Có 2 dạng Transaction :

* Unmanaged : tự tay quản lý
* Managed : tự động do sqlize quản lý

## Unmanaged transactions

const t = await sequelize.transaction();

try {

const user = await User.create({

firstName: 'Bart',

lastName: 'Simpson'

}, { transaction: t });

await user.addSibling({

firstName: 'Lisa',

lastName: 'Simpson'

}, { transaction: t });

await t.commit();

} catch (error) {

await t.rollback();

}

await User.create({ name: 'Foo Bar' }, { transaction: t });

await userUpdate.update({ name: 'Foo Barrrr' }, { transaction: t });

await User.findAll({

where: {

name: 'Foo Bar'

},

transaction: t

});

## Managed Transactions

Auto rollback bởi sqlize

try {

const result = await sequelize.transaction(async (t) => {

const user = await User.create({

firstName: 'Abraham',

lastName: 'Lincoln'

}, { transaction: t });

throw new Error(); // khi throw err thì sqlize tự rollback

await user.setShooter({

firstName: 'John',

lastName: 'Boothe'

}, { transaction: t });

return user;

});

} catch (error) {

}

## Isolation Level

const { Transaction } = require('sequelize');

// The following are valid isolation levels:

Transaction.ISOLATION\_LEVELS.READ\_UNCOMMITTED // "READ UNCOMMITTED"

Transaction.ISOLATION\_LEVELS.READ\_COMMITTED // "READ COMMITTED"

Transaction.ISOLATION\_LEVELS.REPEATABLE\_READ // "REPEATABLE READ"

Transaction.ISOLATION\_LEVELS.SERIALIZABLE // "SERIALIZABLE"

const { Transaction } = require('sequelize');

await sequelize.transaction({

isolationLevel: Transaction.ISOLATION\_LEVELS.SERIALIZABLE

}, async (t) => {

// Your code

});

## Locks

return User.findAll({

limit: 1,

lock: true,

transaction: t1

});

return User.findAll({

limit: 1,

lock: true,

skipLocked: true,

transaction: t2

});