**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп’ютерних наук та кібернетики

**Звіт**

до лабораторної роботи № 2

з дисципліни

**«**Проєктування та супровід баз даних та знань**»**

Виконав студент 1-го курсу

ОНП “Штучний інтелект”

Ходаков Максим Олегович

Київ – 2025

Опис предметної області:

**Назва: Банки, біржі, холдинги та валюти**

**Основні сутності**

**Банк** — емітент ETF та джерело рейтингових списків «Top Funds».

**ETF (Фонд)** — інвестиційний продукт, що наявний на біржах, виплачує розподіли та має портфель позицій.

**Журнал фактів (FactSheet)** — офіційний опис фонду (документ/URL) з датами; прив’язаний до конкретного ETF.

**Виплата/Розподіл (Distribution)** — окрема виплатна подія фонду інвесторам.

**Цінний папір (Security у загальному сенсі)** — надсутність для всіх інструментів, які можуть входити до портфеля або індексу.

**Акція (EquitySecurity)** — підтип «Цінного паперу».

**Облігація (BondSecurity)** — підтип «Цінного паперу».

**Холдинг (Holding)** — позиція фонду у певному цінному папері на момент часу. Має **self-зв’язок** «складається з».

**Індекс** — бенчмарк для фонду, що складається з багатьох складових слабких сутностей.

**Склад індексу (IndexConstituent)** — входження цінного паперу до індексу на дату; слабка сутність.

**Рейтинговий список від Банку (TopList)** — публікація «Top Funds».

**Елемент рейтингу (TopListItem)** — включення конкретного фонду до конкретного списку з позицією/категорією; слабка сутність.

**Біржа** — торговельний майданчик.

**Валюта** — валюта лістингу/котирування.

**Лістинг** — **тернарний** зв’язок (ETF × Біржа × Валюта) з атрибутами (ticker, is\_primary).

**Реалізація схеми MongoDB на основі MsSQL-структури**

# 1. Загальна характеристика проєкту

Мета: відтворити логіку і структуру SQL-схеми для системи аналітики ETF у середовищі MongoDB. Основний принцип – зберегти семантику сутностей, зв’язків і бізнес-правил, використовуючи JSON Schema, унікальні індекси та компаунд-ключі.

# 2. Відповідність сутностей SQL → MongoDB

|  |  |  |
| --- | --- | --- |
| SQL таблиця | MongoDB колекція | Примітки |
| etl.\_enum\_etf\_status, \_enum\_dist\_type, \_enum\_dist\_freq, \_enum\_sec\_type | \_enum\_etf\_status, \_enum\_dist\_type, \_enum\_dist\_freq, \_enum\_sec\_type | Довідники ENUM із унікальними значеннями |
| etl.bank | bank | Унікальний SWIFT, обов’язкові поля name/address |
| etl.exchange | exchange | Унікальний MIC, поле country |
| etl.currency | currency | Унікальний код валют (3 символи) |
| etl.etf | etf | Основна сутність фонду з полем статусу (enum) |
| etl.distribution | distribution | Перевірка послідовності дат через $expr |
| etl.security, equity\_security, bond\_security | security | Об’єднані у єдину колекцію з дискримінатором secType |
| etl.holding | holding | Self-зв’язок через parentHoldingId |
| etl.idx | idx | Довідник індексів |
| etl.index\_constituent | index\_constituent | Компаунд PK через унікальний індекс |
| etl.toplist, etl.toplist\_item | toplist, toplist\_item | Публікації банку з рейтингами фондів |
| etl.listing | listing | Тернарна сутність ETF × Exchange × Currency |

# 3. Обмеження цілісності та валідації

- Первинні ключі: `\_id: ObjectId`; компаундні через unique індекси.  
- Зовнішні ключі: відсутні, зв’язки реалізовано через ObjectId.  
- CHECK: реалізовано через JSON Schema або `$expr`.  
- Унікальність: забезпечено індексами (`unique: true`, `sparse: true`).

# 4. Індексація

Основні індекси:  
- bank.swift, exchange.mic, currency.code — унікальні;  
- security.isin/cusip/sedol — sparse unique;  
- listing(fundId, exchangeId, currencyId) — компаунд unique;  
- listing(exchangeId, ticker) — unique;  
- listing(fundId) — partial unique (isPrimary=true).

# 5. Відмінності від реляційної моделі

1. Відсутні тригери — логіка перевірки ієрархії переноситься на рівень сервісу.  
2. Немає зовнішніх ключів — контролюються додатком.  
3. Об’єднання підтипів Security у єдину колекцію.  
4. Додано поле `holding.path` для швидкого пошуку по дереву.  
5. Використано partial індекси для бізнес-правил.

# 6. Демонстраційні дані

Скрипт містить вставки прикладів даних для тестування зв’язків:  
- Bank, Exchange, Currency  
- ETF, Distribution, Listing  
- Security (equity/bond), Holding (parent/child)  
- Index, Index Constituent  
- Toplist, Toplist Item

# 7. Висновок

MongoDB-схема повністю відтворює структуру SQL-схеми за сутностями, полями, унікальністю та обмеженнями.

Репозиторій: <https://github.com/maksymkhodakov/DesignAndMaintenanceOfDatabaseAndKnowledgeLab1>

**Додаток**

use etl;  
  
*// Helpers  
function* existsCollection(*name*) {  
 return db.getCollectionInfos({ *name* }).length > 0;  
}  
*function* dropIfExists(*name*) {  
 if (existsCollection(*name*)) db.getCollection(*name*).drop();  
}  
  
*// ============================ ENUMS ================================*db.getCollectionInfos({ name: /^\_enum\_/ }).forEach(*ci* => db.getCollection(*ci*.name).drop());  
  
db.createCollection("\_enum\_etf\_status");  
db.getCollection("\_enum\_etf\_status").createIndex({ val: 1 }, { unique: true, name: "UQ\_etf\_status" });  
db.getCollection("\_enum\_etf\_status").insertMany([  
 { val: "active" }, { val: "inactive" }, { val: "liquidated" }, { val: "suspended" }  
]);  
  
db.createCollection("\_enum\_dist\_type");  
db.getCollection("\_enum\_dist\_type").createIndex({ val: 1 }, { unique: true, name: "UQ\_dist\_type" });  
db.getCollection("\_enum\_dist\_type").insertMany([  
 { val: "dividend" }, { val: "capital\_gains" }, { val: "return\_of\_capital" }, { val: "interest" }  
]);  
  
db.createCollection("\_enum\_dist\_freq");  
db.getCollection("\_enum\_dist\_freq").createIndex({ val: 1 }, { unique: true, name: "UQ\_dist\_freq" });  
db.getCollection("\_enum\_dist\_freq").insertMany([  
 { val: "monthly" }, { val: "quarterly" }, { val: "semiannual" }, { val: "annual" }, { val: "irregular" }  
]);  
  
db.createCollection("\_enum\_sec\_type");  
db.getCollection("\_enum\_sec\_type").createIndex({ val: 1 }, { unique: true, name: "UQ\_sec\_type" });  
db.getCollection("\_enum\_sec\_type").insertMany([{ val: "equity" }, { val: "bond" }]);  
  
*// ============================ BANK ================================*dropIfExists("bank");  
db.createCollection("bank", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["name","swift","address"],  
 properties: {  
 name: { bsonType: "string", maxLength: 200 },  
 swift: { bsonType: "string", maxLength: 20 },  
 address:{ bsonType: "string", maxLength: 500 }  
 } } }});  
db.getCollection("bank").createIndex({ swift: 1 }, { unique: true, name: "UQ\_bank\_swift" });  
  
*// ============================ EXCHANGE ============================*dropIfExists("exchange");  
db.createCollection("exchange", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["name","mic","country"],  
 properties: {  
 name: { bsonType: "string", maxLength: 200 },  
 mic: { bsonType: "string", maxLength: 20 },  
 country: { bsonType: "string", maxLength: 100 }  
 } } }});  
db.getCollection("exchange").createIndex({ mic: 1 }, { unique: true, name: "UQ\_exchange\_mic" });  
  
*// ============================ CURRENCY ============================*dropIfExists("currency");  
db.createCollection("currency", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["code","name"],  
 properties: {  
 code: { bsonType: "string", minLength: 3, maxLength: 3 },  
 name: { bsonType: "string", maxLength: 100 }  
 } } }});  
db.getCollection("currency").createIndex({ code: 1 }, { unique: true, name: "UQ\_currency\_code" });  
  
*// ============================ ETF ================================*dropIfExists("etf");  
db.createCollection("etf", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["bankId","topFundName","effectiveDate","status"],  
 properties: {  
 *bankId*: { bsonType: "objectId" },  
 topFundName: { bsonType: "string", maxLength: 250 },  
 topFundTicker:{ bsonType: ["string","null"], maxLength: 64 },  
 effectiveDate: { bsonType: "date" },  
 status: { enum: ["active","inactive","liquidated","suspended"] },  
 nav: { bsonType: ["double","decimal","null"] },  
 mer: { bsonType: ["double","decimal","null"] },  
 factUrl: { bsonType: ["string","null"], maxLength: 1000 }  
 } } }});  
db.getCollection("etf").createIndex({ *bankId*: 1 }, { name: "IX\_etf\_bank\_id" });  
  
*// ============================ DISTRIBUTION =========================*dropIfExists("distribution");  
db.createCollection("distribution", {  
 validator: { $and: [  
 { $jsonSchema: {  
 bsonType: "object",  
 required: ["etfId","recordDate","exDate","payDate","amountPerShare","distType","distFreq"],  
 properties: {  
 *etfId*: { bsonType: "objectId" },  
 recordDate: { bsonType: "date" },  
 exDate: { bsonType: "date" },  
 payDate: { bsonType: "date" },  
 amountPerShare: { bsonType: ["double","decimal"], minimum: 0 },  
 distType: { enum: ["dividend","capital\_gains","return\_of\_capital","interest"] },  
 distFreq: { enum: ["monthly","quarterly","semiannual","annual","irregular"] }  
 } } },  
 { $expr: { $and: [ { $lte: ["$recordDate","$exDate"] }, { $lte: ["$exDate","$payDate"] } ] } }  
 ] }});  
db.getCollection("distribution").createIndex({ *etfId*: 1 }, { name: "IX\_distribution\_etf\_id" });  
  
*// ============================ SECURITY ============================*dropIfExists("security");  
db.createCollection("security", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["name","secType"],  
 properties: {  
 isin: { bsonType: ["string","null"], maxLength: 20 },  
 cusip: { bsonType: ["string","null"], maxLength: 20 },  
 sedol: { bsonType: ["string","null"], maxLength: 20 },  
 topFmcId: { bsonType: ["string","null"], maxLength: 64 },  
 name: { bsonType: "string", maxLength: 250 },  
 secType: { enum: ["equity","bond"] },  
 equityName: { bsonType: ["string","null"], maxLength: 40 },  
 coupon: { bsonType: ["double","decimal","null"] },  
 maturityDate: { bsonType: ["date","null"] },  
 rating: { bsonType: ["string","null"], maxLength: 40 }  
 },  
 oneOf: [ { properties: { secType: { enum: ["equity"] } } },  
 { properties: { secType: { enum: ["bond"] } } } ]  
 } }});  
db.getCollection("security").createIndex({ isin: 1 }, { unique: true, sparse: true, name: "UQ\_security\_isin" });  
db.getCollection("security").createIndex({ cusip: 1 }, { unique: true, sparse: true, name: "UQ\_security\_cusip" });  
db.getCollection("security").createIndex({ sedol: 1 }, { unique: true, sparse: true, name: "UQ\_security\_sedol" });  
  
*// ============================ HOLDING =============================*dropIfExists("holding");  
db.createCollection("holding", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["fundId","securityId","marketValueLocal","marketValueBase","units"],  
 properties: {  
 fundId: { bsonType: "objectId" },  
 securityId: { bsonType: "objectId" },  
 marketValueLocal: { bsonType: ["double","decimal"], minimum: 0 },  
 marketValueBase: { bsonType: ["double","decimal"], minimum: 0 },  
 units: { bsonType: ["double","decimal"], minimum: 0 },  
 parentHoldingId: { bsonType: ["objectId","null"] },  
 path: { bsonType: ["array","null"], items: { bsonType: "objectId" } }  
 } } }});  
db.getCollection("holding").createIndex({ fundId: 1 }, { name: "IX\_holding\_fund\_id" });  
db.getCollection("holding").createIndex({ securityId: 1 }, { name: "IX\_holding\_security\_id" });  
db.getCollection("holding").createIndex({ parentHoldingId: 1 }, { name: "IX\_holding\_parent\_id" });  
db.getCollection("holding").createIndex({ fundId: 1, path: 1 }, { name: "IX\_holding\_path" });  
  
*// ============================ IDX ================================*dropIfExists("idx");  
db.createCollection("idx", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["name","provider"],  
 properties: {  
 name: { bsonType: "string", maxLength: 200 },  
 provider: { bsonType: "string", maxLength: 200 }  
 } } }});  
  
*// ===================== INDEX\_CONSTITUENT ==========================*dropIfExists("index\_constituent");  
db.createCollection("index\_constituent", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["indexId","securityId","asOfDate","targetWeight"],  
 properties: {  
 *indexId*: { bsonType: "objectId" },  
 securityId: { bsonType: "objectId" },  
 asOfDate: { bsonType: "date" },  
 targetWeight: { bsonType: ["double","decimal"], minimum: 0, maximum: 1 }  
 } } }});  
db.getCollection("index\_constituent").createIndex(  
 { *indexId*: 1, securityId: 1, asOfDate: 1 },  
 { unique: true, name: "PK\_index\_constituent" }  
);  
db.getCollection("index\_constituent").createIndex({ *indexId*: 1, asOfDate: 1 }, { name: "IX\_index\_const\_idx\_date" });  
  
*// ============================ TOPLIST =============================*dropIfExists("toplist");  
db.createCollection("toplist", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["bankId","name","publishedAt"],  
 properties: {  
 *bankId*: { bsonType: "objectId" },  
 name: { bsonType: "string", maxLength: 250 },  
 publishedAt: { bsonType: "date" },  
 notes: { bsonType: ["string","null"], maxLength: 1000 }  
 } } }});  
db.getCollection("toplist").createIndex({ *bankId*: 1 }, { name: "IX\_toplist\_bank\_id" });  
  
*// ========================= TOPLIST\_ITEM ===========================*dropIfExists("toplist\_item");  
db.createCollection("toplist\_item", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["toplistId","rank"],  
 properties: {  
 *toplistId*: { bsonType: "objectId" },  
 rank: { bsonType: "int", minimum: 1 },  
 category: { bsonType: ["string","null"], maxLength: 100 }  
 } } }});  
db.getCollection("toplist\_item").createIndex({ *toplistId*: 1 }, { name: "IX\_toplist\_item\_toplist" });  
  
*// ============================ LISTING =============================*dropIfExists("listing");  
db.createCollection("listing", {  
 validator: { $jsonSchema: {  
 bsonType: "object",  
 required: ["fundId","exchangeId","currencyId","ticker","isPrimary"],  
 properties: {  
 fundId: { bsonType: "objectId" },  
 *exchangeId*: { bsonType: "objectId" },  
 *currencyId*: { bsonType: "objectId" },  
 ticker: { bsonType: "string", maxLength: 64 },  
 isPrimary: { bsonType: "bool" }  
 } } }});  
*// PK (fund,exchange,currency)*db.getCollection("listing").createIndex(  
 { fundId: 1, *exchangeId*: 1, *currencyId*: 1 },  
 { unique: true, name: "PK\_listing" }  
);  
*// Unique (exchange, ticker)*db.getCollection("listing").createIndex(  
 { *exchangeId*: 1, ticker: 1 },  
 { unique: true, name: "UQ\_listing\_exchange\_ticker" }  
);  
*// Exactly one primary per fund (partial unique)*db.getCollection("listing").createIndex(  
 { fundId: 1 },  
 { unique: true, partialFilterExpression: { isPrimary: true }, name: "IX\_listing\_one\_primary\_per\_fund" }  
);  
  
*// ============================ DEMO DATA ============================*const *bankId* = db.getCollection("bank").insertOne({ name: "Cdn Bank", swift: "CDNBBCA1", address: "Toronto, ON" }).insertedId;  
const *exchangeId* = db.getCollection("exchange").insertOne({ name: "TSX", mic: "XTSE", country: "CA" }).insertedId;  
const *currencyId* = db.getCollection("currency").insertOne({ code: "CAD", name: "Canadian Dollar" }).insertedId;  
  
const *etfId* = db.getCollection("etf").insertOne({  
 *bankId*,  
 topFundName: "Cdn Total Market",  
 topFundTicker: null,  
 effectiveDate: ISODate("2020-01-01T00:00:00Z"),  
 status: "active",  
 nav: NumberDecimal("100.12"),  
 mer: NumberDecimal("0.0600"),  
 factUrl: "https://bank.example/factsheet/ctm"  
}).insertedId;  
  
db.getCollection("listing").insertOne({ fundId: *etfId*, *exchangeId*, *currencyId*, ticker: "CTM", isPrimary: true });  
  
db.getCollection("distribution").insertOne({  
 *etfId*,  
 recordDate: ISODate("2024-12-20T00:00:00Z"),  
 exDate: ISODate("2024-12-23T00:00:00Z"),  
 payDate: ISODate("2024-12-30T00:00:00Z"),  
 amountPerShare: NumberDecimal("0.245"),  
 distType: "dividend",  
 distFreq: "quarterly"  
});  
  
const *secEquityId* = db.getCollection("security").insertOne({  
 isin: "ISIN0001", cusip: "CUSIP001", sedol: "SEDOL01", topFmcId: "PRV1",  
 name: "ACME Corp", secType: "equity", equityName: null  
}).insertedId;  
  
const *secBondId* = db.getCollection("security").insertOne({  
 isin: "ISIN0002", cusip: "CUSIP002", sedol: "SEDOL02", topFmcId: "PRV2",  
 name: "Gov Bond 2030", secType: "bond",  
 coupon: NumberDecimal("0.035"), maturityDate: ISODate("2030-06-30T00:00:00Z"), rating: "AA"  
}).insertedId;  
  
const *holdingAId* = db.getCollection("holding").insertOne({  
 fundId: *etfId*, securityId: *secEquityId*,  
 marketValueLocal: NumberDecimal("100000"), marketValueBase: NumberDecimal("100000"),  
 units: NumberDecimal("1000"), parentHoldingId: null  
}).insertedId;  
  
const *holdingBId* = db.getCollection("holding").insertOne({  
 fundId: *etfId*, securityId: *secBondId*,  
 marketValueLocal: NumberDecimal("50000"), marketValueBase: NumberDecimal("50000"),  
 units: NumberDecimal("600"), parentHoldingId: null  
}).insertedId;  
  
const *indexId* = db.getCollection("idx").insertOne({ name: "S&P Canada Broad", provider: "S&P" }).insertedId;  
db.getCollection("index\_constituent").insertMany([  
 { *indexId*, securityId: *secEquityId*, asOfDate: ISODate("2025-01-01T00:00:00Z"), targetWeight: NumberDecimal("0.060000") },  
 { *indexId*, securityId: *secBondId*, asOfDate: ISODate("2025-01-01T00:00:00Z"), targetWeight: NumberDecimal("0.040000") }  
]);  
  
const *toplistId* = db.getCollection("toplist").insertOne({  
 *bankId*, name: "Top Funds — Jan 2025",  
 publishedAt: ISODate("2025-01-15T00:00:00Z"),  
 notes: "Income & Broad Market"  
}).insertedId;  
  
db.getCollection("toplist\_item").insertOne({ *toplistId*, rank: 1, category: "Broad Market" });  
  
print("=== MongoDB ETL schema created & demo data inserted ===");