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the search criteria
- $criteria = array("Last Name" => "Wood");
- // Specify the information to be removed
- $update = array('$unset' => array("Phone" => 1));
- // Perform the update
- $collection->update($criteria,$update);

APPEND A VALUE $PUSH

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the search criteria
- $criteria = array("Last Name" => "Wood");
- // Specify the information to be added
- $update = array('$push' => array("E-Mail" => "vw@mongo.db"));
- // Perform the update
- $collection->update($criteria,$update);

ADDING MULTIPLE VALUES $PUSHALL

- // Connect to the database
- $c = new Mongo();
- // Select the collection ‘people’ from the database ‘contacts’
- $collection = $c->contacts->people;
- // Specify the search criteria
- $criteria = array("Last Name" => "Wood");
- // Specify the information to be added
- $update = array(
- '$pushAll' => array(
- "E-Mail" => array("vicwo@mongo.db","vicwo@example.com")));
- // Perform the update
- $collection->update($criteria,$update);

ADDING DATA TO AN ARRAY $ADDTOSET

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the search criteria
- $criteria = array("Last Name" => "Wood");
- // Specify the information to be added (successful because it doesn’t exist yet)
- $update = array('$addToSet' => array("E-Mail" => "vicwo@example.com"));
- // Perform the update
- $collection->update($criteria,$update);

$ADDTOSET

```
// Connect to the database
$c = new Mongo();
// Select the collection 'people' from the database 'contacts'
$collection = $c->contacts->people;
// Specify the search criteria
$criteria = array("Last Name" => "Wood");
// Specify the information to be added (partially successful, some
// examples were already there)
$update = array(
'$addToSet' => array
(
"E-Mail" => array
(
'$each' => array
(
"vw@mongo.db",
"vicky@mongo.db",
"vicky@example.com" ) ) ) );
// Perform the update
$collection->update($criteria,$update);
```

REMOVING AN ELEMENT FROM AN ARRAY $POP

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the search criteria
- $criteria = array("Last Name" => "Wood");
- // Pop out the first e-mail address found in the list
- $update = array('$pop' => array("E-Mail" => -1));
- // Perform the update
- $collection->update($criteria,$update);

REMOVING EACH OCCURRENCE OF A VALUE $PULL

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the search criteria
- $criteria = array("Last Name" => "Wood");
- // Pull out each occurrence of the e-mail address "vicky@example.com"
- $update = array('$pull' => array("E-Mail" => "vicky@example.com"));
- // Perform the update
- $collection->update($criteria,$update);

REMOVING EACH OCCURRENCE OF MULTIPLE ELEMENTS

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the search criteria
- $criteria = array("Last Name" => "Wood");
- // Pull out each occurrence of the e-mail addresses below
- $update = array(
- '$pullAll' => array(
- "E-Mail" => array("vw@mongo.db","vw@office.com")));
- // Perform the update
- $collection->update($criteria,$update);

UPSERTING DATA WITH SAVE()

```
// Specify the document to be saved
$contact = array(
"First Name" => "Kenji",
"Last Name" => "Kitahara",
"Address" => array(
"Street" => "149 Bartlett Avenue",
"Place" => "Southfield",
"Postal Code" => "MI 48075",
"Country" => "USA"
),
"E-Mail" => array( "kk@example.com", "kk@office.com"
),
"Phone" => "248-510-1562", "Age" => 34
);
// Connect to the database
$c = new Mongo();
// Select the collection 'people'
$collection = $c->contacts->people;
// Save via the save() function
$collection->save($contact);
// Realizing you forgot something, let's upsert this contact:
$contact['Category'] = 'Work';
// Perform the upsert
$collection->save($contact);
```

MODIFYING A DOCUMENT AUTOMATICALLY

- Findandmodify()
- Parameters:
- query: Specifies a filter for the query.
- sort: Sorts the matching documents in a specified order.
- remove: If set to true, the first matching document will be removed.
- update: Specifies the information to update the document.
- new: If set to true, returns the updated document, rather than the selected document.
- fields: Specifies the fields you would like to see returned, rather than the entire document.
- upsert: If set to true, performs an upsert.

EXAMPLE FINDANDMODIFY()

- // Connect to the database
- $c = new Mongo();
- // Specify the database in which to work
- $db = $c->contacts;
- // Perform a findAndModify()
- print_r($db->command(
- array(
- "findandmodify" => "people",
- "query" => array("Last Name" => "Kitahara"),
- "update" => array('$push' => array("E-Mail" => "kitahara@mongo.db"))
-)));

EXAMPLE REMOVE SORT PARAMETER

- // Connect to the database
- $c = new Mongo();
- // Specify the database in which to work
- $db = $c->contacts;
- // Perform a findAndModify()
- print_r($db->command(
- array(
- "findandmodify" => "people",
- "query" => array("Category" => "Work"),
- "sort" => array("Age" => -1),
- "remove" => true
-)));

DELETING DATA REMOVE()

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the search criteria
- $criteria = array("Last Name" => "Wood");
- // Specify the options
- $options = array('justOne' => true, 'safe' => true);
- // Perform the removal
- $collection->remove($criteria,$options);

REMOVES MULTIPLE DOCUMENTS

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $collection = $c->contacts->people;
- // Specify the search criteria using Regular Expressions
- $criteria = array("E-Mail" => new MongoRegex("/@office.com/i"));
- // Specify the options
- $options = array("justOne" => false);
- // Perform the removal
- $collection->remove($criteria,$options);

DROP ENTIRE COLLECTION

- // Connect to the database
- $c = new Mongo();
- // Select the collection to remove
- $collection = $c->contacts->people;
- // Remove the collection and return the results
- print_r($collection->drop());

DROP ENTIRE DATABASE

- // Connect to the database
- $c = new Mongo();
- // Select the database to remove
- $db = $c->contacts;
- // Remove the database and return the results
- print_r($db->drop());

DBREF

- DBRef enables you to create links between two different documents stored in different locations;
- DBRef is used to create reference automatically.
- There are two ways to perform this operation
- 1. Manual Referencing
- 2. DBRef to create link automatically.

MANUAL REFERENCING

```
// Connect to the database
$c = new Mongo();
$db = $c->contacts;
// Select the collections we want to store our contacts and addresses in
$people = $db->people;
$addresses = $db->addresses;
// Specify an address:
$address = array( "Street" => "St. Annastraat 44", "Place" => "Monster",
"Postal Code" => "2681 SR", "Country" => "Netherlands"
);
// Save the address
$addresses->insert($address);

// Add a contact living at the address
$contact = array( "First Name" => "Melvyn", "Last Name" => "Babel", "Age" => 35,
"Address" => $address['_id']
);
$people->insert($contact);
```

DBREF CREATE()

```
// Connect to the database
$c = new Mongo();
$db = $c->contacts;
// Select the collections we want to store our contacts and addresses in
$people = $db->people;
$addresses = $db->addresses;
// Specify an address:
$address = array("Street" => "WA Visser het Hooftlaan 2621", "Place" => "Driebergen",
"Postal Code" => "3972 SR", "Country" => "Netherlands" );
// Save the address
$addresses->insert($address);
// Create a reference to the address
$addressRef = MongoDBRef::create($addresses->getName(), $address['_id']);
// Add a contact living at the address
$contact = array( "First Name" => "Ivo", "Last Name" => "Lauw", "Age" => 24,
"Address" => $addressRef
);
$people->insert($contact);
```

RETRIEVING THE INFORMATION GET()

- // Connect to the database
- $c = new Mongo();
- // Select the collection 'people' from the database 'contacts'
- $people = $c->contacts->people;
- // Define the search parameters
- $lastname = array("Last Name" => "Lauw");
- // Find our contact, and store under the $person variable
- $person = $people->findOne(array("Last Name" => "Lauw"));
- // Dereference the address
- $address = MongoDBRef::get($people->db, $person['Address']);
- // Print out the address from the matching contact
- print_r($address);

GRIDFS

- PHP driver classes to work with GridFS:
- MongoGridFS: Stores and retrieves files from the database.
  - delete(), find(), storeUpload()
- MongoGridFSFile: Works on a specific file in the database.
  - __construct(), getFilename(), getSize(), and write().
- MongoGridFSCursor: Works on the cursor.
  - __construct(), current(), getNext(), and key().

STORING FILES STOREUPLOAD()

- // Connect to the database
- $c = new Mongo();
- // Select the name of the database
- $db = $c->contacts;
- // Define the GridFS class to ensure we can handle the files
- $gridFS = $db->getGridFS();
- // Specify the parameters of the filename to be stored (optional)
- $name = $_FILES['File']['name'];
- // Upload the file into the database
- $id = $gridFS->storeUpload('File',$name);

RETRIEVING FILES GETFILENAME()

- // Connect to the database
- $c = new Mongo();
- $db = $c->contacts;
- // Initialize GridFS
- $gridFS = $db->getGridFS();
- // Find all files in the GridFS storage and store under the $cursor parameter
- $cursor = $gridFS->find();
- // Return all the names of the files
- foreach ($cursor as $object) {
- echo "Filename:".$object->getFilename();
- }

DELETING DATA

- // Connect to the database
- $c = new Mongo();
- // Specify the database name
- $db = $c->contacts;
- // Initialize GridFS
- $gridFS = $db->getGridFS();
- // Specify the file via it’s ID
- $id = new MongoId('4c555c70be909680010800000');
- $file = $gridFS->findOne(array('_id' => $id));
- // Remove the file using the remove() function
- $gridFS->delete($id);

PYTHON AND MONGODB

- MongoDB uses BSON-styled documents, and PHP uses associative arrays.
- In a similar vein, Python has what it calls *dictionaries*

```
item = {
"Type" : "Laptop",
"ItemNumber" : "1234EXD",
"Status" : "In use",
"Location" : {
"Department" : "Development",
"Building" : "2B",
"Floor" : 12,
"Desk" : 120101,
"Owner" : "Anderson, Thomas"
},
"Tags" : ["Laptop","Development","In Use"]
}
```

USING PYMONGO MODULES

- Connecting and Disconnecting Database
- **>>> import pymongo**
- **>>> from pymongo import Connection**
- **>>> c = Connection()**
- **>>> db = c.inventory**
- **>>> db**
- Database(Connection('localhost', 27017), u'inventory’)
- **>>> collection = db.items**

INSERTING DATA

- **>>> item = {**
- **... "Type" : "Laptop",**
- **... "ItemNumber" : "1234EXD",**
- **... "Status" : "In use", ... "Location" : { ... "Department" : "Development", ... "Building" : "2B",**
- **... "Floor" : 12, ... "Desk" : 120101, ... "Owner" : "Anderson, Thomas"**
- **... },**
- **... "Tags" : ["Laptop","Development","In Use"]**
- **... }**
- **>>> collection.insert(item)**
- ObjectId('4c57207b4abffe0e0c000000')

FINDING DATA

- **Finding a Single Document:**
- **>>> collection.find_one()**
- **>>> collection.find_one({"ItemNumber" : "3456TFS"})**

- **Finding Multiple Document:**
- **>>> for doc in collection.find():**
- **... Doc**
- **>>> for doc in collection.find({"Location.Department" : "Development"}):**
- **... doc**

QUERIES WITH SORT, LIMIT AND SKIP

- >>> for doc in collection.find ({"Status" : "In use"},
- ... {"ItemNumber":"true", "Location.Owner" : "True"})
- ... .sort("ItemNumber"):
- ... Doc
- >>> for doc in collection.find({}, {"ItemNumber" : "true"}).limit(2):
- ... Doc
- >>> for doc in collection.find({}, {"ItemNumber" : "true"}).skip(2):
- ... Doc
- >>> for doc in collection.find({'Status' : 'In use'},
- ... {'ItemNumber':'true', 'Location.Owner':'true'})
- ... .limit(2).skip(1).sort("ItemNumber"):
- ... doc

AGGREGATING QUERIES

- Count():
- **>>> collection.find({}).count()**
- **>>> collection.find({"Status" : "In use", "Location.Owner" : "Walker, Jan"}).count()**
- Distinct():
- **>>> collection.distinct("ItemNumber")**

GROUPING DATA WITH MAP REDUCE

- >>> from pymongo.code import Code
- >>> map = Code("function() {"
- ... "this.Tags.forEach(function(t) {"
- ... " emit(t, 1);"
- ... "});"
- ... "}")
- >>> reduce = Code("function (key, values) {"
- ... " var Total = 0;"
- ... " for (var i = 0; i < values.length; i++) {"
- ... " Total += values[i];"
- ... " }"
- ... " return Total;"
- ... "}")

MAP_REDUCE

- >>> result = collection.map_reduce(map, reduce)
- >>> for tag in result.find():
- ... Tag

- >>> result = collection.map_reduce(map, reduce, query={"Status" : "In use"}, limit=4,
- out="Tags")
- >>> for tag in result.find():
- ... tag

SPECIFYING AN INDEX WITH HINT()

- >>> from pymongo import ASCENDING
- >>> collection.create_index([("ItemNumber", ASCENDING)])

- >>> for doc in collection.find({"Location.Owner" : "Walker, Jan"}) .hint([("ItemNumber",
- ASCENDING)]):
- ... doc

QUERIES WITH CONDITIONAL OPERATORS

- >>> for doc in collection.find({"Location.Desk" : {"$lt" : 120102} }):
- ... Doc
- >>> for doc in collection.find({"Location.Desk" : {"$gt" : 120102} }):
- ... Doc
- >>> for doc in collection.find({"Location.Desk" : {"$lte" : 120102} }):
- ... Doc
- >>> for doc in collection.find({"Location.Desk" : {"$gte" : 120102} }):
- ... Doc
- >>> collection.find({"Status" : {"$ne" : "In use"}}).count()

ARRAY OF MATCHES WITH $IN

- >>> for doc in collection.find({"Tags" : {"$in" : ["Not used","Development"]}} ,
- {"ItemNumber":"true"}).limit(2):
- ... Doc
- >>> for doc in collection.find({"Tags" : {"$nin" : ["Development"]}}, {"ItemNumber":"true"}):
- ... Doc
- >>> for doc in collection.find({"Tags" : {"$all" : ["Storage","Not used"]}},
- {"ItemNumber":"true"}):
- ... Doc
- >>> for doc in collection.find({ "Location.Building" : "2B", "$or" : [{ "Location.Department"
- : "Storage" },
- ... { "Location.Owner" : "Anderson, Thomas"}] }):
- ... doc

RETRIEVING ITEMS WITH $SLICE

- >>> collection.find_one({"ItemNumber" : "6789SID"}, {"PreviousLocation" : {"$slice" : 3} })
- >>> collection.find_one({"ItemNumber" : "6789SID"}, {"PreviousLocation" : {"$slice" : -3} })
- >>> collection.find_one({"ItemNumber" : "6789SID"}, {"PreviousLocation" : {"$slice" : [5, 3] }
- })

REGULAR EXPRESSION

- >>> import re

- >>> for doc in collection.find({"ItemNumber" : re.compile("4")}, {"ItemNumber" : "true"}):
- ... Doc

- >>> for doc in collection.find({"ItemNumber" : re.compile(".FS$")}, {"ItemNumber" : "true"}):
- ... Doc

- >>> for doc in collection.find({"Location.Owner" : re.compile("^anderson.", re.IGNORECASE)},
- ... {"ItemNumber" : "true", "Location.Owner" : "true"}):
- ... doc

MODIFYING THE DATA

- // Define the updated data
- **>>> update = ({**
- **"Type" : "Chair",**
- **"Status" : "In use",**
- **"Tags" : ["Chair","In use","Marketing"], "ItemNumber" : "6789SID",**
- **"Location" : { "Department" : "Marketing", "Building" : "2B", "DeskNumber" : 131131,**
- **"Owner" : "Martin, Lisa“ }**
- **})**
- // Now, perform the update
- **>>> collection.update({"ItemNumber" : "6789SID"}, update)**

$INC $SET $UNSET

- Increasing an Integer value with $Inc
- >>> collection.update({"ItemNumber" : "6789SID"}, {"$inc" : {"Location.DeskNumber" : 20}})

- Changing an Existing value with $set
- >>> collection.update({"Location.Department" : "Development"},
- ... {"$set" : {"Location.Building" : "3B"} },
- ... upsert = True, multi = True)

- Removing a key/value field with $unset
- >>> collection.update({"Status" : "Not used", "ItemNumber" : "2345FDX"},
- ... {"$unset" : {"Location.Building" : 1 } })

$PUSH $PUSHALL $ADDTOSET

- >>> collection.update({"Location.Owner" : "Anderson, Thomas"},
- ... {"$push" : {"Tags" : "Anderson"} }, multi = True)

- >>> collection.update({"Location.Owner" : re.compile("^Walker,")},
- ... {"$pushAll" : {"Tags" : ["Walker","Warranty"] } })

- >>> collection.update({"Type" : "Chair"}, {"$addToSet" : {"Tags" : "Warranty"} }, multi =
- True)

REMOVING AN ELEMENT $POP $PULL

- >>> collection.update({"Type" : "Chair"}, {"$pop" : {"Tags" : -1}})
- >>> collection.update({"Type" : "Chair"}, {"$pop" : {"Tags" : 1}})
- >>> collection.update({"Type" : "Chair"}, {"$pull" : {"Tags" : "Double"} }, multi = False)

FINDANDMODIFY()

- >>> db.command("findandmodify", "items", query = {"Type" : "Desktop"},
- ... update = {"$set" : {"Status" : "In repair"} }, new = True)

- >>> db.command("findandmodify", "items", query = {"Type" : "Desktop"},
- ... sort = {"ItemNumber" : -1}, remove = True)

DELETING DATA

- >>> collection.remove({"Status" : "In use"})
- >>> collection.find_one({"Status" : "In use"})

- >>> db.items.drop()

- >>> c.drop_database("inventory")

CREATING LINK BETWEEN TWO DOCUMENT

- >>> jan = {
- ... "First Name" : "Jan", ... "Last Name" : "Walker", ... "Display Name" : "Walker, Jan",
- ... "Department" : "Development", ... "Building" : "2B", ... "Floor" : 12, ... "Desk" : 120103,
- ... "E-Mail" : "jw@example.com"
- ... }
- >>> people = db.people
- >>> people.insert(jan)

- >>> laptop = {
- ... "Type" : "Laptop", ... "Status" : "In use", ... "ItemNumber" : "12345ABC",
- ... "Tags" : ["Warranty","In use","Laptop"],
- ... "Owner" : jan["_id"]
- ... }
- >>> items = db.items
- >>> items.insert(laptop)

DBREF

- >>> from pymongo.dbref import DBRef
- >>> mike = {
- ... "First Name" : "Mike", ... "Last Name" : "Wazowski", ... "Display Name" : "Wazowski, Mike",
- ... "Department" : "Entertainment", ... "Building" : "2B", ... "Floor" : 10, ... "Desk" : 120789,
- ... "E-Mail" : "mw@monsters.inc"
- ... }
- >>> people.save(mike)

- >>> laptop = {
- ... "Type" : "Laptop",
- ... "Status" : "In use",
- ... "ItemNumber" : "2345DEF",
- ... "Tags" : ["Warranty","In use","Laptop"],
- ... "Owner" : DBRef('people', mike["_id"])
- ... }
- >>> items.save(laptop)

CREATING A BLOG APPLICATION WITH PHP

- **Functionalities:**
- Adding, deleting, and modifying posts.
- Viewing posts on the front page.
- Searching for a post using regular expressions.
- Retrieving author names through DBRef.

DESIGNING THE APPLICATION :POST

```
{
"Title" : "This is a blogtitle",
"Author" :
{
"$ref" : "authors",
"$id" : ObjectId("4c612fa774740000000034b5")
},
"Date" : "Tue Aug 10 2010 13:14:08 GMT+0200",
"Message" : "Hooray! This is a blog",
"Comments" : [
{
"Name" : "Louis",
"Comment" : "First!"
},
{
"Name" : "Stewie",
"Comment" : "Second"
}
]
}
```

DESIGNING THE APPLICATION :AUTHOR

- {
- "Name" : "Eelco",
- "E-Mail" : "eelco@mongo.db",
- "Interests" : "MongoDB",
- "_id" : ObjectId("4c612fa77474000000034b5")
- }

EXAMPLE

- **> use blog**
- switched to db blog
- **> author = ({ "Name" : "Eelco", "E-Mail" : "eelco@mongo.db", "Interests" : "MongoDB" })**
- { "Name" : "Eelco", "E-Mail" : "eelco@mongo.db", "Interests" : "MongoDB" }
- **> db.authors.save(author)**

EXAMPLE

- > post = ({
- "Title" : "This is a blogtitle",
- "Author" : new DBRef ('authors', author._id),
- "Date" : new Date(),
- "Message" : "Hooray! This is a blog",
- "Comments" : [{ "Name" : "Louis", "Comment" : "First!"},
- {"Name" : "Stewie", "Comment" : "Second“ }]
- })
- > db.posts.save(post)

PHP SNIPPET

```
<?php
// Let's define some variables!
$db = "blog"; // This is the name of the database
$col_authors = "authors"; // This is the name of the authors collection
$col_posts = "posts"; // This is the name of the posts collection
$limit = 10; // This is the total number of posts displayed
// Connect to the database
$c = new Mongo();
// Execute a search and store the posts under the cursor variable
$cursor = $c->$db->$col_posts->find()->limit($limit)->sort(array('_id'=>-1));
// For each document it finds within the collection, print it's contents
while ($document = $cursor->getNext())
{
print_r($document);
}
```

```
while ($document = $cursor->getNext())
{
// Get the author's name via MongoDBRef
$ref = $c->$db->getDBRef($document["Author"]);
$author = $ref["Name"];
// Translate the date back using date()
$date = date('M d, Y @ h:i', $document["Date"]->sec);
// Show the title, author, date and message
print "<h1>$document[Title]</h1><br>";
print "<i>By $author on $date</i>";
print "<p>$document[Message]</p>";
// Show clickable link to view the comments when found, and count them
$postid = new MongoId($document['_id']);
if(isset($document["Comments"]))
{
$count = count($document["Comments"]);
}
else {
$count = 0;
}
$count = count($document["Comments"]);
print "<a href='view.php?id=$postid'>View Comments ($count)</a>";
}
```

PAGING

- // Set up some PHP code to check if the $page is set and if not, set $page to '1'.
- if(isset($_GET['page'])){
- $page = $_GET['page'];
- }
- else {
- $page = 1;
- }
- // Let's define some more variables for paging
- $offset = ($page -1) * $limit;
- $nextpage = ($page +1);
- $prevpage = ($page -1);
- $cursor = $c->$db->$col_posts->find()->skip($offset)->limit($limit)->sort(array('_id'=>-1));

PAGING

```
// Perform a count on all posts within the collection and add links based upon the outcome
$posts = $c->$db->$col_posts->find()->count();
if ($page > 1)
{
print "<a href='?page=$prevpage'>Previous Page</a>";
if ($page * $limit < $posts)
{
print "<a href='?page=$nextpage'>Next page</a>";
}
}
else {
if ($page * $limit < $posts) {
print "<a href='?page=$nextpage'>Next page</a>";
}
}
?>
```

ADMIN PAGE

- <html>
- <head>
- <title>MongoDB PHP</title>
- </head>
- <body>
- <frameset cols="15%,85%">
- <frame src="nav.html"/>
- <frame src="posts.php" name="contents"/>
- </frameset>
- </body>
- </html>

NAVIGATION.HTML

- <html>
- <body>
- View Posts
- Add Posts
- </body>
- </html>

ADD.PHP

```
<?php
// First, lets connect to the database and get a list of authors
$c = new Mongo();
$db = "blog";
$col_posts = "posts";
$col_authors = "authors";
$cursor = $c->$db->$col_authors->find();
print "<h1>Let's add a post!</h1>";
print "<form action='add.php' name='addpost' method='posts'>";
print "<input type='text' name='Title' value='Fill in the title'><br>";
print "Post as: <select name='names'>";
while ($names = $cursor->getNext())
{
print "<option value=$names[_id]>$names[Name]";
}
print "</select><br>";
print "<textarea cols='40' rows='8' name='Message'>Type here</textarea>";
print "<input type='submit' name='addpost' value='Add'/>";
print "</form>";
// Specify the behavior when clicking on Add
if(isset($_GET["addpost"]))
{
// Create a new array to store the changed values in
$arr = array();
$arr['Title'] = addslashes($_GET['Title']);
$arr['Message'] = addslashes($_GET['Message']);
$arr['Author'] = MongoDBRef::create(
$c->$db->$col_authors->getName(),
new MongoId($_GET['names'])
);
$arr['Date'] = new MongoDate();
$c->$db->$col_posts->insert($arr);
print "Post added. You can leave this page.";
}
?>
```

POST.PHP

```php
<?php

// Let's define some variables!
$db = "blog";                           // This is the name of the database
$col_authors = "authors";               // This is the name of the authors collection
$col_posts = "posts";                   // This is the name of the posts collection
$limit = 10;                            // This is the total number of posts displayed

// Set up some PHP code to check if the $page is set and if not, set $page to '1'.
if(isset($_GET['page'])){
    $page = $_GET['page'];
}
else {
    $page = 1;
}

if(isset($_POST["search"]))
{
    // Use MongoRegEx to specify the search criteria
    $regex = new MongoRegex("/$_POST[search]/i");
    $query = array("Message" => $regex);
}
else
{
    $query = array();
}

// Let's define some more variables for paging
$offset = ($page -1) * $limit;
$nextpage = ($page +1);
$prevpage = ($page -1);

// Add a form for searching the posts
print "<form method='post' name='search' action='posts.php'>";
print "<input type='text' name='search'>";
print "<input type='submit' value='Search'>";
print "</form>";

// Add a link to add a post
print "<a href='add.php'>Add a post</a>";
```

MODIFY POST

- // Specify what ought to happen when the Modify link has been clicked
- if(isset($_GET["modify"]))
- {
- // Use the posted ID to retrieve the data, and input it in a form
- $filter = array("_id" => new MongoId($_GET["modify"]));
- $post = $c->$db->$col_posts->findOne($filter);
- print "<form action='posts.php' name='modifypost' method='post'>";
- print "Title<input type='text' name='Title' value='$post[Title]'/>
";
- print "Message <textarea rows='5' cols='40' name='Message'>$post[Message]</textarea>";
- print "<input type='hidden' name='id' value='$_GET[modify]'/>";
- print "<input type='submit' name='modifypost' value='Change'/>";
- print "</form>";
- }

DELETE

- // Specify what ought to happen when the Delete link has been clicked
- if(isset($_GET["del"]))
- {
- // Use the posted ID as the _id, and convert it via MongoId
- $id = array("_id" => new MongoId($_GET["del"]));
- // Specify the options
- $options = array('justOne' => true, 'safe' => true);
- // Connect to the database and remove the document
- $c->$db->$col_posts->remove($id, $options);
- // In case the document was deleted successfully, report a success
- print "Post removed successfully";
- }

DATABASE ADMINISTRATION

- Back up and restore your MongoDB system.
- Use the supplied MongoDB shell to perform common tasks.
- Control access to your server with authentication.
- Monitor your database instances.

BACKING UP THE MONGODB SERVER

- As a new MongoDB administrator, the first skill you should learn is how to back up and restore your MongoDB server.
- The MongoDB backup utility is called mongodump;
- $> cd ~
- $> mkdir testmongobackup
- $> cd testmongobackup
- $> mongodump

```
$ mongodump
connected to: 127.0.0.1
all dbs
DATABASE: blog to dump/blog
    blog.authors to dump/blog/authors.bson
        2 objects
    blog.system.indexes to dump/blog/system.indexes.bson
         3 objects
    blog.posts to dump/blog/posts.bson
        1 objects
    blog.tagcloud to dump/blog/tagcloud.bson
        4 objects
DATABASE: admin to dump/admin
```

COMMAND TO RESTORE

- $> cd ~/testmongobackup
- $> mongorestore --drop

```
connected to: 127.0.0.1
Thu Aug 12 13:31:20    dump/blog/authors.bson
Thu Aug 12 13:31:20    going into namespace [blog.authors]
Thu Aug 12 13:31:20    dropping
Thu Aug 12 13:31:20    2 objects found
Thu Aug 12 13:31:20    dump/blog/posts.bson
Thu Aug 12 13:31:20    going into namespace [blog.posts]
Thu Aug 12 13:31:20    dropping
Thu Aug 12 13:31:20    1 objects found
Thu Aug 12 13:31:20    dump/blog/system.indexes.bson
Thu Aug 12 13:31:20    going into namespace [blog.system.indexes]
Thu Aug 12 13:31:20    dropping
Thu Aug 12 13:31:20    3 objects found
```

BACKING UP & RESTORE A SINGLE DATABASE

- $mkdir ~/backuptemp
- $cd ~/backuptemp
- $mongodump -d blog -c posts
- $mongodump -d blog -c tagcloud

- $cd ~/testmongobackup
- $mongorestore -d blog --drop
- $cd ~/testmongobackup
- $mongorestore -d blog -c posts --drop

MONGODUMP OPTIONS

```
$mongodump --help
options:
  --help                        produce help message
  -v [ --verbose ]              be more verbose (include multiple times for more
                                verbosity e.g. -vvvvv)
  -h [ --host ] arg             mongo host to connect to ("left,right" for pairs)
  --port arg                    server port. Can also use --host hostname:port
  -d [ --db ] arg               database to use
  -c [ --collection ] arg       collection to use (some commands)
  -u [ --username ] arg         username
  -p [ --password ] arg         password
  --ipv6                        enable IPv6 support (disabled by default)
  --dbpath arg                  directly access mongod database files in the given
                                path, instead of connecting to a mongod  server -
                                needs to lock the data directory, so cannot be used
                                if a mongod is currently accessing the same path
  --directoryperdb              if dbpath specified, each db is in a separate
                                directory
  -o [ --out ] arg (=dump)      output directory
  -q [ --query ] arg            json query
```

MONGORESTORE OPTIONS

```
$mongorestore --help
usage: mongorestore [options] [directory or filename to restore from]
options:
  --help                  produce help message
  -v [ --verbose ]        be more verbose (include multiple times for more
                          verbosity e.g. -vvvvv)
  -h [ --host ] arg       mongo host to connect to ("left,right" for pairs)
  --port arg              server port. Can also use --host hostname:port
  -d [ --db ] arg         database to use
  -c [ --collection ] arg collection to use (some commands)
  -u [ --username ] arg   username
  -p [ --password ] arg   password
  --ipv6                  enable IPv6 support (disabled by default)
  --dbpath arg            directly access mongod database files in the given
                          path, instead of connecting to a mongod  server -
                          needs to lock the data directory, so cannot be used
                          if a mongod is currently accessing the same path
  --directoryperdb        if dbpath specified, each db is in a separate
                          directory
  --objcheck              validate object before inserting
  --filter arg            filter to apply before inserting
  --drop                  drop each collection before import
  --indexesLast           wait to add indexes (faster if data isn't inserted in
                          index order)
```

AUTOMATING BACKUPS

```
#!/bin/bash
########################################
# Edit these to define source and destinations

MONGO_DBS=""
BACKUP_TMP=~/tmp
BACKUP_DEST=~/backups
MONGODUMP_BIN=/usr/bin/mongodump
TAR_BIN=/usr/bin/tar

########################################

BACKUPFILE_DATE=`date +%Y%m%d-%H%M`


# _do_store_archive <Database> <Dump_dir> <Dest_Dir> <Dest_file>

function _do_store_archive {
        mkdir -p $3
        cd $2
        tar -cvzf $3/$4 dump
}

# _do_backup <Database name>

function _do_backup {
        UNIQ_DIR="$BACKUP_TMP/$1"`date "+%s"`
        mkdir -p $UNIQ_DIR/dump
        echo "dumping Mongo Database $1"
        if [ "all" = "$1" ]; then
                $MONGODUMP_BIN -o $UNIQ_DIR/dump
        else
                $MONGODUMP_BIN -d $1 -o $UNIQ_DIR/dump
        fi
        KEY="database-$BACKUPFILE_DATE.tgz"
        echo "Archiving Mongo database to $BACKUP_DEST/$1/$KEY"
        DEST_DIR=$BACKUP_DEST/$1

        _do_store_archive  $1 $UNIQ_DIR $DEST_DIR $KEY

        rm -rf $UNIQ_DIR
}
```

BACKING UP LARGE DATABASES

- Using a slave server for backups.
- Creating snapshots with a Journaling Filesystem
  - A snapshot is like a bookmark that allows you to read the drive exactly as it was when the snapshot was taken.
- The procedure for using a snapshot goes something like this:
  - 1. Create a snapshot.
  - 2. Copy data from the snapshot or restore the snapshot to another volume, depending on your volume manager.
  - 3. Release the snapshot; doing so releases all preserved disk blocks that are no longer needed back into the free space chain on the drive.
  - 4. Back up the data from the copied data while the server is still running.

IMPORTING DATA INTO MONGODB

```
$mongoimport --help
options:
  --help                  produce help message
  -v [ --verbose ]        be more verbose (include multiple times for more
                          verbosity e.g. -vvvvv)
  -h [ --host ] arg       mongo host to connect to ("left,right" for pairs)
  --port arg              server port. Can also use --host hostname:port
  -d [ --db ] arg         database to use
  -c [ --collection ] arg collection to use (some commands)
  -u [ --username ] arg   username
  -p [ --password ] arg   password
  --ipv6                  enable IPv6 support (disabled by default)
  --dbpath arg            directly access mongod database files in the given
                          path, instead of connecting to a mongod  server -
                          needs to lock the data directory, so cannot be used
                          if a mongod is currently accessing the same path
  --directoryperdb        if dbpath specified, each db is in a separate
                          directory
  -f [ --fields ] arg     comma separated list of field names e.g. -f name,age
  --fieldFile arg         file with fields names - 1 per line
  --ignoreBlanks          if given, empty fields in csv and tsv will be ignored
  --type arg              type of file to import.  default: json (json,csv,tsv)
  --file arg              file to import from; if not specified stdin is used
  --drop                  drop collection first
  --headerline            CSV,TSV only - use first line as headers
  --upsert                insert or update objects that already exist
  --upsertFields arg      comma-separated fields for the query part of the
                          upsert. You should make sure this is indexed
  --stopOnError           stop importing at first error rather than continuing
  --jsonArray             load a json array, not one item per line. Currently
                          limited to 4MB.
```

IMPORT

- The mongoimport utility can load data from three different file formats:
- 1. CSV: In this file format, each line represents a document, and the fields are separated by a comma.
- 2. TSV: This file format is similar to CSV; however, it uses a tab character as the delimiter. This format is popular because it does not require the escaping of any text characters other than those for new lines.
- 3. JSON: This format contains one block of JSON per line that represents a document. Unlike the other formats, JSON can support documents with variable schemas

EXPORTING DATA FROM MONGODB

```
$mongoexport --help
options:
  --help                    produce help message
  -v [ --verbose ]          be more verbose (include multiple times for more
                            verbosity e.g. -vvvvv)
  -h [ --host ] arg         mongo host to connect to ("left,right" for pairs)
  --port arg                server port. Can also use --host hostname:port
  -d [ --db ] arg           database to use
  -c [ --collection ] arg collection to use (some commands)
  -u [ --username ] arg     username
  -p [ --password ] arg     password
  --ipv6                    enable IPv6 support (disabled by default)
  --dbpath arg              directly access mongod database files in the given
                            path, instead of connecting to a mongod  server -
                            needs to lock the data directory, so cannot be used
                            if a mongod is currently accessing the same path
  --directoryperdb          if dbpath specified, each db is in a separate
                            directory
  -f [ --fields ] arg       comma separated list of field names e.g. -f name,age
  --fieldFile arg           file with fields names - 1 per line
  -q [ --query ] arg        query filter, as a JSON string
  --csv                     export to csv instead of json
  -o [ --out ] arg          output file; if not specified, stdout is used
  --jsonArray               output to a json array rather than one object per
                            line
```

SECURE YOUR DATA

- Restricting Access to MongoDB Server
  - MongoDB supports a simple authentication system that allows you to control who has access to each database, and what level of access they are granted.
- Protecting server with Authentication
  - MongoDB supports a simple authentication model that allows the administrator to restrict access to databases on a per user basis.
  - Adding an Admin user

```
$mongo

> use admin

>db.addUser("admin", "adminpassword")
```

  - Enabling Authenticaion

```
$sudo service mongodb restart
```

SECURE YOUR DATA

- Changing a User's Credentials
- **$mongo**
- **>use admin**
- **> db.addUser**
- function (username, pass, readOnly) {
- readOnly = readOnly || false;
- var c = this.getCollection("system.users");
- var u = c.findOne({user:username}) || {user:username};
- u.readOnly = readOnly;
- u.pwd = hex_md5(username + ":mongo:" + pass);
- print(tojson(u));
- c.save(u);
- }

MONITORING MONGODB

- The MongoDB distribution contains a simple status-monitoring tool called mongostat.
- This tool is designed mainly to provide a simple overview of what is happening on the server.
- The main columns of interest are the first six columns.
- Other columns useful when diagnosing problems with the installation include:
  - %idx miss - % of queries that could not use an index.
  - Conn – shows how many connections are opened to mongodb instance.
  - % locked – Shows amount of time the server had its collection locked.

MONGOSTAT TOOL

```
$ mongostat
connected to: 127.0.0.1
insert/s query/s update/s delete/s getmore/s command/s mapped  vsize    res % locked % idx miss  conn     time
       0       3        4        0          0          1   1056   1749    527 0.000404          0    47 11:02:17
      12       0        0        0          0          3   1056   1789    527 0.000143          0    51 11:02:18
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:19
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:20
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:21
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:22
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:23
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:24
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:25
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:26
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:27
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:28
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:29
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:30
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:31
       0       0        0        0          0          1   1056   1789    527        0          0    51 11:02:32
       0       3        4        0          0          1   1056   1789    527  0.00039          0    51 11:02:33
      12       0        0        0          0          3   1056   1829    527 0.000156          0    55 11:02:34
       0       0        0        0          0          1   1056   1829    527        0          0    55 11:02:35
       0       0        0        0          0          1   1056   1829    527        0          0    55 11:02:36
insert/s query/s update/s delete/s getmore/s command/s mapped  vsize    res % locked % idx miss  conn     time
```

THANK YOU