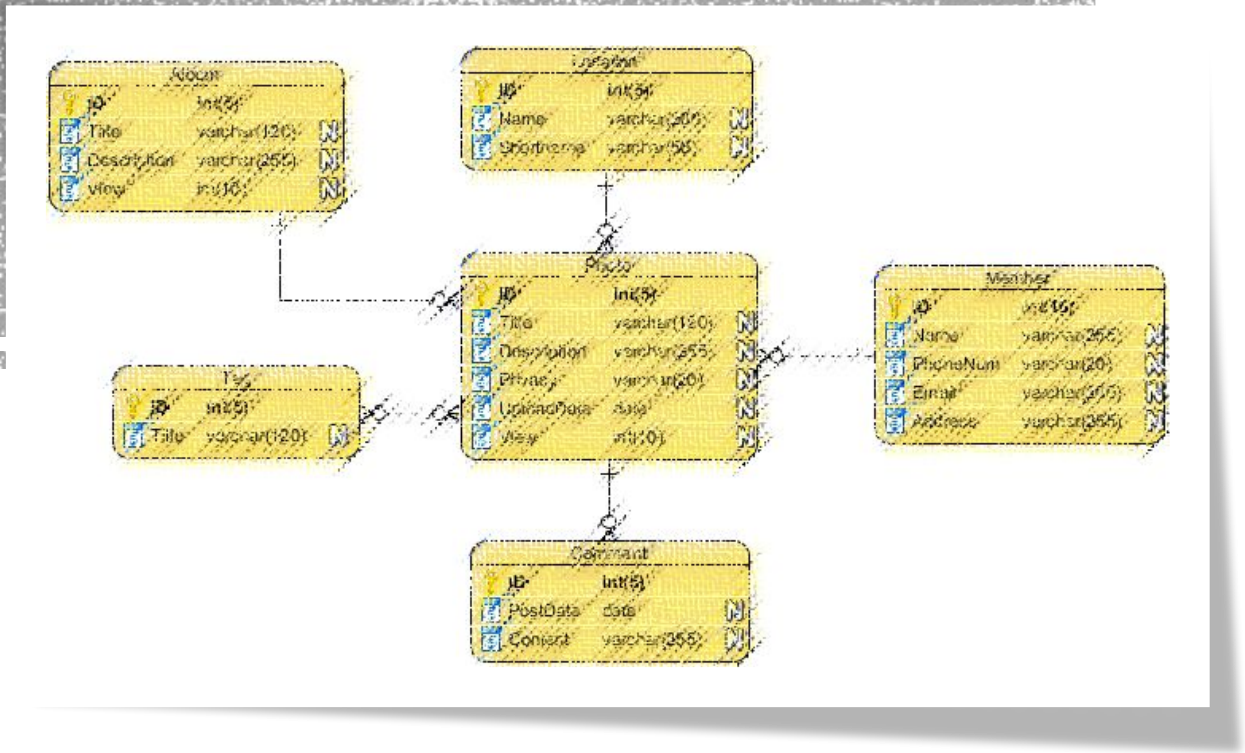
Database Denormalization

SWE 4601 Software Design and Architecture



Normalization & Denormalization

- **Normalization** is a technique of eliminating the redundant data from the database.
- **Denormalization** is the inverse process of normalization where the redundancy is added to the data to improve the performance of the specific application.
- By updating, inserting, and deleting records through foreign key relationships, normalization reduces instances of missed or orphaned records in your database.
- Number of tables   Increases in normalization

  Decreases in denormalization
- While normalized data is optimized for entity level transactions, denormalized data is optimized for answering business questions and driving decision making.

Pros of Normalization

- Complies to ACID property
  - Atomicity
  - Consistency
  - Isolation
  - Durability
- Updates run quickly
- Inserts run quickly
- Clear understanding of data
- No redundancy

So, why denormalization?

Normalized DB require join

A lot of join

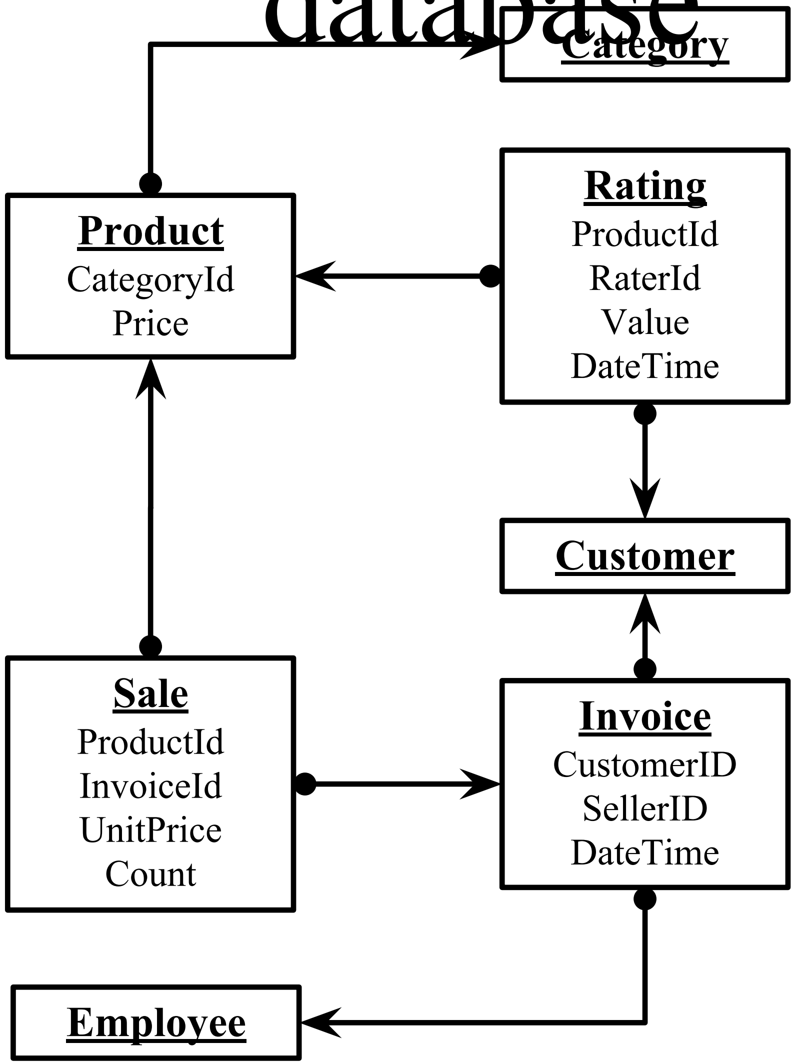
May be crazy lot of join

So What?

¯_(ツ)_/¯

Join is expensive

The kid's shop database



Category

Product
CategoryId
Price

Rating
ProductId
RaterId
Value
DateTime

Customer

Sale
ProductId
InvoiceId
UnitPrice
Count

Invoice
CustomerID
SellerID
DateTime

Employee

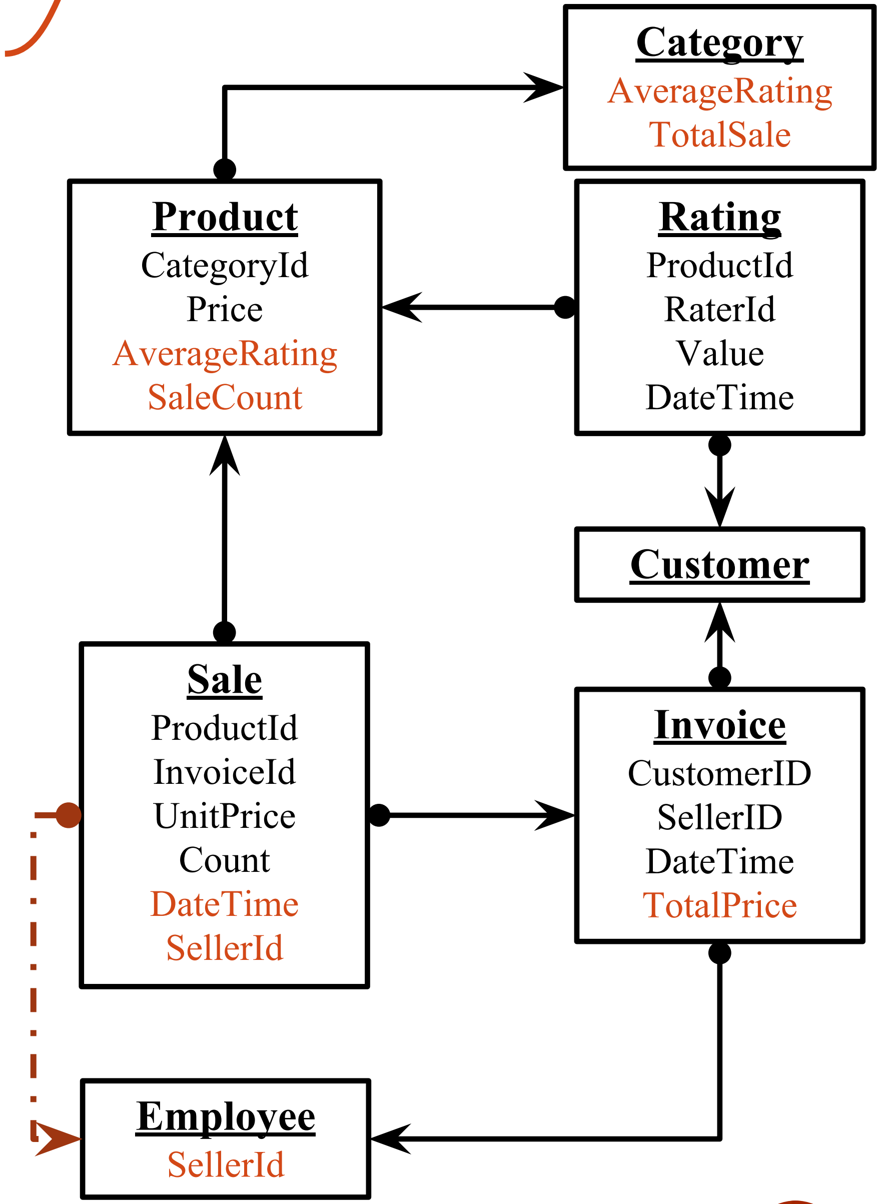
Which tables need to be joined for the following use cases?

1. A customer wants to see top rated products
2. A sales manager wants to find products with most number of sells
3. A sales manager wants to find products with most sale amount last month
4. A sales manager wants to award the top 3 salespersons of the last month
5. A marketing manager wants to award discount coupons to customers who purchased more than 10,000 Taka last month
6. A customer wants to see popular categories
7. Top management want to see a chart showing total sale of top 5 categories in a May 2020
8. A sales manager wants to see salespersons' sale per category

Speed up the queries by redundancy

of immutable data

1. A customer wants to see top rated products (Previously Product, Rating)
2. A sales manager wants to find products with most number of sales (Previously Product, Sale)
3. A sales manager wants to find products with most sale amount last month (Previously Product, Sale, Invoice)
4. A sales manager wants to award the top 3 sales persons of the last month (Previously Sale, Invoice, Employee)
5. A marketing manager wants to award discount coupons to customers who purchased more than 10000 Taka last month (Previously Customer, Invoice, Sale)
6. A customer wants to see popular categories (Previously Category, Product, Rating)
7. Top management want to see a chart showing total sale of top 5 categories in a May 2020 (Previously Sale, category, product, Invoice, rating)
8. A sales manager wants to see sales persons' sale per category (Previously Employee, Invoice, sale, Product, Category)



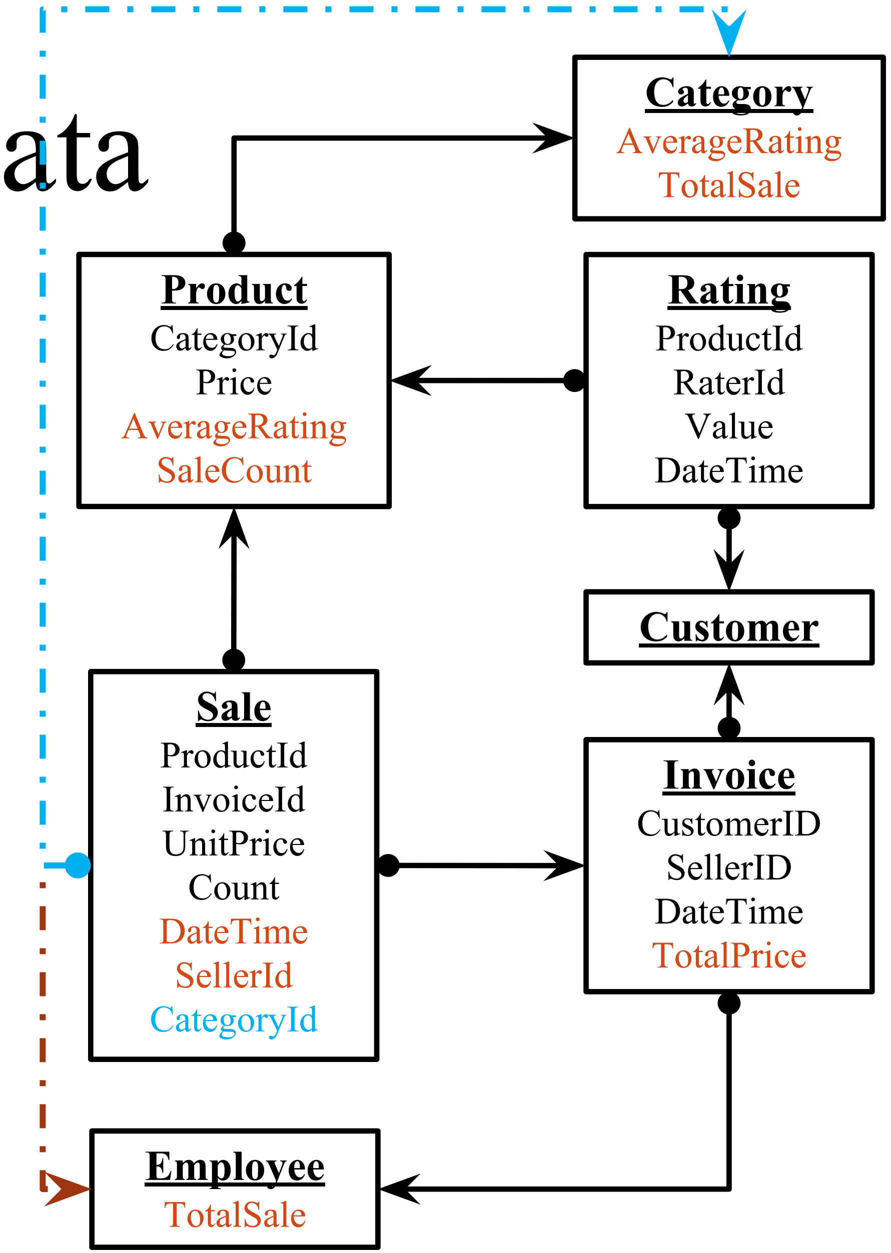
What if we keep a categoryId in Sale?

Redundancy of mutable data

- Wrong updates can be done from different parts of the code
  - Solution: Make sure update of one data is done by only one piece of code
- Updates can be slow
  - No big deal if update is not so frequent

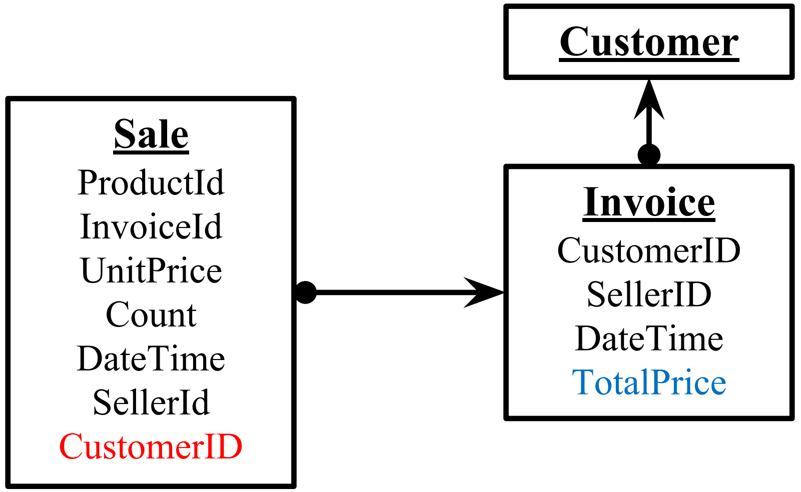
Lets solve now

7. Top management want to see a chart showing total sale of top 5 categories in a May 2020 (Previously Category, product, Sale)
8. A sales manager wants to see sales persons' sale per category (Previously Employee, Sale, Product, Category)



Choices need to be made

- A marketing manager wants to award discount coupons to customers who purchased more than 10,000 Taka last month (Customer, Invoice, Sale)



So, what did we gain by using redundancy?

- Faster query
- Easier query
- Meaningful query, examples
  - A customer wants to see top rated products
    - queries just product
  - A customer wants to see popular categories
    - queries just Category
  - Top management want to see a chart showing total sale of top 5 categories in a May 2020
    - queries Sale and Category
  - A sales manager wants to see sales persons’ sale per category
    - queries Employee, Sale, Category