UNIVERSITETI I EVROPËS JUGLINDORE
УНИВЕРЗИТЕТ НА ЈУГОИСТОЧНА ЕВРОПА
SOUTH EAST EUROPEAN UNIVERSITY

FACULTY OF CONTEMPORARY SCIENCES AND TECHNOLOGIES

Course: NoSql

Project:

DENTAL CLINIC - MIGRATION FROM RELATIONAL TO NOSQL DATABASE USING MONGODB

[Student]
Era Emurli
Jona Sela

[Profesori]
MSc. Benjamin Besimi

CONTENTS

CONTENTS 2
1. INTRODUCTION 3
1.1 Project Overview 3
1.2 Objectives 3
2. Relational Database Design and Data Modeling 4
2.1 ER Diagram and Schema Overview 4
2.2 Table Descriptions, Relationships, Data Types, Integrity Constraints 5
3. Data Population 13
3.1 Sample Records per Table 13
3.2 SQL Server Data 13
4. Choice of NoSQL Database 16
4.1 Selected NoSQL Database - MongoDB 16
4.2 Comparison with Redis and Cassandra 16
4.3 Justification of Choice Based on Use Case 16
5. NoSQL Database Modeling 17
5.1 MongoDB Document Structure 17
5.2 Mapping from Relational to NoSQL 19
5.3 Design Decisions for Embedding vs Referencing 19
6. Data Migration Process 20
6.1 Migration Strategy Overview 20
6.2 Python Script for Migration 20
6.3 Data Transformation Techniques 20
6.4 Error Handling and Data Validation 21
6.4 Output 21
7. Conclusion 22

1. INTRODUCTION

1.1 Project Overview

This project focuses on the migration of data from a traditional relational database (SQL Server) to a NoSQL database (MongoDB). The aim is to explore and apply concepts of both relational and non-relational database systems by modeling, populating, and programmatically migrating data. The relational database used in this project simulates a real-world **Dental Clinic** system, including patients, employees, appointments, bills, medical history, and insurance data. The project demonstrates how structured tabular data can be transformed into flexible, document-based data suited for NoSQL environments.

1.2 Objectives

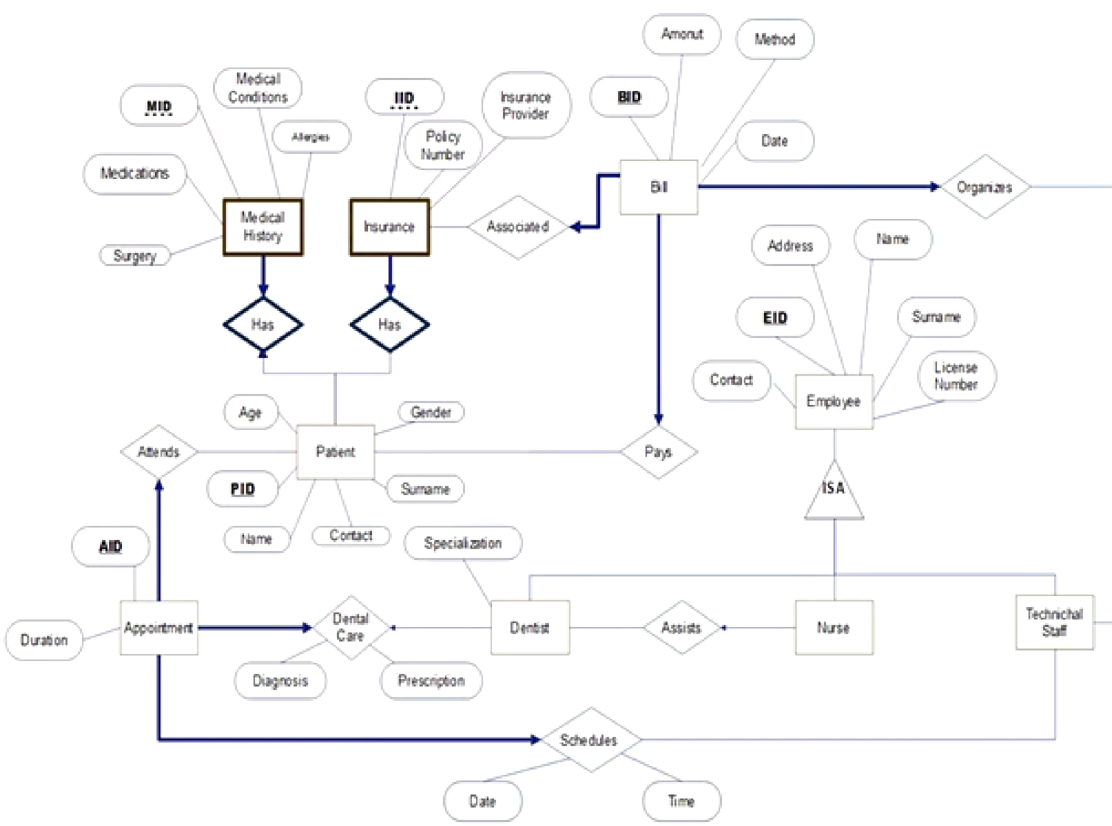
The primary objective of our project is to demonstrate the understanding and practical implementation of both relational and non-relational database systems through the design and migration of a complete Dental Clinic system. This involves creating a normalized relational database populated with meaningful and interrelated data, followed by the selection of a suitable NoSQL database - MongoDB - based on its scalability, performance, and compatibility with document-based data structures. The project also aims to develop an equivalent NoSQL schema to represent the same data in a document-oriented format. A Python-based migration script is implemented to handle the transformation and transfer of data from SQL Server to MongoDB, ensuring that data types and structures are accurately mapped. The entire workflow is documented in detail, and the final product is presented through a demonstration to highlight the outcomes and learning achievements.

2. Relational Database Design and Data Modeling

The relational database for the Dental Clinic system was designed using SQL Server, focusing on a normalized structure that ensures data integrity, minimizes redundancy, and represents real-world entities accurately. The schema includes eight core tables: *Patient, Employee, Dentist, Nurse, TechStaff, Appointment, Bill, MedicalHistory, and Insurance*. Each table was designed with appropriate data types and constraints, such as primary keys, foreign keys, unique constraints, and check constraints. These design choices reflect a real-world clinical environment and enable effective data management.

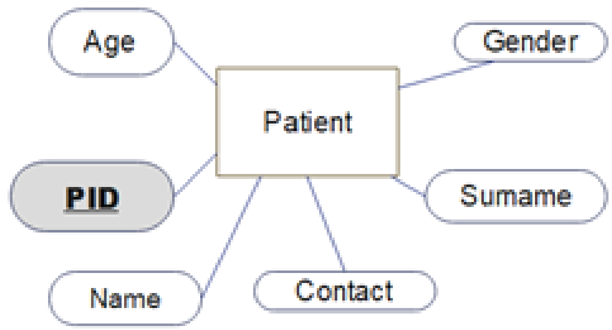
2.1 ER Diagram and Schema Overview

The tables in the relational schema are connected using foreign keys to enforce referential integrity. For example, the Appointment table has foreign keys to the Patient, Dentist, and TechStaff tables, ensuring that each appointment is associated with valid existing records. The Bill table references Patient, Insurance, and TechStaff. Weak entities like MedicalHistory and Insurance are linked to Patient using composite primary keys and unique constraints. Various constraints such as CHECK constraints (e.g., gender must be 'M' or 'F'), NOT NULL constraints, and CASCADE options for updates and deletions were applied to ensure robust data integrity throughout the database.

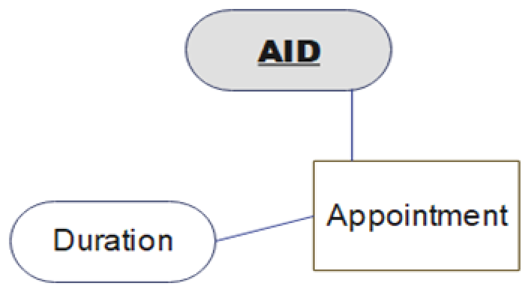


2.2 Table Descriptions, Relationships, Data Types, Integrity Constraints

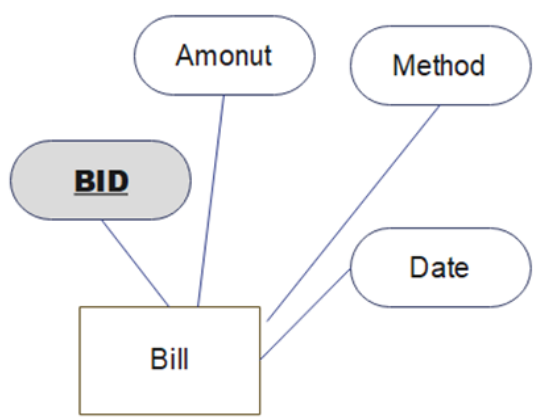
The first entity type is **Patient** and its attributes are PID, Name, Surname, Gender, Contact and Age. The primary key in this entity type is PID.



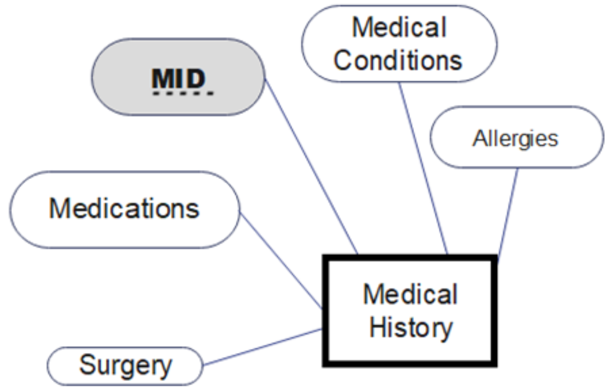
The second entity type is **Appointment** where its attributes are AID and Duration, at the same time as the primary key of the entity we will have AID.



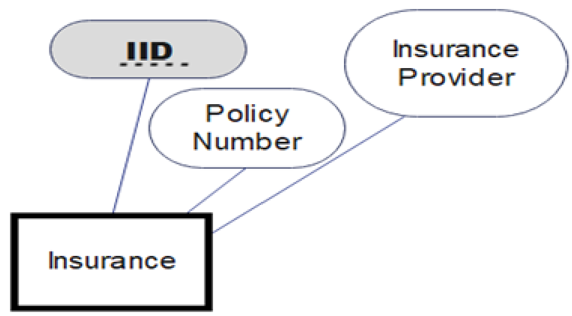
The third entity type is **Bill** where its attributes are BID, amount, method and date. The BID attribute is the primary key of the entity type.



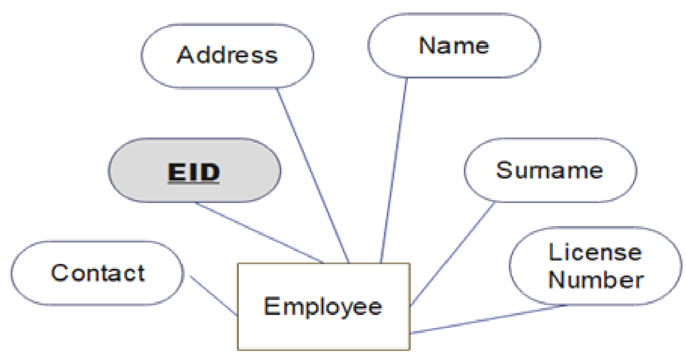
The fourth entity type is **Medical History** which is defined as a weak entity and cannot be uniquely identified by its attributes alone, therefore we must use a foreign key in relation to its attributes to create a primary key. The group of attributes that can uniquely identify a weak entity is called a partial key or we also encounter it as a discriminator. Where as its attributes we will have MID, Medications, Surgery, Medical Conditions and Allergies.



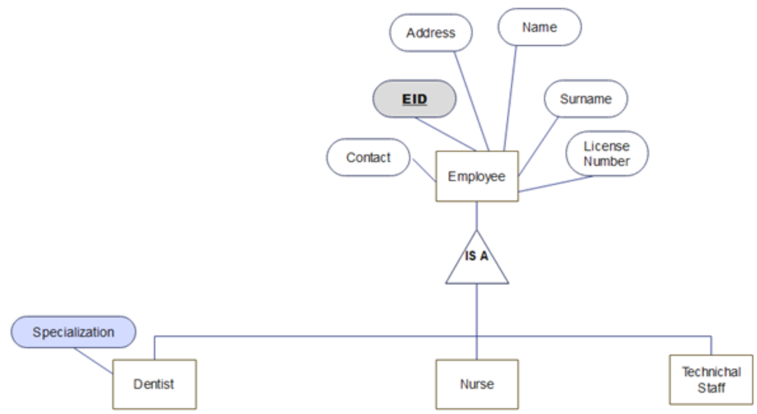
The fifth entity type is **Insurance**, which is also presented as a weak entity similar to the above entity, which also uses a partial key which is IID. Its attributes will be IID, Policy Number and Insurance Provider.



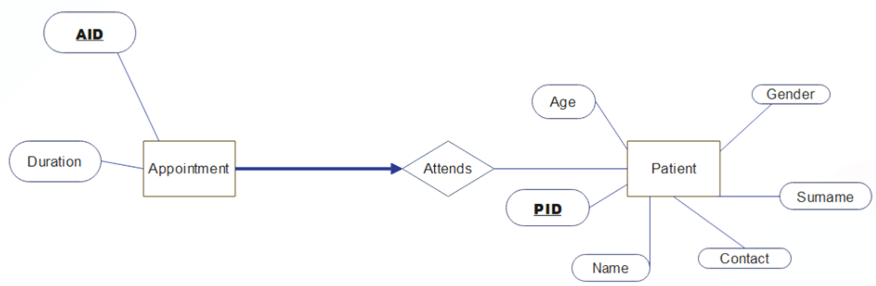
The sixth entity type is **Employee**, where its attributes will be EID, Name, Surname, License Number, Address, Contact. EID is used as the primary key.



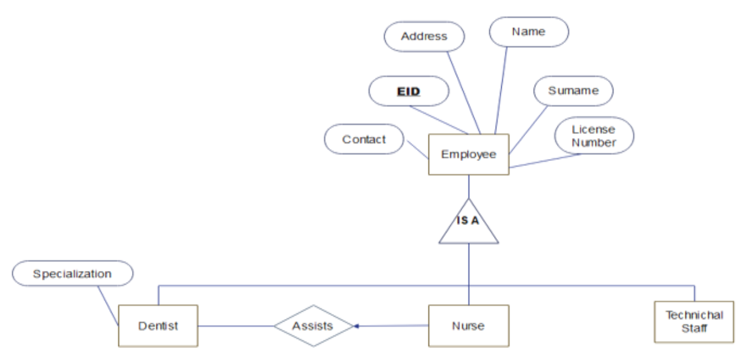
However, based on the description of our Clinic, we have concluded that employees can be dentists, nurses and technical staff. **Therefore, the Employee entity type can be divided into three subclasses** and this entity type can be best represented in the hierarchy. The three subclasses Dentist, Nurse and Technical Staff will inherit all the attributes of Employee even though Dentist has its own attribute, which is Specialization, which attribute belongs only to that entity and not to other subclasses.



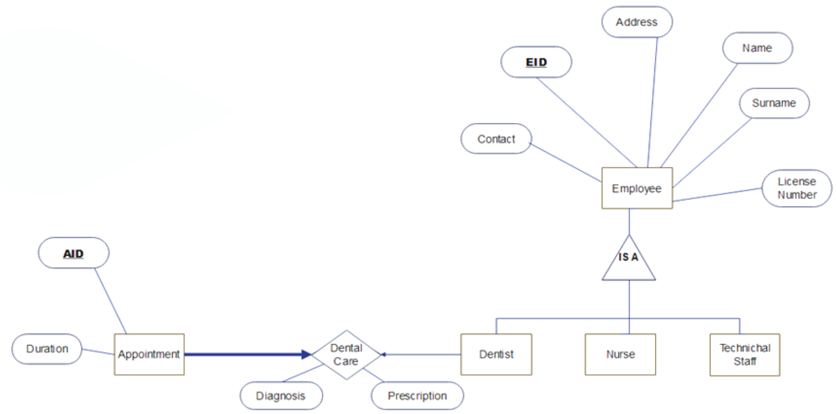
***In this step* we will define the relationships between the entity types. The first relationship is between Patient and Appointment,** where Patient has PID (patient identification number) as the primary key and other attributes while Appointment has AID (appointment identification number) as the primary key and other attributes.



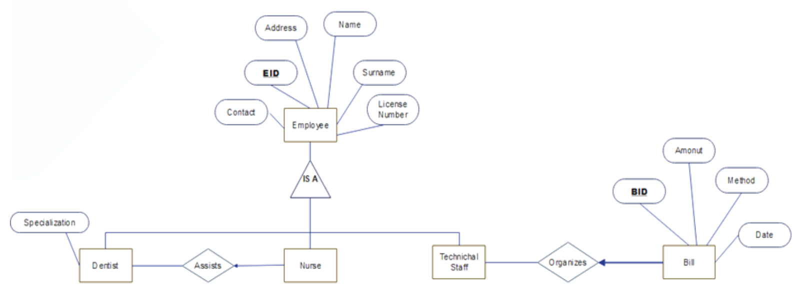
The second relationship is between Dentist and Nurse, where both entity types inherit EID (employee identification number) from the superclass Employee, while Dentist has Specialization as its own attribute which belongs only to him. The relationship is called Assists because Nurse assists Dentist during medical interventions.



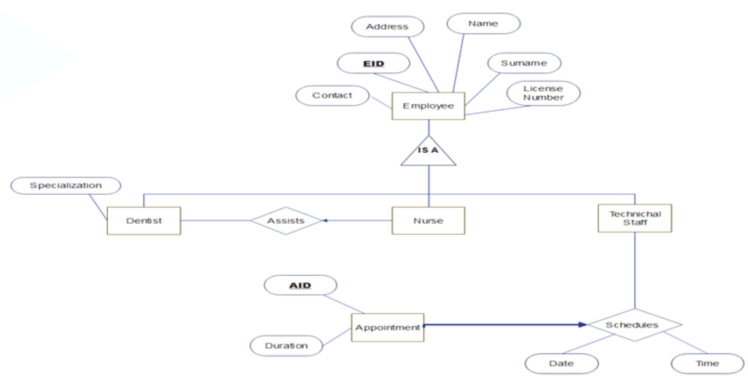
The third relationship is between Appointment and Dentist, where Appointment has AID as the primary key while Dentist has the primary key EID inherited from Employee as the primary key. In this relationship only the subclass Dentist is a participant in the relationship although the entire hierarchical structure seems to participate in it. Also the relationship between the two types has its own additional attributes such as Diagnosis and Prescription.



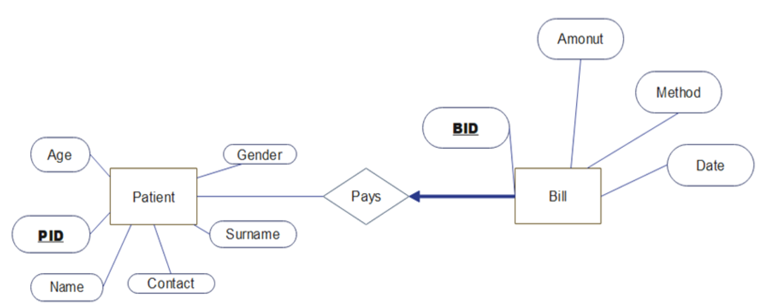
The fourth relationship is between Technical Staff and Bill, where Technical Staff as the primary key has EID inherited from Employee and Bill as the primary key has BID (bill identification number) and other attributes such as Amount, Method, Date.



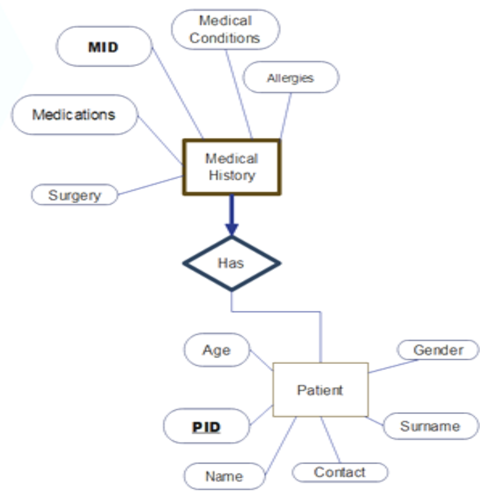
The fifth relationship is between Technical Staff and Appointment, where Technical Staff inherits EID as the primary key while Appointment has AID as the primary key and Duration as a supplementary attribute. The relationship is named Schedules and has Date and Time as the relationship attributes.



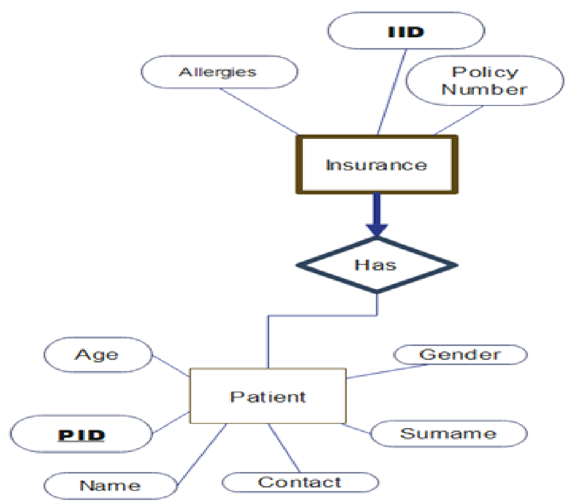
The sixth relationship is between Patient and Bill, where Patient has a primary key PID and supplementary attributes Name, Surname, Gender, Contact and Age, while Bill has a primary key BID and its supplementary attributes are Amount, Method, Date. The relationship is called Pays.



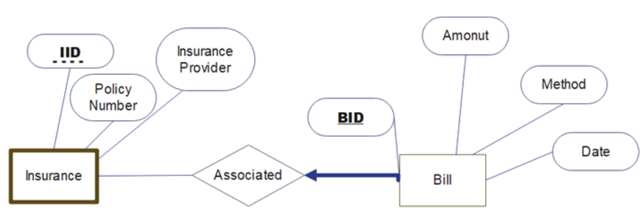
The seventh relationship is between Medical History and Patient, where Patient as the primary key has PID and supplementary attributes Name, Surname, Gender, Contact and Age, while Medical History as the primary key has MID and supplementary attributes Surgery, Medications, Medical Conditions and Allergies.



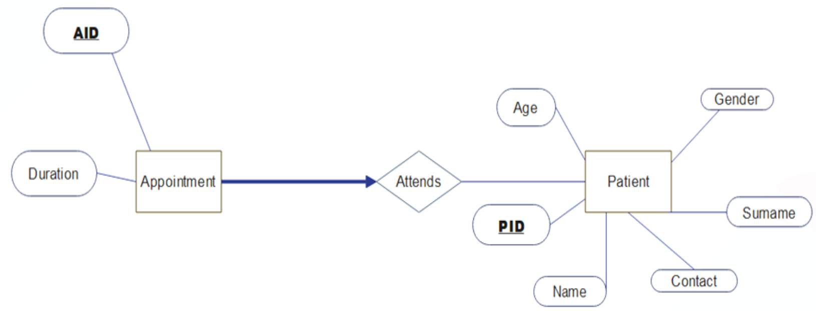
The eighth relationship is between Insurance and Patient, where Insurance has IID as its primary key and Allergies and Policy Number as its supplementary attributes, while Patient has PID as its primary key and Name, Surname, Gender, Contact, and Age as its supplementary attributes.



The ninth relationship is between Insurance and Bill, where Insurance has the primary key IID and its supplementary attributes are Allergies and Policy Number, while Bill has the primary key BID and its supplementary attributes are Amount, Method, Date. The relationship is called Associated.

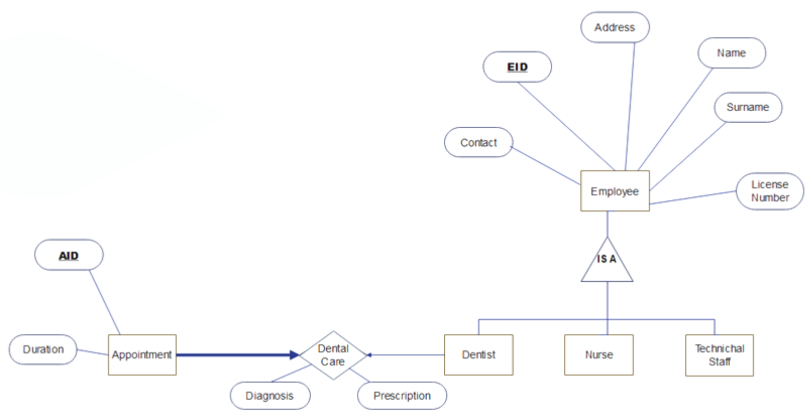


***In this step*, we will define the key and participation obligations in the relationships between entities**. Based on the problem description where Patient has a relation attends to Appointment in Ordinance, then the Appointment entity has both a key obligation and a participation obligation in the relationship because an Appointment belongs to only one Patient and exists only as a dependency of the Patient entity. On the other hand, the Patient entity has no obligation because a Patient can have many Appointments.



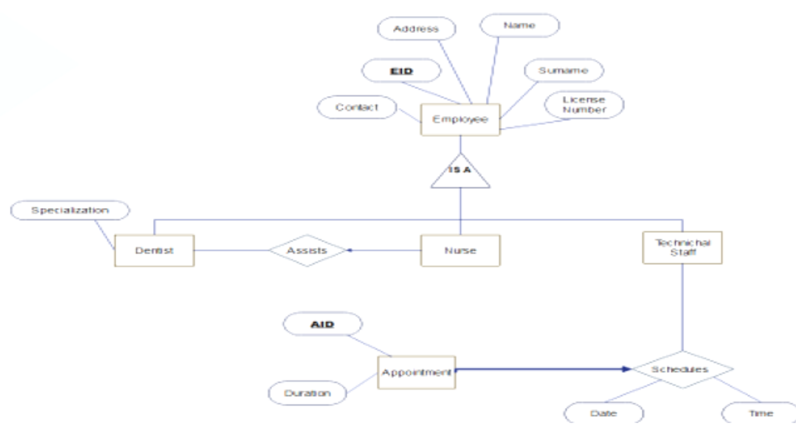
Cardinality: 1:M (one-to-many).

In the relationship between Appointment and Dentist which is Dental Care where the Dentist diagnoses and writes a prescription for the patient through Appointment. So the Appointment entity also has both a key obligation and a participation obligation, so Appointment belongs to only one Dentist and its existence depends on the Dentist entity. On the other hand, the Dentist entity has no obligation in the relationship.



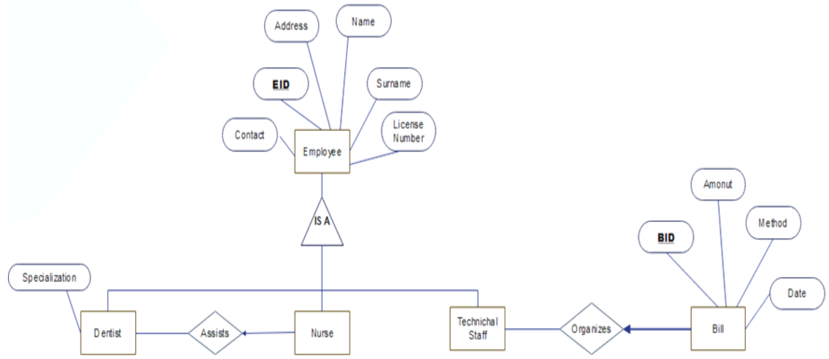
Cardinality: 1:1 (one-to-one).

In the relationship between Appointment and Technical Staff called Schedules where Technical Staff plans schedules for Appointment where also defines Date and Time. Appointment has obligation in key and in participation because an Appointment can be organized only by one Tech Staff and the Tech Staff entity can organize many Appointments.



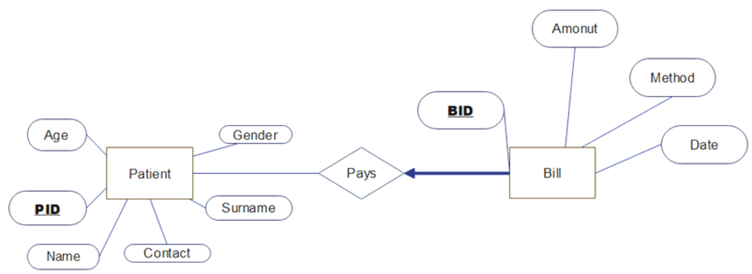
Cardinality: 1:M (one-to-many).

In the relationship between Bill and Technical Staff there is a relationship between where Technical Staff organizes more Bills. But Bills has a key obligation and a participation obligation because a Bill can only be organized by a Technical Staff and the entity Bill will only exist if there is a Technical Staff entity who will organize them. On the other hand Technical Staff has no key obligation and no participation obligation.



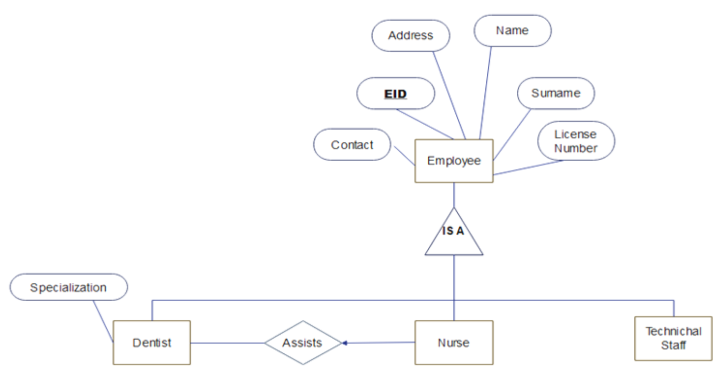
Cardinality: 1:M (one-to-many).

In the relationship between Patient and Bill which is the Pays relationship, where Patient pays Bill for the Dental Prescription service where Bill refers to at most one patient and the entity Bill has a participation obligation and a key obligation. Bill belongs to only one patient and its existence depends on the entity Patient. On the other hand, the entity Patient has no key obligation or participation obligation.



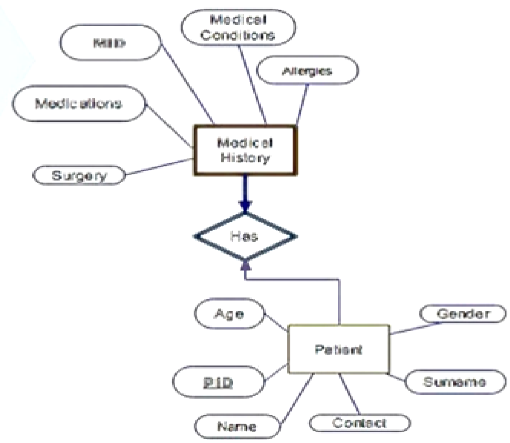
Cardinality: 1:M (one-to-many).

There is a relationship between Nurse and Dentist, where Nurse can be an assistant and assists at most one Dentist but at the same time a Dentist can have more than one Nurse as an assistant or no Nurse as an assistant. So Nurse has a key constraint in the relationship. On the other hand the entity Dentist has no kind of constraint in the relationship.



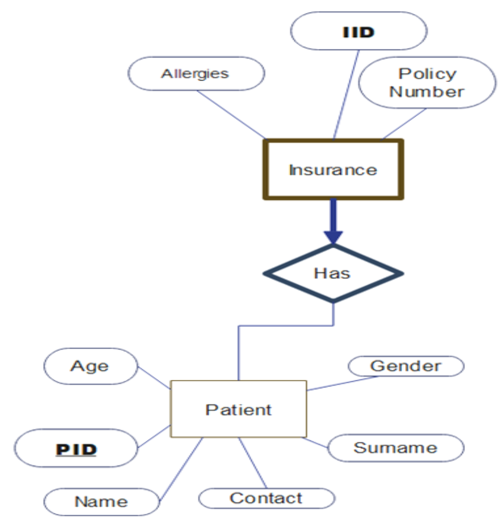
Cardinality: M:1 (many-to-one).

The relationship between Medical History and Patient is a relationship where a Medical History belongs to only one Patient as a result of the fact that a Medical History cannot belong to two different patients, so Medical History has obligations in the key and in participation in the relationship. This is a continuation of what Medical History from the first step was presented as a weak entity and has exactly the same obligations. On the other hand, the Patient entity has no obligations in the relationship.



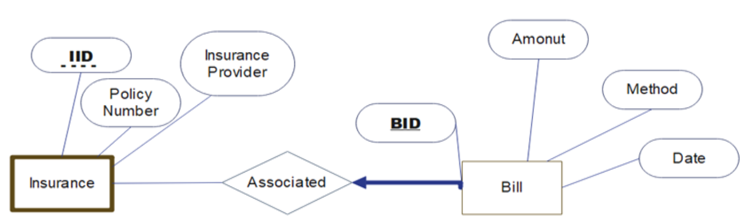
Cardinality: 1:1 (one-to-one).

The relationship between Insurance and Patient is a relationship where an Insurance belongs to only one Patient as a result of the fact that the same Insurance cannot belong to two different patients, so Insurance has a key and participation obligation in the relationship. This is a continuation of Insurance from the first step, it is presented as a weak entity and has exactly the same obligations. On the other hand, the Patient entity has no obligation in the relationship.



Cardinality: 1:M (one-to-many).

The relationship between Bill and Insurance is an "associated" relationship where a Bill can be covered by more than one Insurance (insurance) so there can be more health insurance that a payslip can cover. So we conclude that the entity Bill has a key obligation and a participation obligation in the relationship, whereas Insurance has no obligation in the relationship and no participation obligation in the relationship.



Cardinality: 1:M (one-to-many).

3. Data Population

3.1 Sample Records per Table

Each table in the relational database was populated with meaningful sample data to simulate a real-world dental clinic scenario. For example, the **Patient** table includes over 20 patients with realistic names, ages, genders. The **Appointment** table links patients with dentists and technical staff for various procedures, each with specified durations, diagnoses, and timestamps. The **Bill** table contains transaction records for each patient, including payment method and amount. Similarly, the **MedicalHistory** table captures medical backgrounds like allergies or surgeries, and the **Employee**, **Dentist**, **Nurse**, and **TechStaff** tables are populated with staff members and their corresponding roles.

All tables were filled with at least 15–20 records to ensure sufficient data for testing relationships, constraints, and the eventual migration to the NoSQL system.

3.2 SQL Server Data

```sql
SELECT * FROM Patient;
SELECT * FROM Employee;
SELECT * FROM TechStaff;
SELECT * FROM Bill;
SELECT * FROM Dentist;
SELECT * FROM MedicalHistory;
SELECT * FROM Appointment;
SELECT * FROM Insurance;
SELECT * FROM Dentist;
```

68 %

Results | Messages

	PID	Pname	Psurname	Ppage	Pgender	Pcontact
1	0001	Jona	Sela	20	F	777-777-777
2	0002	Era	Sela	21	F	222-222-222
3	0003	Ajeta	Dauti	21	F	444-444-444
4	0004	Gjilizar	Zhuta	20	F	333-333-333
5	0005	Jon	Ziba	25	M	111-111-111
6	0006	Jeta	Sela	20	F	666-666-666
7	0007	Lin	Nushi	50	M	345-983-123
8	0008	Melisa	Kaba	23	F	999-999-999
9	0009	Gerti	Oda	60	M	504-503-403
10	0010	Gerta	Ismaili	28	F	700-893-010
11	0011	Jana	Sela	20	F	777-777-777
12	0012	Erra	Sela	21	F	222-222-222
13	0013	Aieta	Dauti	21	F	444-444-444
14	0014	Gilizar	Zhuta	20	F	333-333-333
15	0015	Ron	Ziba	25	M	111-111-111
16	0016	Leta	Sela	20	F	666-666-666
17	0017	Len	Nushi	50	M	345-983-123
18	0018	Merlisa	Kaba	23	F	999-999-999
19	0019	Gert	Oda	60	M	504-503-403
20	0020	Gerald	Ismaili	28	F	700-893-010

```sql
SELECT * FROM Patient;
SELECT * FROM Employee;
SELECT * FROM TechStaff;
SELECT * FROM Bill;
SELECT * FROM Dentist;
SELECT * FROM MedicalHistory;
SELECT * FROM Appointment;
SELECT * FROM Insurance;
SELECT * FROM Dentist;
```

68 %

Results | Messages

	EID	Ename	Esurname	Eaddress	Econtact	Elisencenum
1	D0001	Lyra	Vita	Toronto, Canada, Red st	534-503-403	F1111
2	D0002	Kela	Zhuta	Marks Engels st	604-453-403	A7777
3	D0003	Albulena	Jonuzi	722 East St	004-503-8893	X1717
4	D0004	Kaltrina	Bilali	111 Beka St	504-503-097	E2032
5	D0005	Ardian	Vrenezi	202 Elz St	500-233-403	D0393
6	D0006	Hanife	Vinca	303 Orbit St	474-503-403	L7070
7	D0007	Armend	Jakupi	404 Star St	564-903-403	D1010
8	D0008	Eva	Poposka	505 Moon St	504-503-403	K2322
9	D0009	Ajan	Zuta	606 New St	777-503-403	T7373
10	D0010	Leon	Lila	707 Stella St	564-666-403	P6661
11	D0011	Lira	Vita	Toronto, Canada, Red st	534-503-403	F1111
12	D0012	Keta	Zhuta	Marks Engels st	604-453-403	A7777
13	D0013	Arlbulena	Jonuzi	722 East St	004-503-8893	X1717
14	D0014	Katalea	Bilali	111 Beka St	504-503-097	E2032
15	D0015	Adrian	Vrenezi	202 Elz St	500-233-403	D0393
16	D0016	Anife	Vinca	303 Orbit St	474-503-403	L7070
17	D0017	Admend	Jakupi	404 Star St	564-903-403	D1010
18	D0018	Eta	Poposka	505 Moon St	504-503-403	K2322
19	D0019	Ajani	Zuta	606 New St	777-503-403	T7373
20	D0020	Leoni	Lila	707 Stella St	564-666-403	P6661

```sql
SELECT * FROM Patient;
SELECT * FROM Employee;
SELECT * FROM TechStaff;
SELECT * FROM Bill;
SELECT * FROM Dentist;
SELECT * FROM MedicalHistory;
SELECT * FROM Appointment;
SELECT * FROM Insurance;
SELECT * FROM Dentist;
```

68 %

Results | Messages

	TID
1	T0001
2	T0002
3	T0003
4	T0004
5	T0005
6	T0006
7	T0007
8	T0008
9	T0009
10	T0010
11	T0011
12	T0012
13	T0013
14	T0014
15	T0015
16	T0016
17	T0017
18	T0018
19	T0019
20	T0020

```sql
SELECT * FROM Patient;
SELECT * FROM Employee;
SELECT * FROM TechStaff;
SELECT * FROM Bill;
SELECT * FROM Dentist;
SELECT * FROM MedicalHistory;
SELECT * FROM Appointment;
SELECT * FROM Insurance;
SELECT * FROM Dentist;
```

68 %

Results | Messages

	BID	Bamount	Bdate	BMethod	PID	IID	TID
1	B0001	111.00	2024-01-05	Cash	0001	I0001	T0001
2	B0007	111.00	2024-01-05	Cash	0001	I0001	T0001
3	B0008	150.50	2024-05-06	CreditCard	0002	I0002	T0002
4	B0009	222.00	2024-02-07	DebitCard	0003	I0003	T0003
5	B0010	777.25	2024-01-05	Cash	0004	I0004	T0004
6	B0011	310.00	2024-06-01	CreditCard	0006	I0006	T0006
7	B0012	200.00	2024-06-02	Cash	0007	I0007	T0007
8	B0013	555.50	2024-06-03	DebitCard	0008	I0008	T0008
9	B0014	399.99	2024-06-04	CreditCard	0009	I0009	T0009
10	B0015	150.00	2024-06-05	Cash	0010	I0010	T0010
11	B0016	180.75	2024-06-06	DebitCard	0011	I0011	T0011
12	B0017	220.00	2024-06-07	Cash	0012	I0012	T0012
13	B0018	325.00	2024-06-08	CreditCard	0013	I0013	T0013
14	B0019	405.25	2024-06-09	Cash	0014	I0014	T0014
15	B0020	505.00	2024-06-10	CreditCard	0015	I0015	T0015
16	B0021	275.00	2024-06-11	DebitCard	0016	I0016	T0016
17	B0022	345.75	2024-06-12	Cash	0017	I0017	T0017
18	B0023	290.00	2024-06-13	CreditCard	0018	I0018	T0018
19	B0024	399.50	2024-06-14	DebitCard	0019	I0019	T0019
20	B0055	420.00	2024-08-09	CreditCard	0005	I0005	T0005

```sql
SELECT * FROM Patient;
SELECT * FROM Employee;
SELECT * FROM TechStaff;
SELECT * FROM Bill;
SELECT * FROM Dentist;
SELECT * FROM MedicalHistory;
SELECT * FROM Appointment;
SELECT * FROM Insurance;
SELECT * FROM Dentist;
```

68 %

Results | Messages

	DID	Dspecialization
1	D0001	Orthodontics
2	D0002	Endodontics
3	D0003	Pediatric Dentistry
4	D0004	Periodontics
5	D0005	Oral Surgery
6	D0006	Prosthodontics
7	D0007	Oral Pathology
8	D0008	Public Health Dentistry
9	D0009	Oral Radiology
10	D0010	Implantology
11	D0011	Cosmetic Dentistry
12	D0012	Geriatric Dentistry
13	D0013	Maxillofacial Surgery
14	D0014	Laser Dentistry
15	D0015	Restorative Dentistry
16	D0016	Special Needs Dentistry
17	D0017	Temporomandibular Disorders
18	D0018	Oral Medicine
19	D0019	Forensic Odontology
20	D0020	Hospital Dentistry

```sql
SELECT * FROM Patient;
SELECT * FROM Employee;
SELECT * FROM TechStaff;
SELECT * FROM Bill;
SELECT * FROM Dentist;
SELECT * FROM MedicalHistory;
SELECT * FROM Appointment;
SELECT * FROM Insurance;
SELECT * FROM Dentist;
```

68 %

Results | Messages

	IID	PolicyNum	Insurance_provider	PID
1	I0011	POL001	ABC Insurance	0001
2	I0022	POL002	XYZ Insurance	0002
3	I0033	POL003	DEF Insurance	0003
4	I0044	POL004	GHI Insurance	0004
5	I0055	POL005	JKL Insurance	0005

```sql
SELECT * FROM Patient;
SELECT * FROM Employee;
SELECT * FROM TechStaff;
SELECT * FROM Bill;
SELECT * FROM Dentist;
SELECT * FROM MedicalHistory;
SELECT * FROM Appointment;
SELECT * FROM Insurance;
SELECT * FROM Dentist;
```

68 %

Results Messages

	AID	Aduration	PID	TID	DID	Diagnosis	Perception	Date	Time
1	A0001	60 minutes	0004	T0001	D0001	Regular check-up	Braces	2024-05-05	10:00:00.0000000
2	A0002	85 minutes	0003	T0002	D0002	Tooth extraction	follow-up care	2024-05-06	11:00:00.0000000
3	A0003	70 minutes	0001	T0003	D0003	Dental cleaning	oral hygiene	2024-05-07	12:00:00.0000000
4	A0004	40 minutes	0004	T0004	D0004	Root canal treatment	post-treatment care	2024-05-08	13:00:00.0000000
5	A0005	90 minutes	0002	T0005	D0005	Regular check-up	Medication	2024-05-09	14:00:00.0000000
6	A0006	55 minutes	0005	T0006	D0006	Teeth whitening	Sensitivity warning	2024-05-10	09:00:00.0000000
7	A0007	60 minutes	0006	T0007	D0007	Cavity filling	Avoid sweets	2024-05-11	10:30:00.0000000
8	A0008	45 minutes	0007	T0008	D0008	X-ray examination	Clear scan	2024-05-12	11:45:00.0000000
9	A0009	75 minutes	0008	T0009	D0009	Tooth implant	Healing check	2024-05-13	14:00:00.0000000
10	A0010	80 minutes	0009	T0010	D0010	Gum treatment	Antibiotics needed	2024-05-14	15:15:00.0000000
11	A0011	65 minutes	0010	T0011	D0011	Dental bridge	Soft food advice	2024-05-15	09:30:00.0000000
12	A0012	90 minutes	0011	T0012	D0012	Regular check-up	Maintain hygiene	2024-05-16	10:00:00.0000000
13	A0013	40 minutes	0012	T0013	D0013	Braces adjustment	Pain relief tips	2024-05-17	11:00:00.0000000
14	A0014	50 minutes	0013	T0014	D0014	Retainer fitting	Usage instructions	2024-05-18	12:00:00.0000000
15	A0015	70 minutes	0014	T0015	D0015	Tooth polishing	Avoid coffee	2024-05-19	13:00:00.0000000
16	A0016	30 minutes	0015	T0016	D0001	Initial consultation	X-ray scheduled	2024-05-20	14:00:00.0000000
17	A0017	45 minutes	0016	T0017	D0002	Toothache	Painkillers prescribed	2024-05-21	15:00:00.0000000
18	A0018	35 minutes	0017	T0018	D0003	Wisdom tooth pain	Surgery scheduled	2024-05-22	16:00:00.0000000
19	A0019	85 minutes	0018	T0019	D0004	Mouth infection	Antibiotics treatment	2024-05-23	09:00:00.0000000
20	A0020	100 minutes	0019	T0020	D0005	Full dental exam	Good condition	2024-05-24	10:00:00.0000000

```sql
SELECT * FROM Patient;
SELECT * FROM Employee;
SELECT * FROM TechStaff;
SELECT * FROM Bill;
SELECT * FROM Dentist;
SELECT * FROM MedicalHistory;
SELECT * FROM Appointment;
SELECT * FROM Insurance;
SELECT * FROM Dentist;
```

68 %

Results Messages

	MID	Mmed	Msurgery	Mcodnition	Mallergies	PID
1	M0001	Ibuprofen	NULL	Toothache	None	0001
2	M0002	Lisinopril	NULL	Gum disease (Periodontitis)	Penicillin	0002
3	M0003	Salbutamol	NULL	Dry mouth due to inhaler use	Dust	0003
4	M0004	NULL	Tooth extraction	Severe tooth decay	None	0004
5	M0005	Metformin	NULL	Gingivitis (linked to diabetes)	Insulin	0005
6	M0006	Ibuprofen	Wisdom tooth removal	Impacted wisdom teeth	None	0006
7	M0007	Paracetamol	NULL	Tooth abscess	None	0007
8	M0008	Amoxicillin	NULL	Periodontitis	Penicillin	0008
9	M0009	NULL	Root canal	Pulpitis	Latex	0009
10	M0010	Chlorhexidine	NULL	Gingivitis	None	0010
11	M0011	NULL	NULL	Tooth decay	Peanuts	0011
12	M0012	Metronidazole	Dental implant	Tooth loss	None	0012
13	M0013	Ibuprofen	NULL	Jaw pain (TMJ disorder)	None	0013
14	M0014	NULL	NULL	Bruxism (teeth grinding)	None	0014
15	M0015	Clindamycin	NULL	Post-extraction infection	Sulfa drugs	0015

4. Choice of NoSQL Database

4.1 Selected NoSQL Database - MongoDB

For this project, our selected NoSQL database is **MongoDB**. MongoDB is a document-oriented database that stores data in flexible, JSON-like documents. It supports embedded documents and arrays, making it highly suitable for hierarchical data structures. Its schema-less nature allows the system to handle diverse patient records, appointments, and related entities without needing strict relational constraints.

4.2 Comparison with Redis and Cassandra

Redis is an in-memory key-value store known for its speed and simplicity, but it is not ideal for storing complex, structured data with relationships like those in a dental clinic system. Redis lacks advanced querying and data modeling capabilities required for this type of application.

Cassandra is a column-family store best suited for large-scale, distributed applications with heavy write throughput. While it offers excellent scalability, its data model is not ideal for representing the complex relationships between patients, appointments, and billing data.

In contrast, **MongoDB** supports:

- Nested documents for embedding related data (e.g., appointments within a patient document),
- A rich query language,
- Easy scalability,
- And flexible schemas, making it better suited for our healthcare use case.

4.3 Justification of Choice Based on Use Case

The **dental clinic** database contains patient data that is closely linked with appointments, medical histories, and bills. These relationships can be embedded within a single document in MongoDB, reducing the need for expensive joins and improving read efficiency.

MongoDB's document model allows storing all relevant patient data - including appointments and bills — in a single nested structure, mirroring real-world data usage. This model also supports future growth, where new fields (e.g., insurance claims, prescriptions) can be added without altering a rigid schema.

Therefore, **MongoDB** is the most appropriate NoSQL solution due to its **flexibility, natural fit for hierarchical data, and the best querying capabilities**.

5.NoSQL Database Modeling

5.1 MongoDB Document Structure

In the NoSQL database, data is stored using **MongoDB's document-oriented model**. The primary document is the **Patient** document, which includes not only the basic patient information but also related data from other tables such as appointments, bills, medical history, and insurance. Each document uses a JSON-like structure that encapsulates all relevant information for a single patient.

_id: ObjectId('6849b91402de26a11d37343f')
pid : "0001"
name : "Jona"
surname : "Sela"
age : 20
gender : "F"
contact : "777-777-777"
appointments : Array (1)
0: Object
AID : "A0003"
Aduration : "70 minutes"
PID : "0001"
TID : "T0003"
DID : "D0003"
Diagnosis : "Dental cleaning"
Perception : "oral hygiene"
Date : "2024-05-07"
Time : "12:00:00"
bills : Array (2)
0: Object
BID : "B0001"
Bamount : 111
Bdate : "2024-01-05"
BMethod : "Cash"
PID : "0001"
IID : "I0001"
TID : "T0001"
1: Object
BID : "B0007"
Bamount : 111
Bdate : "2024-01-05"
BMethod : "Cash"
PID : "0001"
IID : "I0001"
TID : "T0001"
medical_history : Object
MID : "M0001"
Mmed : "Ibuprofen"
Msurgery : null
Mcodnition : "Toothache"
Mallergies : "None"
PID : "0001"
insurances : Array (1)
0: Object
IID : "I0011"
PolicyNum : "POL001"
Insurance_provider : "ABC Insurance"
PID : "0001"

Employees Collection

```
_id: ObjectId('6849b91502de26a11d373453')
EID : "D0001"
Ename : "Lyra"
Esurname : "Vita"
Eaddress : "Toronto, Canada, Red st"
Econtact : "534-503-403"
Elisencenum : "F1111"
```

```
_id: ObjectId('6849b91502de26a11d373454')
EID : "D0002"
Ename : "Kela"
Esurname : "Zhuta"
Eaddress : "Marks Engels st"
Econtact : "604-453-403"
Elisencenum : "A7777"
```

Dentists Collection

```
_id: ObjectId('6849b91502de26a11d373467')
DID : "D0001"
Dspecialization : "Orthodontics"
```

```
_id: ObjectId('6849b91502de26a11d373468')
DID : "D0002"
Dspecialization : "Endodontics"
```

```
_id: ObjectId('6849b91502de26a11d373469')
DID : "D0003"
Dspecialization : "Pediatric Dentistry"
```

```
_id: ObjectId('6849b91502de26a11d37346a')
DID : "D0004"
Dspecialization : "Periodontics"
```

```
_id: ObjectId('6849b91502de26a11d37346b')
DID : "D0005"
Dspecialization : "Oral Surgery"
```

Nurse Collection

```
_id: ObjectId('6849b91502de26a11d37347b')
NID : "N0001"
DID : "D0001"
```

```
_id: ObjectId('6849b91502de26a11d37347c')
NID : "N0002"
DID : "D0002"
```

```
_id: ObjectId('6849b91502de26a11d37347d')
NID : "N0003"
DID : "D0003"
```

```
_id: ObjectId('6849b91502de26a11d37347e')
NID : "N0004"
DID : "D0004"
```

Tech Staff Collection

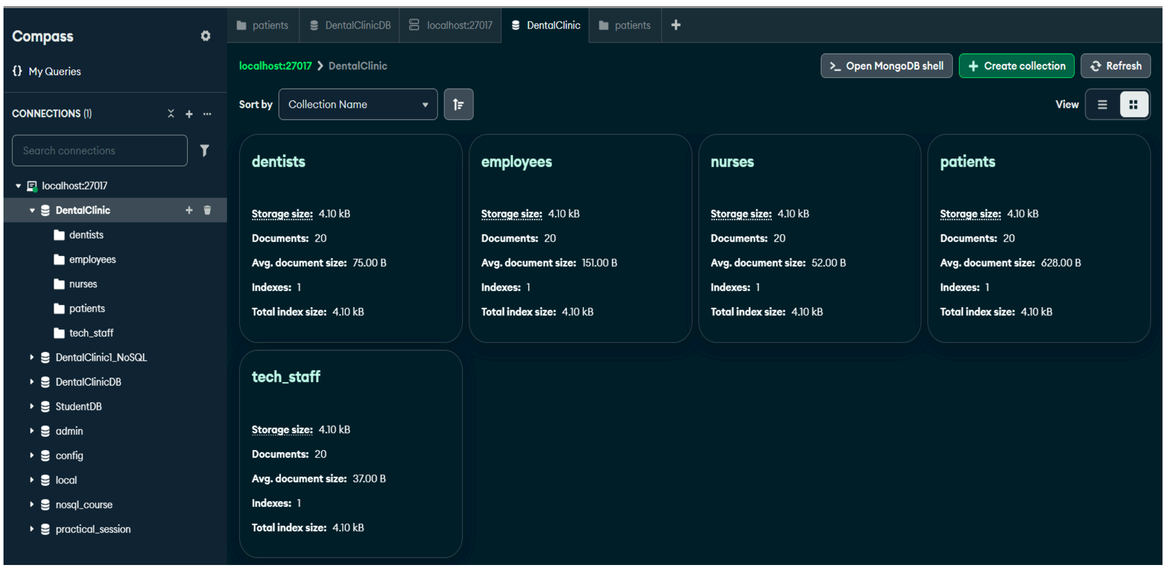
```
_id: ObjectId('6849b91502de26a11d37348f')
TID : "T0001"
```

```
_id: ObjectId('6849b91502de26a11d373490')
TID : "T0002"
```

5.2 Mapping from Relational to NoSQL

The relational database includes multiple normalized tables: Patient, Appointment, MedicalHistory, Bill, Insurance, Employee, etc. These were mapped to MongoDB documents as follows:

- The Patient table was the central entity and became the main document.
- Tables with a one-to-many relationship to Patient (like Appointment, Bill, Insurance) were embedded as arrays of sub-documents.
- The MedicalHistory table (1-to-1) was embedded as a nested sub-document.
- Other tables like Employee, Dentist, Nurse, and TechStaff were stored as separate collections, since they are not directly part of the patient's document and can be referenced if needed.



5.3 Design Decisions for Embedding vs Referencing

In designing the MongoDB schema, we made careful choices between embedding and referencing based on the relationships:

Embedded Documents:

- Used for data tightly coupled with the patient, such as appointments, bills, and medical history.
- Embedding improves read performance and keeps related data together.

Referenced Documents:

- Used for entities like Employee, Dentist, and Nurse, which are shared across multiple patients or records.
- Referencing avoids data duplication and supports consistency when employees change.

6. Data Migration Process

6.1 Migration Strategy Overview

The migration process involved transferring data from the relational SQL Server database to a NoSQL MongoDB database. The strategy was to extract data from normalized tables in SQL Server and restructure them into MongoDB's document-oriented model. The focus was on preserving relationships, ensuring data integrity, and creating embedded structures where applicable to optimize read performance.

The approach followed the ETL model - Extract, Transform, and Load. Data was extracted using SQL queries, transformed into appropriate document formats, and then loaded into MongoDB using Python and the pymongo library.

6.2 Python Script for Migration

The migration script was written in Python using pyodbc for connecting to the SQL Server and pymongo for inserting documents into MongoDB. The script:

- Connected to the SQL Server database using Windows authentication.
- Retrieved patient records and all their associated appointments, bills, insurance entries, and medical history using SQL queries.
- Assembled each patient's data into a single MongoDB document.
- Inserted each document into the patients collection.
- Migrated other tables such as Employee, Dentist, Nurse, and TechStaff into their own collections.
- Logging was implemented to track progress and catch errors during the migration process.

6.3 Data Transformation Techniques

- Several transformations were necessary to adapt relational data to MongoDB:
- Date and Time Conversion: SQL DATE and TIME types were converted into ISO 8601 format strings compatible with MongoDB.
- Decimal Values: Numeric data types like numeric(10,2) were cast to float to ensure compatibility with MongoDB's JSON-based structure.
- Embedding: One-to-many relationships such as Appointment, Bill, and Insurance were embedded as arrays within the patient document.
- Flattening: Related rows (from MedicalHistory) were flattened into a sub-document to simplify the document structure.
- These transformations helped preserve the meaning and usability of the data in a document-oriented context.

6.4 Error Handling and Data Validation

To ensure that the migration process was resilient and reliable, the script included:

- Try-Except Blocks: Wrapped the entire process in error-handling logic to catch SQL or connection errors.
- Logging: All actions and errors were logged using the logging module, allowing for easy debugging.
- Schema Checks: Before inserting data, key fields were validated (e.g., primary keys like PID).
- Data Type Safety: The convert_for_mongo() function ensured that all SQL Server data types were transformed into MongoDB-compatible types.

6.4 Output

```
Inserted patient 0001 into MongoDB.
Inserted patient 0002 into MongoDB.
Inserted patient 0003 into MongoDB.
Inserted patient 0004 into MongoDB.
Inserted patient 0005 into MongoDB.
Inserted patient 0006 into MongoDB.
Inserted patient 0007 into MongoDB.
Inserted patient 0008 into MongoDB.
Inserted patient 0009 into MongoDB.
Inserted patient 0010 into MongoDB.
Inserted patient 0010 into MongoDB.
Inserted patient 0011 into MongoDB.
Inserted patient 0012 into MongoDB.
Inserted patient 0013 into MongoDB.
Inserted patient 0014 into MongoDB.
Inserted patient 0015 into MongoDB.
Inserted patient 0016 into MongoDB.
Inserted patient 0017 into MongoDB.
Inserted patient 0018 into MongoDB.
Inserted patient 0019 into MongoDB.
Inserted patient 0020 into MongoDB.
✅ Full migration completed including bonus collections.
```

7. Conclusion

In this project, we successfully designed and implemented a relational database system for a dental clinic using SQL Server and then programmatically migrated the data to a NoSQL environment using MongoDB. The transition demonstrated our understanding of both data models, relational and document-based, and how real-world data with complex relationships can be transformed and represented in a flexible NoSQL structure.

The project involved:

- Creating and populating multiple interrelated tables (e.g., Patients, Employees, Bills, Appointments).
- Writing a Python-based migration script to convert and transfer data into MongoDB.
- Embedding nested documents such as appointments and bills within patient documents to better leverage MongoDB's schema flexibility.

Through this process, we learned the strengths and trade-offs of each database model. The relational database offers strong data integrity and structure, while the NoSQL document model provides flexibility and scalability, especially useful for modern applications.

We truly enjoyed working on this project. It gave us valuable hands-on experience and deeper insight into data modeling and database technologies. This experience has sparked our interest in the field, and we are considering further exploration or even a future career path related to data engineering or database systems.