Document-Oriented Modelling

Agenda

- **Document-Oriented Data Models**

- Embedded (denormalised) models

- Normalised models

Document-Oriented Data Models

"The key challenge in data modelling is balancing the needs of the application, the performance characteristics of the database engine, and the data retrieval patterns.

When designing data models, always consider the application usage of the data (i.e. queries, updates, and processing of the data) as well as the inherent structure of the data itself."

Flexible schema

- Unlike SQL databases, where you must determine and declare a table's schema before inserting data, MongoDB's collections, by default, does not require its documents to have the same schema. That is:

  - The documents in a single collection do not need to have the same set of fields and the data type for a field can differ across documents within a collection.

  - To change the structure of the documents in a collection, such as add new fields, remove existing fields, or change the field values to a new type, update the documents to the new structure.

Flexible schema

- This flexibility facilitates the mapping of documents to an entity or an object. Each document can match the data fields of the represented entity, even if the document has substantial variation from other documents in the collection.

- In practice, however, the documents in a collection share a similar structure, and you can enforce document validation rules for a collection during update and insert operations.

Document Structure

- The key decision in designing data models for MongoDB applications revolves around the structure of documents and how the application represents relationships between data.

- There are two options:
  - **Embedded (or Denormalised) Data Model:** MongoDB allows related data to be embedded within a single document.
  - **Normalised Data Model:** Create references to capture the relationships between related documents.

Agenda

- Document-Oriented Data Models

- Embedded (denormalised) models

- Normalised models

Embedded (denormalised) models

- Embedded documents capture relationships between data by storing related data in a single document structure.

- MongoDB documents make it possible to embed document structures in a field or array within a document.

- These *denormalised* data models allow applications to retrieve and manipulate related data in a single database operation.

Embedded (denormalised) models

```
{
    _id: <ObjectId1>,
    username: "123xyz",
    contact: {
                phone: "123-456-7890",
                email: "xyz@example.com"
            },

    access: {
            level: 5,
            group: "dev"

            }
}
```

Embedded sub-document

Embedded sub-document

Embedded (denormalised) models

- In general, use embedded data models when:
  - You have "contains" relationships between entities.
    - See [Model One-to-One Relationships with Embedded Documents](#).
  - You have one-to-many relationships between entities. In these relationships the "many" or child documents always appear with or are viewed in the context of the "one" or parent documents.
    - See [Model One-to-Many Relationships with Embedded Documents](#).

Agenda

- Document-Oriented Data Models

- Embedded (denormalised) models

- Normalised models

Normalised data models

- References store the relationships between data by including links or *references* from one document to another.

- Applications can resolve these references to access the related data.

- Broadly, these are *normalised* data models.

Normalised data models
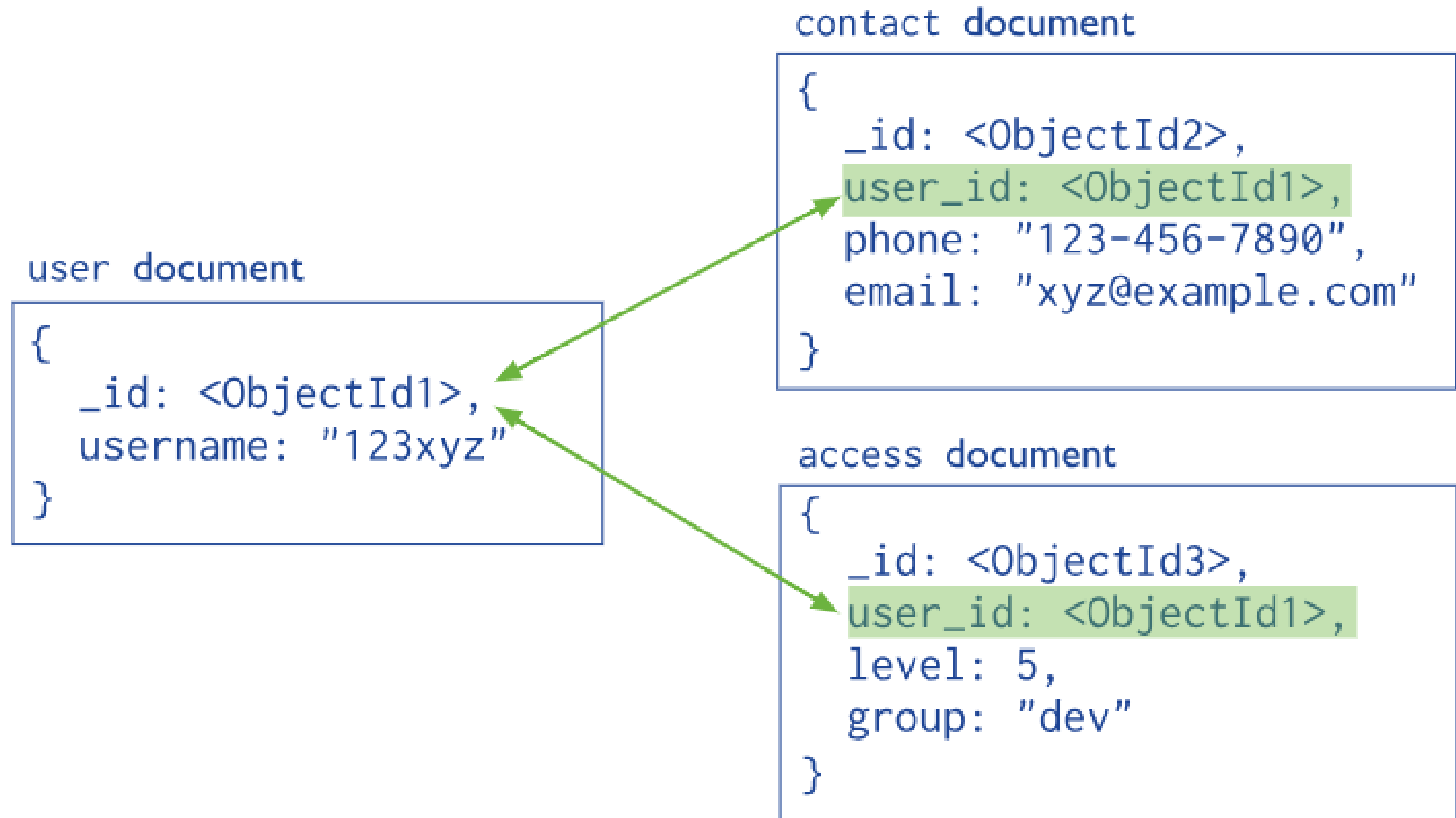
Normalised data models

- In general, use normalised data models:
  - when embedding would result in duplication of data but would not provide sufficient read performance advantages to outweigh the implications of the duplication.
  - to represent more complex many-to-many relationships.
  - to model large hierarchical data sets.

Normalised data models

- References provides more flexibility than embedding. However, client-side applications must issue follow-up queries to resolve the references. In other words, normalised data models can require more round trips to the server.

    - See [Model One-to-Many Relationships with Document References](#) for an example of referencing.