Learning Consolidation

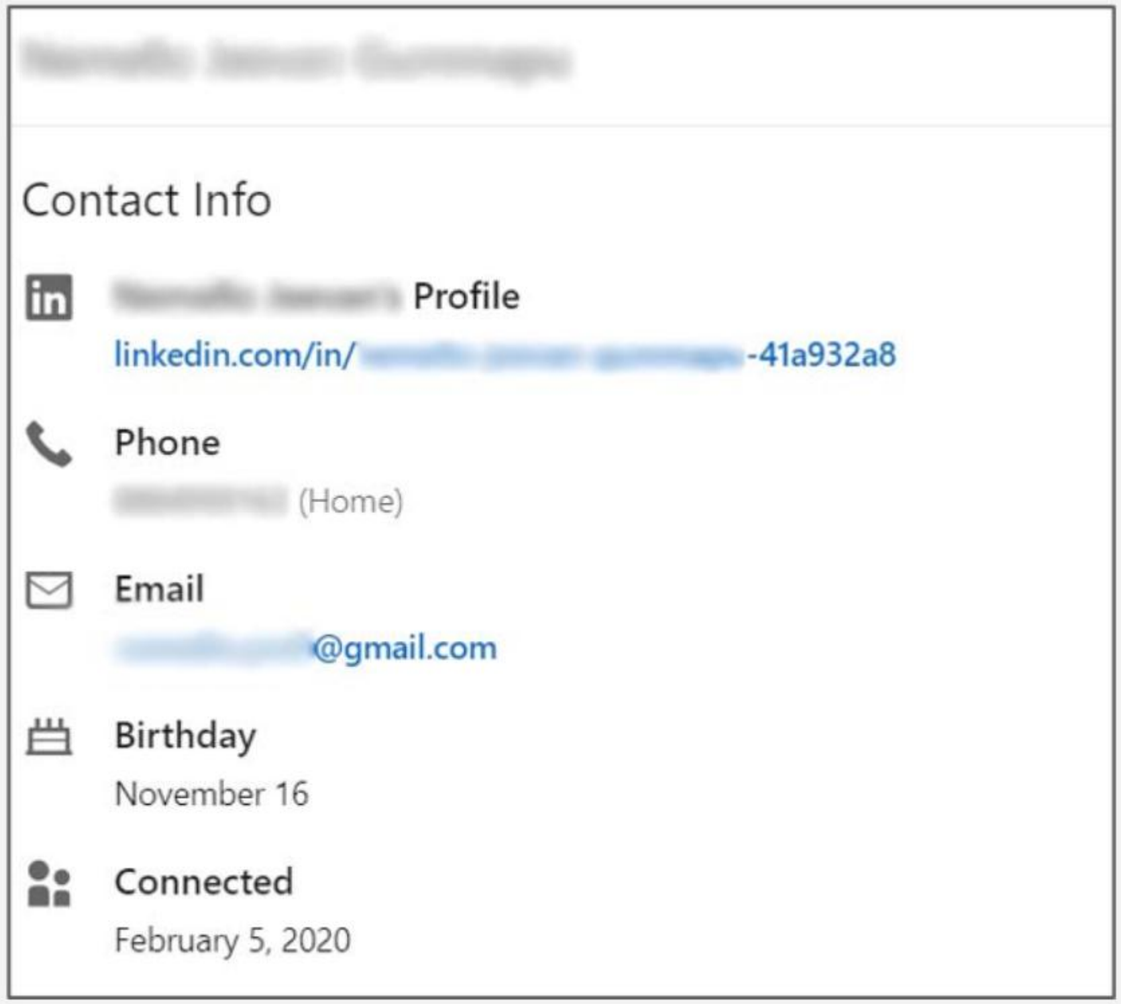
Use a NoSQL Database (MongoDB) to Manage Semi-Structured and Unstructured Data

Learning Objectives

- Explain structured and unstructured data
- Model data in an NoSQL
- Use MongoDB for a database operations

Structured Data

- Structured data consists of clearly defined data types whose pattern makes them easily searchable.
- Structured data is highly specific and is stored in a predefined format.
- The contact information page of a website has a defined structure and can be stored in an RDBMS.
- The RDBMS provides a schema, or predefined format, to store the contact information.



Slide Note

Unstructured data – “everything else” – is comprised of data that is usually not as easily searchable, including formats like audio, video, and social media postings.

Organizations used to store only some key transactional data and a few basic things about their customers. Today, however, organizations can no longer cherry-pick a few key pieces of data. They need to store just about everything.

As the cost of storage drops (even for SSDs), organizations are doing just that. There is also an expectation from customers, partners, and regulators that the organization should store everything in a usable format that will benefit them as well.

The growing amount of unstructured data presents a problem for relational databases. The rows and columns of a relational database are ideal for storing sets of values. Still, most information is composed of much more than that.

Consider something like a person’s medical record. It is incredibly heterogeneous. It includes values (name, date of birth), relationships (to family members or care providers, to symptoms and medications), geospatial data (addresses), metadata (provenance, security attributes), images (CAT scan), and free text (doctors’ notes, transcripts).

Unstructured Data

- A webpage can have different types of data displayed on a single page, like:
  - Geospatial data
  - Aggregate data
  - Audio data
  - Video data
  - Images

Slide Note

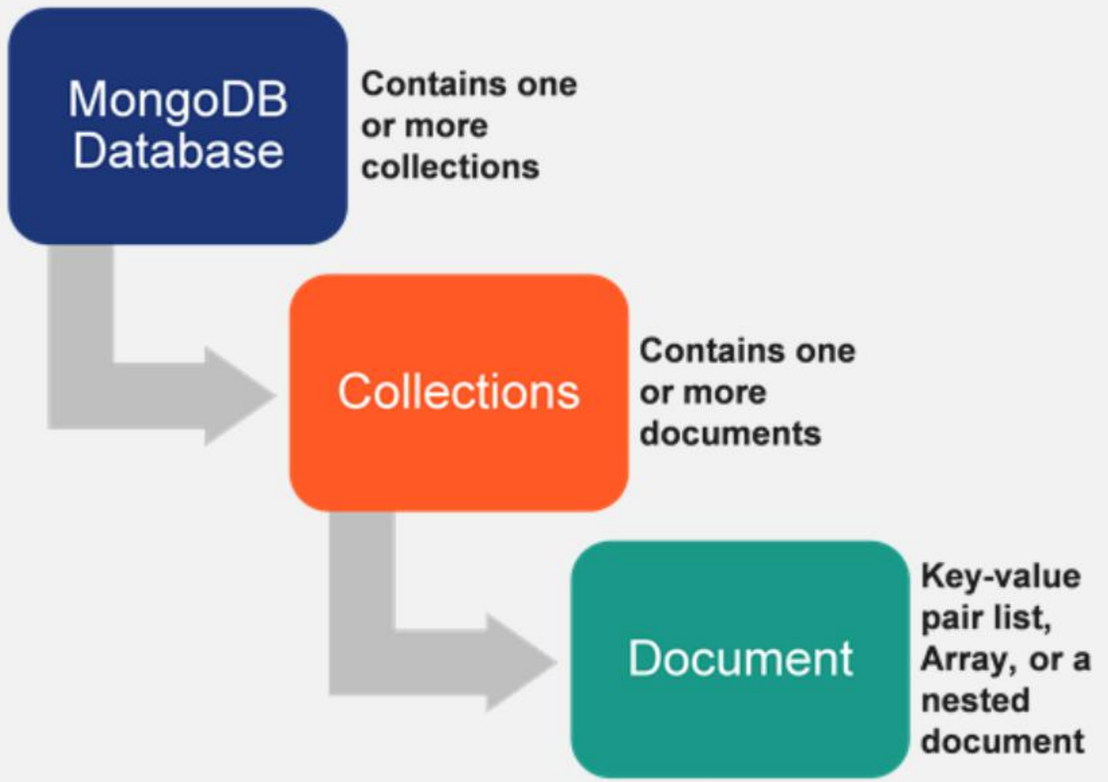
MongoDB – The Document Database

- MongoDB is a general-purpose, document-based, and distributed database built for modern application developers.
- The advantages of using MongoDB are:
  - Documents (i.e., objects) correspond to native data types in many programming languages.
  - Embedded documents and arrays reduce the need for expensive joins.
  - MangoDB provides dynamic schema support and fast read operations.
  - MongoDB represents data as JSON documents in a binary-encoded format called BSON.

Slide Note

Mongo Document Explained (contd.)

- MongoDB stores documents in collections.
- Collections are analogous to tables in relational databases.



Slide Note

Creating Database Operation

- In MongoDB, databases hold one or more collections of documents.
- Create a database in MongoDB using the `use` keyword.
- If the database is available, mongo will switch to the database, if not a database with the specified name is created.

```
> use products_db
switched to db products_db
```

```
use <databasename>
```

Slide Note

By default, the _id field is String. It can be changed when creating the document also.

In MongoDB, documents do not use sequential numeric values for identifiers. Instead, they use a 12-byte hexadecimal value composed from the following pieces of data:

• A 4-byte value representing the seconds since the Unix epoch,

• a 3-byte machine identifier,

• a 2-byte process id, and

• a 3-byte counter, starting with a random value.

This results in our objects having identifiers that resemble the following:

ObjectId(5b5615914434ad438bf3ea43)

Insert Values

- Create a collection and insert values:

`db.<collection name>.insertOne (<values>`

- The `_id` field is automatically generated by MongoDB.
- This is like the primary key field that is used to uniquely identify a record or a document.
- The `_id` or `Object ID` generated is a 12-byte hexadecimal value.

```
> db.inventory.insertOne(
...     { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("60dc389d3d6a77f598d222f0")
}
```

Slide Note

```
> db.inventory.insertMany([
...     { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
...     { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
...     { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
...     { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
...     { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
... ]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("60dc39763d6a77f598d222f1"),
                ObjectId("60dc39763d6a77f598d222f2"),
                ObjectId("60dc39763d6a77f598d222f3"),
                ObjectId("60dc39763d6a77f598d222f4"),
                ObjectId("60dc39763d6a77f598d222f5")
        ]
}
```

Inserting Multiple Records

- Insert multiple records into a collection using the command below:

  `db.<collection name>.insertMany()`

- An array of documents is passed to the method.

Slide Note

Querying a Collection

- Displaying the data from a collection:

```
db.<collection name>.find()
```

- Displaying data in a formatted manner:

```
db.<collection name>.find().pretty()
```

```
> db.inventory.find()
{ "_id" : ObjectId("60dc389d3d6a77f598d222f0"), "item" : "canvas", "qty" : 100, "tags" : [ "cotton" ], "size" : { "h" : 28, "w" : 35.5, "uom" : "cm" } }
> db.inventory.find().pretty()
{
        "_id" : ObjectId("60dc389d3d6a77f598d222f0"),
        "item" : "canvas",
        "qty" : 100,
        "tags" : [
                "cotton"
        ],
        "size" : {
                "h" : 28,
                "w" : 35.5,
                "uom" : "cm"
        }
}
```

Slide Note

Conditional Querying

- Specify Equality – Find all products that have a status as ‘D’.
  - `db.inventory.find( { status: "D" } )`
- Specify Conditions Using Query Operators – Find all products that have the status ‘A’ or ‘D’.
  - `db.inventory.find( { status: { $in: [ "A", "D" ] } } )`
- Specify AND Conditions – Find all products that have status ‘A’ and quantity less than 30.
  - `db.inventory.find( { status: "A", qty: { $lt: 30 } } )`
- Specify OR Conditions – Find all products that have status ‘A’ or quantity less than 30.
  - `db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )`

For query operators refer to the Mongo official document.

Slide Note

Delete Documents

- To delete the documents in a collectio n, the commands below can be used:
  - `db.collection.deleteOne(<filter>)`
    - Removes the first document that matches the filter.
  - `db.collection.deleteMany(<filter>)`
    - Removes all documents that match the filter from a collection.
- `<filter>` - Specifies deletion criteria using the query operators.