Big data formats
structured,
Unstructured
semi-structured

front end which gives data and show the data and retrieve the data
back end which store data e.g. Database
Column refers as fields
Rows refers as records

Page No.
Date 7 6 19

Unit - I

Chapter 1. Big Data

- Data is raw facts which can be store, processed, transmitted.
- Information is processed data or collection of organic data.
- Database is collection of related data, it is storage ~~data~~ area, which can be store storage related type data.
- Big data is large size, which can be complex type data.

* Data :-

- The quantities character or symbols on which operations are performed by a computer which may be stored and transmitted in the form of electric signal and recorded on maganetic, optical or mechanical recording area.

* Big data :-

- Big data is also data but with a Huge size big data is term Huge to describe collection of data that is Huge in size and yet growing exponential with time such a data with is also large and complex that none of the traditional data management able to store or process it effectively.

Structure data - fixed size, type.
unstructure data - not fixed size, type.

Database create nth no. of database you can create in system
RDBMS not follows Big data
nth no. of table we can create in database

Page No.
Date 18 6 19

* Sources of Big Data :-
- Generation of Big Data → Storing of Big Data → utilizati
Of Big Data
example :
Social media, industries, education, entertainment, business, etc.

11/6/2019

* Categories of Big Data :-
- Big data can be found in three basic
(i) structured Data
(ii) Unstructured Data
(iii) Semi-structured Data
(i) structured data :
- Any data that can be store access and process in the form of fixed format is term as structured data.
- Over the period of time science have achieve greater sucess in developing techniques for working such type of ~~the~~ data where the format well known in advanced.
- It also help to derive value out of it
- example of structured data :
Any Relational DataBase
for example :
employee table database or Student table database.
- However nowadays we are forcing issues when size of such data grows to a huge extented where data can not be store in a traditional fixed pattern.

semi-structured data - fixed location, position but not fixed size, type.
e.g. Google History, logfiles
Mongodis server and Mongod is client

Page No.
Date

- Advantage of structure data is easy retrieval, manipulation and storage of data.

(ii) Unstructured Data :

- Any data with unknown form or the structure is classified as unstructured data. e.g. Google search.
- In addition to the size to be Huge unstructured data passes multiple ~~size~~ challenges in terms or processing or ~~derive~~ deriving value out of it.
- Typical example of unstructure data is heterogenous data source containing a combination of simple file, text, files image, video, etc.
- Nowadays organization have ~~a~~ a wealth of data available with them but unfortunately they don't known how to derive the value out of it since the data is in raw or unstructured format.

(iii) semi-structured data :

- semi-structured data can contain both the form of data we can see semi-structured data as a structure in form but it is actually not defined.
- example of semi-structured data :
A data growth over ~~a~~ years.

12/06/2019

imp * Characteristics of Big Data :-
(three V's Of Big Data)
(i) Volume → Large size

(ii) Variety - complex type
(iii) Velocity - growing & rapidly

(i) Volume :-
- The ~~na~~ name big data itself is related to a size which anormous.
- Size of data plays very ~~crucial~~ crucial role in determining value act of data.
- Also ~~weather~~ whether the particalar ~~that~~ data can actually be considered as big data or not is dependent upon volume of data.
- Hence volume is one charateristic we needs to be consider while dealing with big data.

(ii) Variety :-
- The next aspect of big data is it's variety.
- variety refers to heterogenous sources and nature of data both structured and unstructured data previously spread sheet and databases where the only source of data considered by most of the applications.
- Nowadays data in form of image, photos, pdf, audio, etc. is also been consider in the analysis application
- The variety of unstructure data passes certain issues of storage mining and analyzing data.

Segmantation and customization Eg. if we need a shows the direaction of particular city/area ~~mean~~ then you give ~~all particular area~~ direction like Google map

Page No.
Date

(iii) Velocity :-

- The term velocity refers to the speed of generation of data.
- How fast data is generated and processed to meet the demand determine ~~ce~~ the real potential ~~to~~ in the data.
- big data velocity deals with the speed of data flow in form sources like business processing application log s, network and socialmediasite, sensor, ete.
- The flow of data is messive and continous.

13|06|2019

* Benefits of Big Data processing :-
(i) Businesses can utilize outside intelligence while taking decision.
(ii) improved customer service.
(iii) ~~ear~~ early identification of risk to the product or service
(iv) Better operational efficiency

14|06|2019

* Usages of Big Data :-
(i) visibility
(ii) segmentation and customization
(iii) Innovation
(iv) Aiding decision making
(v) discover and ~~analysi~~ analise information.

* Big data challanges :-

(i) policy and procedures

(ii) Access to data

(iii) technology and techniques

~~(i) policy and proc~~

* Legacy system and Big Data :-

(i) structure of Big data

(ii) storage of Big data

(iii) Data processing

* Big data Technologies :-

(i) new storage and processing technologies designed specifically for large unstructured data.

(ii) Parallel processing

(iii) clustering (creating small-small parts)

(iv) Large Grid Environment

(v) High connectivity

(vi) cloud computing (external storage of data) e.g. drive

NoSQL is a term for unstructured data.
NoSQL follows BASE

Sql → Structured query language
↓
fixed data sql follows ACID
commit means to save the data.

Page No.
Date 22 06 19

Q. Explain CAP Theoram

chapter 2.

NoSql

- NO SQL follows BASE.
- BASE (Basically Available soft state Eventually consistent)
- BASE derived by ~~theoram~~ CAP theoram.
- imp CAP theoram stands for consistency, Availability, partition Tallerance
- Big data follows CAP Theoram ~~but~~ follows mysql follows BASE

* Advantages of NOSQL :-
- High scalibility
- Managability and Administaration
- Low cost (we can use small server to store data)
- ~~fexlixible data model~~ flexible data model

* Disadvantage Of NOSQL :-
- Maturity (it is not fully developed)
- Support (Not create in High company. create by small (starter))
- limited query capability (only complex query not handle)
- Administration (it is open source, so we can run individual)
- Experties (They are not expert people, few people have Knowledge about NOSQL)

24/06/2019

imp * Read and write operation :-
- Read means you can view (show) the data we can not edit a data.
- write means you can modify, view, edit the data.

Write Operation :-

N → No. of data copies

R → No. of data copies we Read

W → No. of data copies we write

Example :

(i) N = W

- If we have 30 students data in your database but we can add one more students data in your database Here, all the data is update and Refresh, if the data store starting then all data is update and refresh, if the data store in middle then previous data is refresh and remaining data update, if the data store in bottom then all data is refresh and Write operation is time consuming.

(ii) W = 1, N = R

- Here, write one data and Read all the data therefore, Read operation is time consuming.

(iii) N > W > R

- We can update few data in particular time otherwise it gives lots of time. Not all updated at a same time. use in large amount of data

Read Operation :-

(i) R = 1

- Your are not sure data is first, last, second data any one data we can read.

(ii) R > 1

- Now, we can read more than one data but not all. Here, Read the new and old data. it is time consuming.

Scale in means it generate maximum size of data

W+R means updated data / newly updated data
SQL supports all standard format
NOSQL supports tabular, column, etc. key value, etc. format

Page No.
Date

Scale out means data store in Groups or small small part.

(iii) N < W + R
Reading only that data which is ~~new~~ update new data. Here only reads the new data it is not time consuming. ~~It~~ read it gives only required data to be Read or show.

* SQL Vs. NOSQL :-

Scale in means increases size of database then store the data but in Scale out it divides ~~part~~ data into parts then store the data

parameter	SQL	NOSQL
(i) type	SQL supports all types of standard format.	SQL follows particular type of data format support.
(ii) Development History	SQL develop in 1970	NOSQL develop in 2000
(iii) Example	MySQL, SQL Server, SQL, Oracle, SQLplus.	MongoDB, HBASE, Marklogic, graphicdb
(iv) Data storage model	It stores Rows, and column format.	It stores particular type of data
(v) Schemas	fixed	dynamic
(vi) Scalability	It is used in format Scale.	It is used out format scale.
(vii) support and transaction	ACID	According CAP and BASE format

(viii) consistency	Strong	Normal
(ix) Support	High support	less Support
(x) Maturity	more	less
(xi) Query capability	All query apply	Only simple query apply
(xii) Experties	More	less

26/06/2019

* categories of NOSQL Database :-

(i) Document Based → MongoDB technique

- if you use mongoDB then data store in collections into documents.
- Documents contain for example, if you want to store student info. the each students info makes single document to store student info.

(ii) XML Based database → Marklogic technique

for example.

```
<xml>
<student>
<name> ABC </name>
<id> 01 </id>
</student>
<student> </XML>
```

One student record

→ at the end

- Here, if you want to store data in form xml format then we can use above pattern. as in above e.g. only store the one student info, if you want multiple info then open <student> after all data entred then XML close.

(iii) Graph Based database → GraphDB technique

- if you want to information store in Graph format then we use ~~this to cat~~ Graph Based database by using GraphDB technique.

(iv) Key value store → Redis, cassandra technique

- MongoDB is store unstructure data then we can store data in Key:value format.

if you give cap then compuslory to give size of saturation point value only one collection accept.

Page No. | Date 1 7 19

* Creation of ~~database~~ collection :-

- Three ways to create collection as follows;

(i) db.createCollection(name)

e.g.

db.createCollection("TYITA")

(ii) db.createCollection(name,[option])

~~eg.~~ Options are

(i) capped boolean value (Yes/NO)

(ii) size bytes (saturation point value)

(iii) max (no. of documents)

(iv) autoIndexId boolean (Yes/NO)

e.g.

db.createCollection({"student", {capped:true, size:11045, max:500, autoIndexId:true}})

Here, student is ~~db~~ collection name and we give to option ~~to capped~~ for capcollection.

(iii) if you want to create directy document then automatically create collection but it is not Idle way to create.

~~db~~ e.g.

db.customer.insert({"name":"abc", "rollno.":01})

* if you want to see the list of all collections :-

Syntax :

show collections

* Drop collection :

db.collection_name.drop()

e.g. db.TYITA.drop()

MongoDB is case sensitive.

square bracket indicates we can insert multiple value.

Page No.
Date 5 07 19

* inserting Data or/creating documents :-

syntax :

db.collection_name.insert ({key:value, key:value})

example :

(i) db.student.insert({"name":"ABC", "id":01})

(ii) db.student.insert({"Rollno id":02, "name":"PQR

(iii) db.student.insert({"id":03, "name":"XYZ", "Gender
"F"})

(iv) db.student.insert({"name":"LMN", "id":"IT03",
"gender":"m", "phoneno":[99...., 98....]})

multiple value

(v) db.student.insert({"name":{"firstname":"xyz",
"Lastname":"IJK"}, "id":04,
"gender":"F", "Address":{"city":
"panvel", "state":"Maharastra"}})

* Read :-

- It is also one way inserting data.

example :

user1 = {"name":"ABC", "id":01, "gender":"F"}
db.student.insert({user1})

Here, we insert document by using variable.

Inserting Data by Loop :

eg for e.g :

for(var i=1; i<=20; i++) db.student.insert({"name":
"username"+i, "subject":"NGT", "age":20})

if the parameter is same and particular key is increment / decrement.

Page No.
Date 6 07 19

* Update Data :-

for example :

- db.student.update({"gender":'F'},{$set:{"country":"India"}})

Here, we can update only ~~one row~~ starting condition can be change.

- if you want to all conditions ~~are present then~~ update then use {multi: True/false} by default is false.

e.g. db.student.update({"gender":'F'},{$set:{"country":'India'}},{multi:True})

- if you all data can be update then

db.student.update({},{$set:{"country":'India'}},{multi:True})

- if you want to remove key then we can use $unset.

db.student.update({"gender":'F'},{$unset:{"country":'India'}},{multi:'True'})

* Delete Data :-

- if you want to remove the document OR record but collections ~~name~~ remains same.

db.student.remove({})

8/07/2019

(*) query document :-

(i) selector (According what you seleting data, cond^n)

(ii) projector (what is display}, without cond^n)

for example :

select * from student where address = "panvel";

projector — selector

when MongoDB select or Retrieve by using find.
e.g. db.collection_name.find()
db.collection_name.find().pretty()
preety() → it gives proper structure

db.collection_name.find().pretty()

Syntax :
db.collection_name.find({selector},{projector})

selector gives value in key: value

projector gives value key: 1/0 (true/false) default is 0.

example :
(i) db.collection_name.find()
- it gives all data without cond^n.

(ii) db.student.find({},{"name":1,"Rollno.":1})
- it gives name and rollno column without any cond^n (it means gives all data accourding to name and rollno)

(iii) db.student.find({"address":"panvel"})
- it gives all ~~manubers~~ data whose address is panvel.

(iii) limit:
limit is used to display the Q. particular no. of records.

(iv) db.student.find({"address":"panvel"},
{"name":1,"Rollno":1})

Write Mongodb query to display Age, name, Rollno, where gender is female and should not display identifiers or -id and gender.

db.student.find({"gender":'female'},{"name":1,"age":1,"Rollno.":1,"-id":0,gender:0})

(v) db.student.find({"gender":'F'}).limit(10)
Here, Only display 1st 10 female data with all records.

(vi) db.student.find({"gender":'F'},{"name":1})
.limit(10)

(iv) Skip:

Skip is used to display the middle or last records.

Here, Only display 1st 10 female data with only name column.

(vii) db.student.find({"gender":'F'},{"name":1}).limit(5)
.skip(5)

Here, skip 1st 5 records of female and after 5 records of female display the name.

(v)

Sort is used to display particular order 1 for asending (-1) for desending

(viii) db.student.find({"gender":'F'},{"name":1})
.sort({"name":1})

Here, display only female name according to asending Order.

(ix) db.student.find({"gender":'F'},{"name":1}).sort({"name":1,"Rollno":-1})

Here, display only ~~thouse~~ female name according to Name is asending and Rollno. desending condn is satisfied.

for e.g.

ABC 01		ABC 03
ABC 02	output.	ABC 02
ABC 03		ABC 01

(x) findOne():

if only one record want to display.

db.student.findOne({"gender":'F'})

Here, Only display the one record whose gender is female.

Page No.
Date 10 07 19

* Conditional Operators :-
(i) less than ($lt)
(ii) greater than ($gt)
(iii) not equal ($ne)
(iv) less than equal ($lte)
(v) greater than equal ($gte)
(vi) equal to (:)
(vii) in ($in) → inside
(viii) not in ($nin) → outside.

for example :
(i) db.student.find({"id":05})
(ii) db.student.find({"id":{$lt:05}})
(iii) db.student.find({"id":{$gt:05}})
(iv) db.student.find({"id":{$lte:02}})
(v) db.student.find({"id":{$gte:05}})
(vi) db.student.find({"id":{$ne:03}})
(vii) db.student.find({"class":{$in:[A,C]}})

01 ABC A
02 PQR B
03 XYZ C

in → 01 ABC A
03 XYZ C

(viii) db.student.find({"class":{$nin:[A,C]}})

01 ABC A
02 PQR B
03 XYZ C

not in → 02 PQR B

* AND and OR Operators :-
(i) AND Operators :

01 ABC A F India
02 PQR B F USA
03 XYZ C M India

AND → 01 ABC A F India

Page No.
Date

Syntax:

db.collection_name.find({key:value},{key:value})

db.collection_name.find({$and:[{"key":value},{"key":value}]})

for example:

(i) Write mongoDB query to retrieve data where gender is female AND country is india.

db.Student.find({"gender":'F',"country":'india'})

(ii) Write mongoDB query to display data where id is less than equals to 4 and city is Pune.

db.student.find({"id":{$lte:4},"city":'Pune'})

(iii) OR Operators:

Syntax:

db.collection_name.find({$or:[{key:value},{key:value}]})

for example:

(i) write a mongodb query where id is equal to One OR id is gredter than 5

db.student.find({$or:[{"id":01},"id":{$gt:05}]})

* ANDOR Operator:-

cond^1 AND [cond^2 OR cond^3]

Syntax:

db.collection_name.find({"key":value,$or:[{key:value},{key:value}]})

for example :
Write mongoDB query to find documents where name = ABC AND id = 1 OR 3

```
db.student.find({"name":"ABC", $or:[{"id":1},
                {"id":3}]})
```

07 | 2013

* Using indexes:-
ensureIndex() - for creating index
syntax :
- for collection appling index
db.collection_name.ensureIndex({"key": value})
- Here, value contained 1, -1
1 for appling index in asending order
-1 for appling index in desending order
- It is known as single key Index.

Compound Key Index :-
In this key index, we have two keys to apply index value because we apply the key in name but we have one more document in collection they have no name in document then we can apply compound key Index to collection.
for example :

ABC	01	Pune
PQR	02	Mumbai
XYZ	03	Pune
	04	Mumbai

Namespace allow the area for allococating the data.

Page No.
Date

Syntax:

db.collection_name.ensureIndex({"key":value(1/-1), "key":value(1,-1)})

Unique:

- It is apply for you can have unique data/records in your document. It will not allow to same data again.

Syntax:

db.collection_name.ensureIndex({"key":value(1,-1), "key":value(1,-1)},{unique:true})

dropDups:

- In this, we can use unique keyword in after entering data then we use dropDups.
- dropDups scan all the record if the record is duplicate then second record will be delete.

Syntax:

db.collection_name.ensureIndex({{"key":value(1,-1), "key":value(1,-1)},{unique:true},{dropDups:true}})

13|07|2019 getIndexes():

- if you create the index, then indexes data store in system.index
- if you want to retrieve the data by using getIndexes then it's only display the information of index.

db.collection_name.getIndexes()

$regex also define regular expression
e.g.
name : $regex/(ma)/i

Page No.
Date 13 7 19

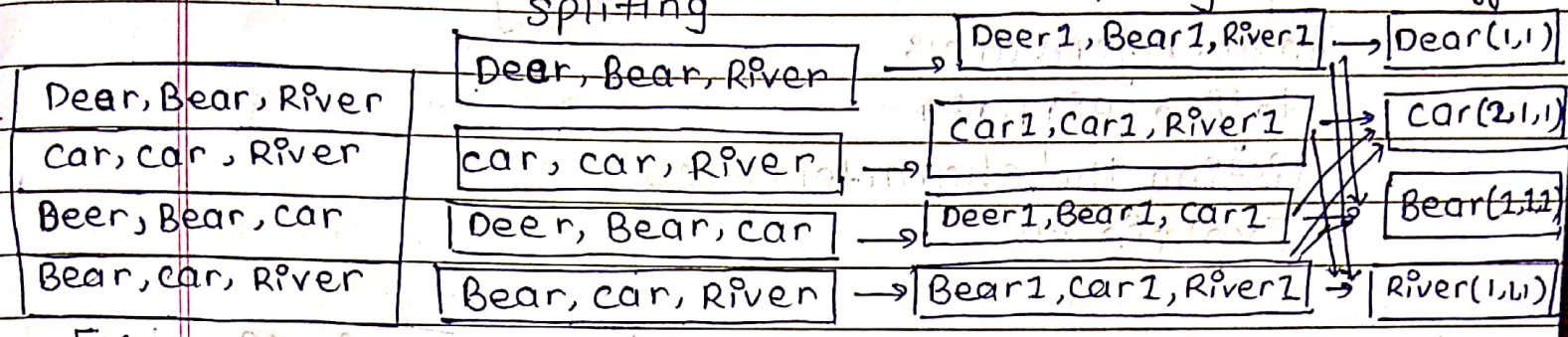
dropIndex() :
- Here, drop the index which is previously created indexes are drop/delete/Remove.
- db.collection_name.dropIndex()
- we can use with option, Sometime we create data document to make index then if you want to drop particular index then used
db.collection_name.dropIndex({"name":1})

ReIndex :
db.collection_name.reIndex()
- Here, use undo the index if you want to dro back to drop index for then we use reIndex.

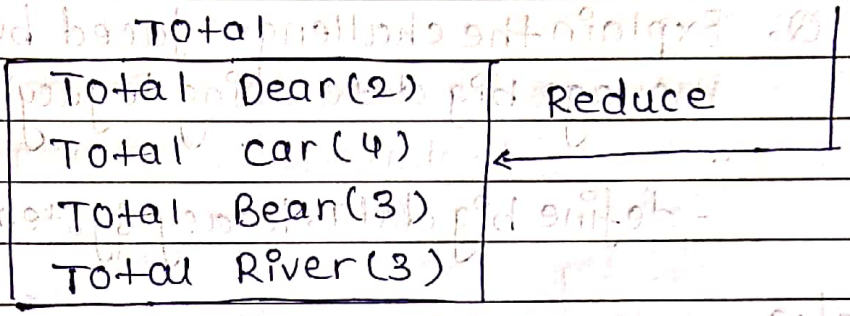
* Regular Expression :-
- if you find the particular Name whose start with ma but here only display the small ma.
db.collection student.find({"name" : /(ma)*})
- if you find the particular Name whose start with ma but you not remainber the ma is small / capital the you can use
db.student.find({"name" : /(ma)*/i})
- if you want to find to letter then more letter in your collection then
db.student.find({"name" : /(Ra/ch)*/i})
Here it will display the option whose start with 'Ra' and 'ch' record.

Page No.
Date 17 07 19

* Mapreduce :-



Example



- Here, your data first splitting in different parts then mapping the splitting data then Shuffle those data then reduce and count the data. it is working of mapreduce.

(i) var map = function() {emit(this.Gender, 1);};

- Here, Genders are divided into two. Here, your documents/data are divided by gender means male are and female Separated.

(ii) var reduce = function(key, value) {return Array.sum (value);};

(iii) db.student.mapReduce(map, reduce) {out: "mapreduce count1"})

(iv) db.mapreduceCount1.find()

- Mapreduce is very complicate or more steps are required.
- Aggregation is directly apply and it is done in only one steps.

Page No.
Date

Q. Explain the characteristics of Big data?

Answer formate:
- define Big Data
- listing the characteristics
- Explain V3

Q. Explain the challenges faced by an organization to manage big data using legacy system.

- define big data, legacy system

9/07/19

(ii) Directly apply Aggregation :-
- To perform particular/specific Group it is known as aggregation.
- first is divided into particular group and then apply the aggregate operation in group.

Syntax :
db.collection_name.aggregate(aggregate operation)

Example :
(i) Write Mongodb query, to display the total quantity of the products.

product_name	quantity	price
pen	10	₹05
Eraser	15	₹3
pencil	20	₹15

if you want to count ~~the~~ then use 1 for particular col^{n}.

if count apply the particular field.

Page No. | Date

for Addition :

db.stock.aggregate([{$group:{_id:"$productname", "total":{$sum:"$quantity"}}}])

_id : for storing data

$sum : for display purpose

(ii) Write MongoDB query to display the ~~sum~~ total no. of male and female in the collection.

for counting :

db.student.aggregate([{$group:{_id:"$gender", "totalCount":{$sum:1}}}])

for using same column to count of male and female

(iii) Write MongoDB query to display the total no. of product in the collection.

db.stock.aggregate([{$group:{_id:"$productname", "totalCount":{$sum:1}}}])

Q. Write a short note on segmantation, customisation and ~~Innovation~~ Innovation. (Innovation - 2 1/2 M, seg. Customisation - 2 1/2 M)

Q. Justify technique and technology in the ~~challege~~ challanges for big data.

20/07/19

- Aggregation not display the details of data. it will display the sum, count, average, minimum, maximum data.

Page No.
Date 20 07 19

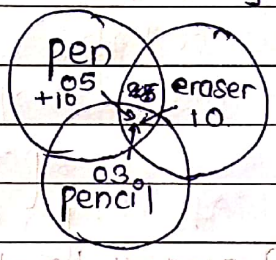
Aggregation function are different operation as follows:

(i) sum (addition of value and count of data)
(ii) avg
(iii) min
(iv) max

Example of sum Operation:

pen	05
pencil	03
eraser	10
pen	10

Group by product



for addition:

for storing data

```
db.stock.aggregate([{$group:{_id:"$productname",
"total":{$sum:"$quantity"}}}])
```

operation

for counting:

```
db.stock.aggregate([{$group:{_id:"$productname",
"totalCount":{$sum:1}}}])
```

Example of average Opertation:

Q. Write mongodb query to find the average quantity of product.

average — Group

```
db.stock.aggregate([{$group:{_id:"$product.name",
"averagequantity":{$avg:"$quantity"}}}])
```

Page No.
Date

Q. Write a mongodb query to find average score subject wise.
(average → score, subject → group)

```
db.Student.aggregate({$group:{_id:"$subject",
"AverageScore":{$avg:"$score"}}})
```

		Ch	Bi	M
01	ABC	20	19	25
02	PQR	19	19	20
03	XYZ	25	25	20

Example of minimum Operation :

Q. Write mongodb query to find minimum marks of particular subject.

```
db.student.aggregate({$group:{_id:"$subject",
"Minimum marks":{$min:"$marks"}}})
```

Example of maximum Operation :

Q. Write a mongodb query to find maximum quantity of product.

```
db.stock.aggregate({$group:{_id:"productname",
"MaximumQuantity":{$max:"$quantity"}}})
```

Q. Write a short note on visibility ?

Q. (ii) Aiding decision Aiding Decisions Making

Q. Variety is characteristic of big data justify ?

Page No.
Date 22 7 19

* Designing an application data model :-
- Embedding - putting an info/ data someone else's database E.g - Fb, book.
- ~~Refershing~~ Referencing :- Refering info/data from one database to another database.
E.g - Book refering

* Desicion of Data Model :-
(i) Manage life cycle of data there are two ways are following :
(a) set timing
(b) Capped collection

(ii) Idexes :- id by default created
(iii) Sharding :
division of data into customers that is distributing data
(iv) collection
(v) Growth of document

Q. Write a short note on sql and give reason why traditional sql is not useful for big data

Q. Differentiate between SQL and NOSQL

Q. Explain brewer's theoram