Contents

[BUILD MONGODB DATABASE 2](#_Toc110034405)

[**1.** **PHÂN TÍCH TẬP DỮ LIỆU** 2](#_Toc110034406)

[**2.** **THIẾT KẾ LƯỢC ĐỒ DATABASE** 2](#_Toc110034407)

[**3.** **TẠO DATABASE TRONG MONGODB** 3](#_Toc110034408)

[**4.** **TRUY VẤN NGHIỆP VỤ** 6](#_Toc110034409)

[**5.** **TẠO INDEX** 7](#_Toc110034410)

BUILD MONGODB DATABASE

FROM FLAT FILE

1. **PHÂN TÍCH TẬP DỮ LIỆU**

Database sẽ gồm 5 collections: relationship, education, user, finance, occupation

+ relationship: gồm 2 trường relationship\_note (thông tin thêm), marital\_status (tình trạng hôn nhân)

+ education: gồm 2 trường education\_num (số cấp độ giáo dục) và education\_level (trình độ giáo dục cao nhất)

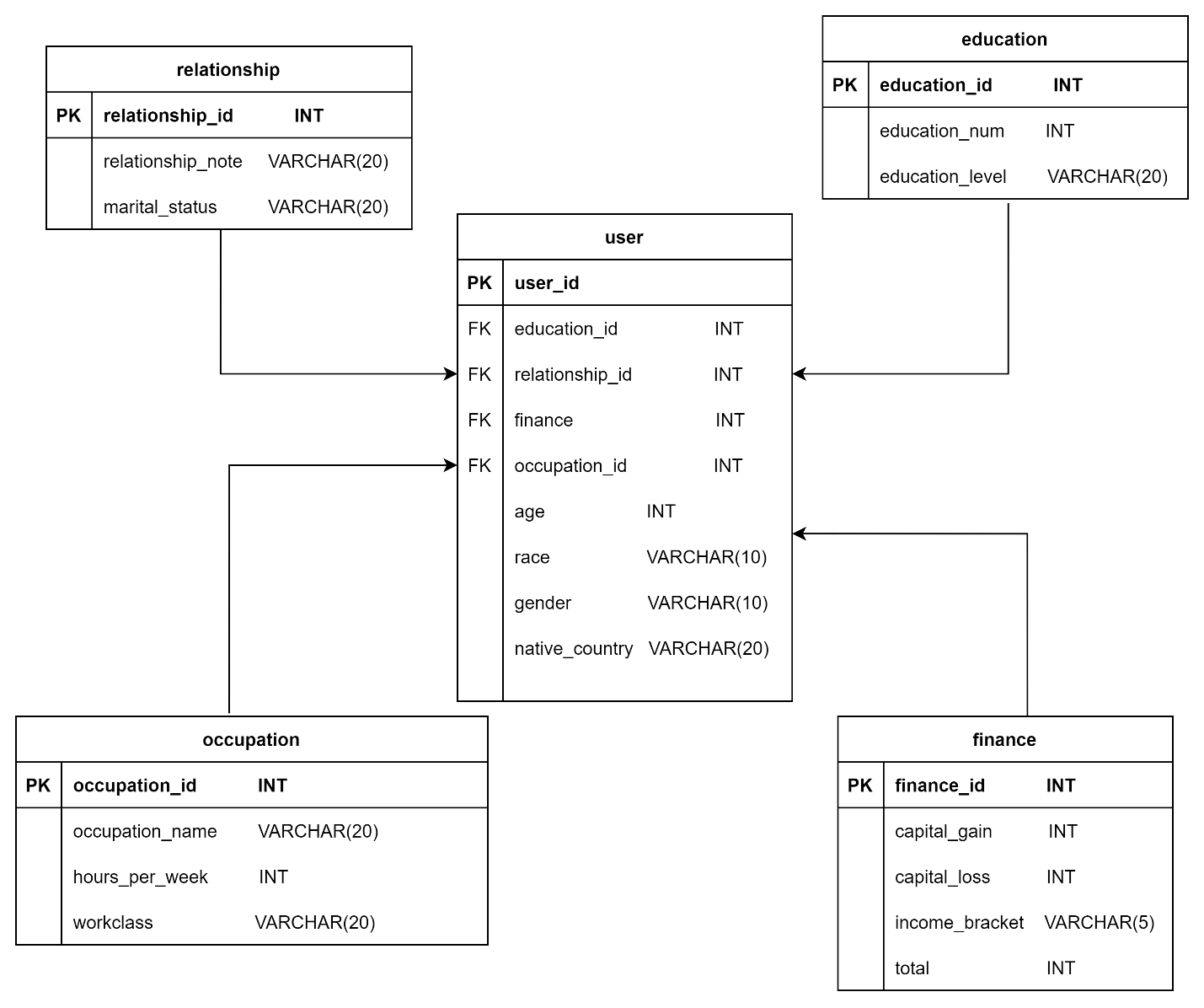
+ occupation: gồm 3 trường occupation\_name (tên nghề nghiệp), hours\_per\_week (số giờ làm việc mỗi tuần), và workclass (hình thức làm việc)

+ finance: gồm 4 trường capital\_gain, capital\_loss, income\_bracket (thu nhập hằng tháng), và total (tổng số tiền trong mọi tài khoản)

+ user: gồm 4 trường age, race, gender, native\_country

COLLECTION USER sẽ kết nối với 4 bảng khác thông qua mã \_id, nhận mã \_id của 4 bảng còn lại làm khóa ngoại

1. **THIẾT KẾ LƯỢC ĐỒ DATABASE**



1. **TẠO DATABASE TRONG MONGODB**
2. Open CMD (run as administrator)
3. Viết script: “**cd C:\Program Files\MongoDB\Server\5.0\bin\”**

để chuyển đến mongodb BIN (chứa mongoimport)

1. Copy file vào mục bin ở trên (để rút ngắn đường dẫn file, nhưng vẫn có thể lưu file ở nơi khác)
2. Tiếp tục viết câu lệnh sau vào cmd: **mongoimport -d asm2 -c persons\_asm2 --type csv --headerline --file "US\_Adult\_Income.csv"**

# EDUCATION

function loadEducation() {

    const bulkInsert = db.education.initializeUnorderedBulkOp();

    // Get all Documents in 'full' Collection

    const documents = db.persons\_asm2.find({});

    // Process each document

    documents.forEach(function (doc) {

        const element = {

            education\_level: doc.education\_level,

            education\_num: doc.education\_num

        };

        // Upsert into education Document

        bulkInsert.find(element).upsert().replaceOne(element);

    });

    bulkInsert.execute();

    return true;

}

# RELATIONSHIP

function loadRelationship() {

    const bulkInsert = db.relationship.initializeUnorderedBulkOp();

    const documents = db.persons\_asm2.find({});

    documents.forEach(function (doc) {

        const element = {

            relationship\_note: doc.relationship\_note,

            marital\_status: doc.marital\_status

        };

        bulkInsert.find(element).upsert().replaceOne(element);

    });

    bulkInsert.execute();

    return true;

}

# FINANCE

function loadFinance() {

    const bulkInsert = db.finance.initializeUnorderedBulkOp();

    const documents = db.persons\_asm2.find({});

    documents.forEach(function (doc) {

        const element = {

            capital\_gain: doc.capital\_gain,

            capital\_loss: doc.capital\_loss,

            income\_bracket: doc.income\_bracket,

            total: doc.total

        };

        bulkInsert.find(element).upsert().replaceOne(element);

    });

    bulkInsert.execute();

    return true;

}

# OCCUPATION

function loadOccupation() {

    const bulkInsert = db.occupation\_tbl.initializeUnorderedBulkOp();

    const documents = db.persons\_asm2.find({});

    documents.forEach(function (doc) {

        const element = {

            occupation\_name: doc.occupation,

            hours\_per\_week: doc.hours\_per\_week,

            workclass: doc.workclass

        };

        bulkInsert.find(element).upsert().replaceOne(element);

    });

    bulkInsert.execute();

    return true;

}

# USER

# KẾT HỢP IMPORT KHÓA NGOẠI CỦA 4 BẢNG

function loadUser() {

    const bulkInsert = db.user.initializeUnorderedBulkOp();

    const documents = db.persons\_asm2.find({});

    documents.forEach(function (doc) {

        const element = {

            age: doc.age,

            race: doc.race,

            gender: doc.gender,

            native\_country: doc.native\_country

        };

        const education = db.education.findOne({

            education: doc.education

        });

        const finance = db.finance.findOne({

            finance: doc.finance

        });

        const relationship = db.relationship.findOne({

            relationship: doc.relationship

        });

        const occupation = db.occupation\_tbl.findOne({

            occupation: doc.occupation\_tbl

        });

        element.education\_id = education.\_id;

        element.finance\_id = finance.\_id;

        element.relationship\_id = relationship.\_id;

        element.occupation\_id = occupation.\_id;

        bulkInsert.find(element).upsert().replaceOne(element);

    });

    bulkInsert.execute();

    return true;}



## **4. TRUY VẤN NGHIỆP VỤ**

1. Có bao nhiêu người là Nam và làm việc nhiều hơn 30 tiếng / tuần ?

db.persons\_asm2.find({

    'gender': 'Male', "hours\_per\_week" : { $gt: 30}

})

1. Có bao nhiêu người ở Mỹ có mức thu nhập > 50K

db.persons\_asm2.find({

    'native\_country' : 'United-States' , 'income\_bracket': '>50K'

 })

1. Tính tổng số dư tài khoản của những người đang ở Mỹ.

db.persons\_asm2.aggregate([

    { $match: { 'native\_country': 'United-States'}},

    { $group: { \_id: {country: '$native\_country'}, totalAmount: { $sum: '$total'}}},

])

1. Tính tổng số giờ làm việc một tuần của những người có mức thu nhập > 50K, và tổng số người tương ứng

db.persons\_asm2.aggregate([

    { $match: { "hours\_per\_week": { $gt: 50}}},

    { $group: { \_id: "person\_work\_>50hours", totalHour: { $sum: "$hours\_per\_week"}, numPerson: { $sum:1}}}])

1. Tìm những người có tổng số tiền trong tài khoản > 100000 và có số giờ làm việc hàng tuần < 55

db.persons\_asm2.find({

    "hours\_per\_week" : { $lt: 55}, "total": { $gt: 100000}

})

## **5. TẠO INDEX**

# INDEX

db.relationship.createIndex({relationship\_note: 1});

* Index theo relationship\_note (sẽ gồm các trường cố định), phần còn lại là “marital\_status” chỉ gồm 2 giá trị nên không cần gộp thành index

db.education.createIndex({education\_num: 1});

* Index theo mã số “education\_num” với nhiều giá trị

db.finance.createIndex({total: 1});

* Index theo tổng số tiền, nhiều khoảng giá trị nên có index sẽ tăng tốc độ tìm kiếm

db.occupation.createIndex({hours\_per\_week: 1});

* Index theo số giờ làm việc hằng tuần, nhiều giá trị và có ý nghĩa trong tìm kiếm nghiệp vụ thực tế

**KẾT QUẢ**

