Ethan Sanchez

schema_design.docx

Part 1: Analyze the MySQL Schema

In the ***emr*** MySQL schema, the tables (in *italics*) are:
- *visit*
- *visit_diagnosis*
- *diagnosis*
- *visit_procedure*
- *clinical_procedures*
- *visit_lab*
- *lab*
- *visit_symptom*
- *symptom*
- *patient*
- *provider*

The primary key ("PK")/foreign key ("FK") relationships and respective one-to-one ("1:1"), many-to-one ("M:1"), and many-to-many ("M:N") relationships are:
- The attribute 'visit_id': a FK of *visit_diagnosis* is the PK of *visit*. (The PK of *visit_diagnosis* is a composite key consisting of 'visit_id' and 'diagnosis_id'.)
  - The relationship between *visit_diagnosis* and *visit* is a M:1 mandatory-to-mandatory relationship where one *visit_diagnosis* can have many *visit*s, but one *visit* must only have one *visit_diagnosis*.
- The attribute 'diagnosis_id': a FK of *visit_diagnosis* is the PK of *diagnosis*.
  - The relationship between *visit_diagnosis* and *diagnosis* is a M:1 mandatory-to-mandatory relationship where one *visit_diagnosis* instance can have several *diagnosis* instances, but one *diagnosis* instance must only have one *visit_diagnosis* instance.
- The attribute 'visit_id': a FK of *visit_procedure* is the PK of *visit*. (The PK of *visit_procedure* is a composite key consisting of 'visit_id' and 'procedure_id'.)
  - The relationship between *visit_procedure* and *visit* is a M:1 mandatory-to-mandatory relationship where one *visit_procedure* instance can have several *visit* instances, but one *visit* instance must only have one *visit_procedure* instance.
- The attribute 'procedure_id': a FK of *visit_procedure* is the PK of *clinical_procedures*.
  - The relationship between *visit_procedure* and *clinical_procedures* is a M:1 mandatory-to-mandatory relationship where one *visit_procedure* can have several *clinical_procedures* instances, but one *clinical_procedures* instance must have only one *visit_procedure*.

- The attribute ‘visit_id’: a FK of *visit_lab* is the PK of *visit*. (The PK of *visit_lab* is a composite key of ‘visit_id’ and ‘lab_id’.)
  - The relationship between *visit_lab* and *visit* is a M:1 mandatory-to-mandatory relationship where one *visit_lab* row can have many *visit*s, but one *visit* must have just one *visit_lab* row.
- The attribute ‘lab_id’: a FK of *visit_lab* is the PK of *lab*.
  - The relationship between *visit_lab* and *lab* is a M:1 mandatory-to-mandatory relationship where one *visit_lab* instance can have several *lab* instances, but one *lab* instance must only have one *visit_lab* instance.
- The attribute ‘visit_id’: a FK of *visit_symptom* is the PK of *visit*. (The PK of *visit_symptom* is a composite key consisting of ‘visit_id’ and ‘symptom_id’.)
  - The relationship between *visit_symptom* and *visit* is a M:1 mandatory-to-mandatory relationship where one *visit_symptom* instance can have several *visit* instances, but one *visit* instance must only have one *visit_symptom* instance.
- The attribute ‘symptom_id’: a FK of *visit_symptom* is the PK of *symptom*.
  - The relationship between *visit_symptom* and *symptom* is a M:1 mandatory-to-mandatory relationship where one *visit_symptom* instance can have several *symptom* instances, but one *symptom* instance must only have one *visit_symptom* instance.
- The attribute ‘patient_id’: a (potentially NULL) FK of *visit* is the PK of *patient*.
  - The relationship between *visit* and *patient* is a M:1 mandatory-to-mandatory relationship where one *visit* instance can have several *patient* instances, but one *patient* instance must only have one *visit* instance.
- The attribute ‘provider_id’: a (potentially NULL) FK of *visit* is the PK of *provider*.
  - The relationship between *visit* and *provider* is a M:1 mandatory-to-mandatory relationship where one *visit* instance can have several *provider* instances, but one *provider* instance must only have one *visit* instance.

Part 2: Design MongoDB Schema

Embedding versus Referencing Summary

Collection	Design	Justification
visit	Base document	Main entity; a separate collection
visit_diagnosis	Embedded *diagnosis*	Useful for browsing diagnoses per visit
diagnosis	Embedded in *visit_diagnosis*	Always accessed with *visit_diagnosis* info
visit_procedure	Embedded *clinical_procedures*	Useful for browsing procedures per visit
clinical_procedures	Embedded in *visit_procedure*	Always accessed with *visit_procedure* info
visit_lab	Embedded *lab*	Useful for browsing labs per visit

Collection	Design	Justification
lab	Embedded in *visit_lab*	Always accessed with *visit_lab* info
visit_symptom	Embedded *symptom*	Useful for browsing symptoms per visit
symptom	Embedded in *visit_symptom*	Always accessed with *visit_symptom* info
patient	Embedded in *visit*	Always accessed with *visit* info
provider	Referenced	Referenced in multiple *visit* instances

Specific Design Decisions (The collections for loading into MongoDB are in ***bold italic*** below.)

1. Embedding *diagnosis* in *visit_diagnosis*:
   - **Rationale**: Diagnoses are tightly coupled to a visit (in this case, 'visit_id') and unlikely to be queried independently due to *visit_diagnosis* containing just _id attributes. Embedding would simplify access and improve read performance for diagnosis names and ICD-10 codes. For example, having the 'diagnosis_id' attribute embedded in the same table as the 'name' attribute from *diagnosis* would grant the new ***visit_diagnosis*** collection simplified access and useful browsing capabilities for attributes that are frequently accessed together, putting a cancer diagnosis in the same collection as its listed procedure ID. The original *visit_diagnosis* is not referenced because it only contains IDs and no other useful information.
2. Embedding *clinical_procedures* in *visit_procedure*:
   - **Rationale**: Clinical procedures are tightly coupled to a visit (as per 'visit_id') and unlikely to be queried independently since *visit_procedure* just has _id attributes. Embedding would improve read performance for procedures' ICD-10 codes, names, and descriptions, while simplifying access. As an example, having the 'procedure_id' attribute embedded in the same table as the 'proc_name' attribute from *clinical_procedures* would simplify access to information about the procedures and enhance browsing capabilities per visit, putting together - within the new ***visit_procedure*** collection - details about an x-ray procedure with its appropriate identification ("ID") marker. The original *visit_procedure* is not referenced because it contains solely IDs and no useful information otherwise.
3. Embedding *lab* in *visit_lab*:
   - **Rationale**: Labs are tightly coupled to a visit (as per 'visit_id') and unlikely to be queried independently due to *visit_lab* containing only _id attributes. Embedding would simplify access and improve read performance for CPT codes and lab names. As an example, having the 'lab_id' attribute embedded in the same table as the 'lab_name' attribute from *lab* would give the new ***visit_lab*** collection simpler access and more useful browsing capabilities for attributes which are frequently accessed together, placing the cholesterol

lab test's CPT code with its corresponding ID value. The original *visit_lab* is not referenced because it only contains IDs and no other useful information.

4. Embedding *symptom* in *visit_symptom*:
   - **Rationale**: Symptoms are tightly coupled to a visit (in this case, 'visit_id') and unlikely to be queried independently since *visit_symptom* just contains _id attributes. Embedding would simplify access and improve read performance for symptom notes. For instance, having the 'visit_id' attribute embedded in the same table as the 'symptom_id' and 'note' attributes from *symptom* would simplify access to information about the symptoms and enhance browsing capabilities per visit, putting together - within the new ***visit_symptom*** collection - details about a coughing symptom with its appropriate ID value. The original *visit_symptom* is not referenced because it only contains IDs and no useful information otherwise.
5. Embedding *patient* in *visit*:
   - **Rationale**: Patients are tightly coupled to a visit (here, 'visit_id') and unlikely to be queried independently. Embedding simplifies access and improves read performance for patient first and last names as well as their genders. For example, having the 'patient_id' attribute embedded in the same table as the 'dob' attribute from *patient* would grant the new ***visit*** collection simpler access for attributes that are frequently accessed together, putting a patient's birthday in the same collection as that patient's ID.
6. Referencing *provider*:
   - **Rationale**: Providers appear in multiple visit records and unique provider records. Embedding would cause excessive data duplication and update anomalies, and keeping this collection referenced would allow for flexibility. As an example, the same HMO provider could be listed in multiple hospital visits in a given day, and the same HMO provider could appear several times in *visit* but appear only once uniquely (as per the ***provider*** collection).

JSON-style schema definitions

-- The ***visit*** collection (*patient* is embedded) --

```
{
        "$schema": "Base document that embeds the original patient table",
        "title": "visit",
        "description": "Each record represents one patient visit on a given day",
        "type": "object",
        "properties": {
                "visit_id": {
                        "description": "The unique identifier for a visit",
                        "type": "integer"
                },
                "patient_id": {
                        "description": "The unique identifier for a patient",
```

```
                    “type”: “integer”
                },
                “provider_id”: {
                    “description”: “The unique identifier for a provider”,
                    “type”: “integer”
                },
                “visit_date”: {
                    “description”: “Date of the visit”,
                    “type”: “date”
                },
                “patient_first_name”: {
                    “description”: “First name of the patient”,
                    “type”: “string”
                },
                “patient_last_name”: {
                    “description”: “Last name of the patient”,
                    “type”: “string”
                },
                “patient_dob”: {
                    “description”: “Date of birth of the patient”,
                    “type”: “date”
                },
                “patient_gender”: {
                    “description”: “Gender of the patient”,
                    “type”: “string”
                },
                “provider_first_name”: {
                    “description”: “First name of the provider”,
                    “type”: “string”
                },
                “provider_last_name”: {
                    “description”: “Last name of the provider”,
                    “type”: “string”
                }
        },
        “required”: [“visit_id”, “patient_id”, “provider_id”]
}
```

-- The ***provider*** collection --

```
{
        “$schema”: “Referenced document that remains as such for flexibility”,
        “title”: “visit”,
        “description”: “Each record represents one provider”,
```

```
        “type”: “object”,
        “properties”: {
                “provider_id”: {
                        “description”: “The unique identifier for a provider”,
                        “type”: “integer”
                },
                “provider_first_name”: {
                        “description”: “First name of the provider”,
                        “type”: “string”
                },
                “provider_last_name”: {
                        “description”: “Last name of the provider”,
                        “type”: “string”
                },
                “specialty”: {
                        “description”: “Specialty of the provider”,
                        “type”: “string”
                }
        },
        “required”: [“provider_id”]
}
```

-- The ***visit_diagnosis*** collection (*diagnosis* is embedded) --

```
{
        “$schema”: “Document that embeds the original diagnosis table”,
        “title”: “visit_diagnosis”,
        “description”: “Each record represents one diagnosis given during a visit on a given day”,
        “type”: “object”,
        “properties”: {
                “visit_id”: {
                        “description”: “The unique identifier for a visit”,
                        “type”: “integer”
                },
                “diagnosis_id”: {
                        “description”: “The unique identifier for a diagnosis”,
                        “type”: “integer”
                },
                “name”: {
                        “description”: “Name of the diagnosis”,
                        “type”: “string”
                },
                “icd10_code”: {
                        “description”: “ICD-10 code of the diagnosis”,
```

```
                    “type”: “string”
                }
        },
        “required”: [“visit_id”, “diagnosis_id”, “name”, “icd10_code”]
}
```

-- The ***visit_procedure*** collection (*clinical_procedures* is embedded) --

```
{
        “$schema”: “Document that embeds the original clinical_procedures table”,
        “title”: “visit_procedure”,
        “description”: “Each record represents one procedure ordered in a visit on a given day”,
        “type”: “object”,
        “properties”: {
                “visit_id”: {
                        “description”: “The unique identifier for a visit”,
                        “type”: “integer”
                },
                “procedure_id”: {
                        “description”: “The unique identifier for a procedure”,
                        “type”: “integer”
                },
                “icd10_code”: {
                        “description”: “ICD-10 code of the procedure”,
                        “type”: “string”
                },
                “proc_name”: {
                        “description”: “Name of the procedure”,
                        “type”: “string”
                },
                “proc_description”: {
                        “description”: “Description of the procedure”,
                        “type”: “string”
                }
        },
        “required”: [“visit_id”, “procedure_id”, “icd10_code”, “proc_name”, “proc_description”]
}
```

-- The ***visit_lab*** collection (*lab* is embedded) --

```
{
        “$schema”: “Document that embeds the original lab table”,
        “title”: “visit_lab”,
        “description”: “Each record represents one lab ordered during a visit on a given day”,
        “type”: “object”,
```

```
	“properties”: {
		“visit_id”: {
			“description”: “The unique identifier for a visit”,
			“type”: “integer”
		},
		“lab_id”: {
			“description”: “The unique identifier for a lab”,
			“type”: “integer”
		},
		“cpt_code”: {
			“description”: “CPT code of the lab”,
			“type”: “string”
		},
		“lab_name”: {
			“description”: “Name of the lab”,
			“type”: “string”
		}
	},
	“required”: [“visit_id”, “lab_id”, “cpt_code”, “lab_name”]
}
```

-- The ***visit_symptom*** collection (*symptom* is embedded) --

```
{
	“$schema”: “Document that embeds the original symptom table”,
	“title”: “visit_symptom”,
	“description”: “Each record represents one symptom identified at a visit on a given day”,
	“type”: “object”,
	“properties”: {
		“visit_id”: {
			“description”: “The unique identifier for a visit”,
			“type”: “integer”
		},
		“symptom_id”: {
			“description”: “The unique identifier for a symptom”,
			“type”: “integer”
		},
		“note”: {
			“description”: “Note of the symptom”,
			“type”: “string”
		}
	},
	“required”: [“visit_id”, “symptom_id”, “note”]
}
```

Part 3: Develop ETL Script

Creating the sample output for 3 patients and their complete visit history

Query for the 'ets103.***visit***.json' portion of 'sample_output.json':
{"$or":[{"patient_id":2},{"patient_id":3},{"patient_id":6}]}

Query for the 'ets103.***provider***.json' portion of 'sample_output.json' ("provider_id" values are chosen according to the chosen "patient_id" values above):
{"$or":[{"provider_id":36},{"provider_id":39},{"provider_id":50}]}

Query for the 'ets103.***visit_diagnosis***.json', 'ets103.***visit_procedure***.json', 'ets103.***visit_lab***.json', and 'ets103.***visit_symptom***.json' portions of 'sample_output.json' ("visit_id" values are chosen according to the chosen "patient_id" values above):
{"$or":[{"visit_id":10},{"visit_id":11},{"visit_id":12},{"visit_id":13},{"visit_id":14},
{"visit_id":15},{"visit_id":16},{"visit_id":17},{"visit_id":18},{"visit_id":19},{"visit_id":20},
{"visit_id":21},{"visit_id":22},{"visit_id":23},{"visit_id":34},{"visit_id":35},{"visit_id":36},
{"visit_id":37}]}

Ultimately, 'sample_output.json' is constructed as a combination of 'ets103.***visit***.json', 'ets103.***provider***.json', 'ets103.***visit_diagnosis***.json', 'ets103.***visit_procedure***.json', 'ets103.***visit_lab***.json', and 'ets103.***visit_symptom***.json' to provide the complete visit history of patients with patient IDs #2, #3, and #6. The individual JSON files and the combined 'sample_output.json' file are all included in the assignment submission.