Exam #1 Review

Zuyin (Alvin) Zheng

Data/Information/Database

Data vs. Information

Data

Discrete, unorganized, raw facts

Information

The transformation of those facts into meaning

Transactional Data vs. Analytical Data

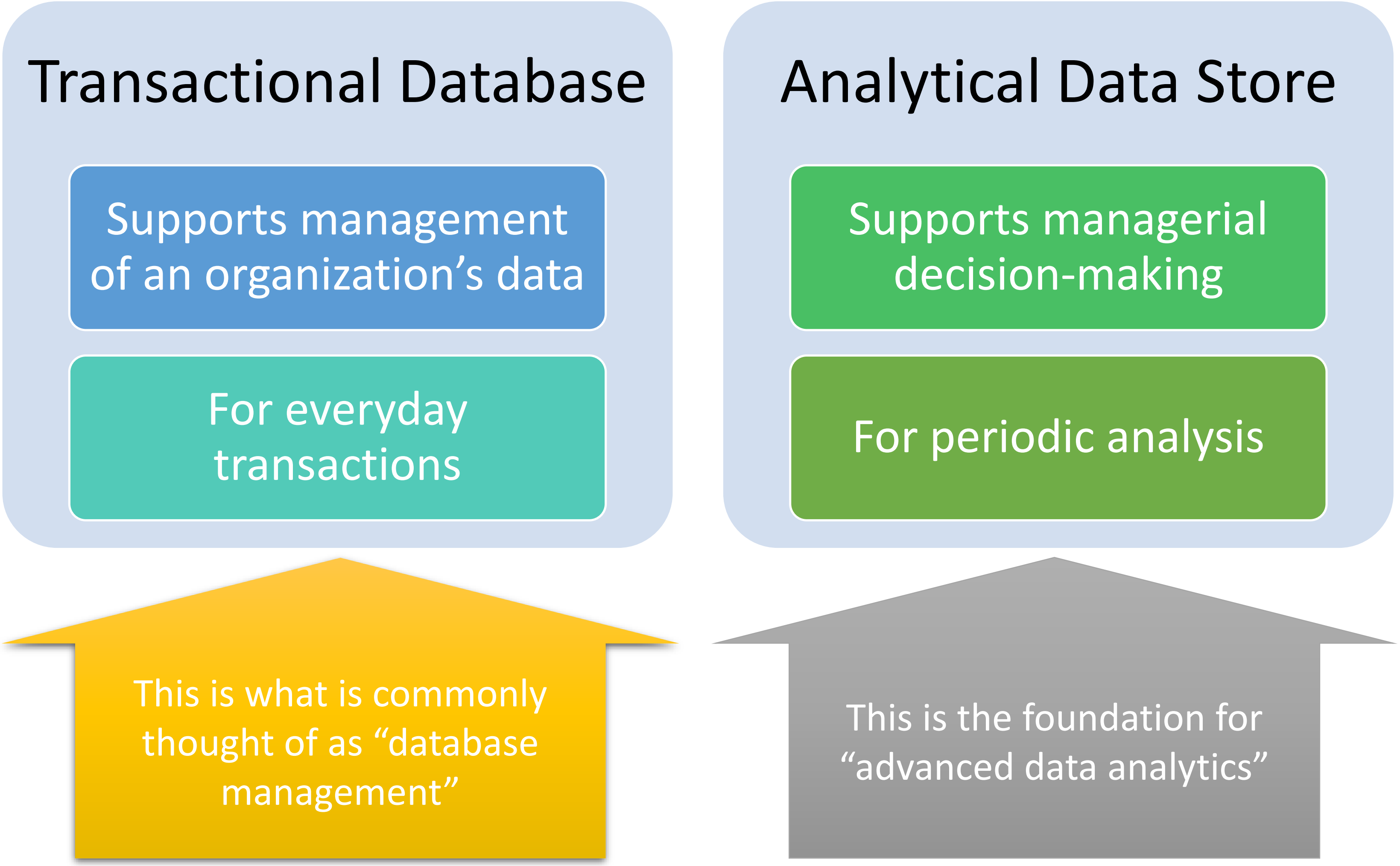
Transactional

- Captures data describing and event
- An exchange between actors
- Real-time

Analytical

- Captures data to support analysis and reporting
- An aggregated view of the business
- Historical

Transactional Database/Analytical Data Store



Transactional Database

Supports management of an organization’s data

For everyday transactions

This is what is commonly thought of as “database management”

Analytical Data Store

Supports managerial decision-making

For periodic analysis

This is the foundation for “advanced data analytics”

The Transactional Database

- Stores real-time, transactional data

In business, a transaction is the exchange of information, goods, or services.

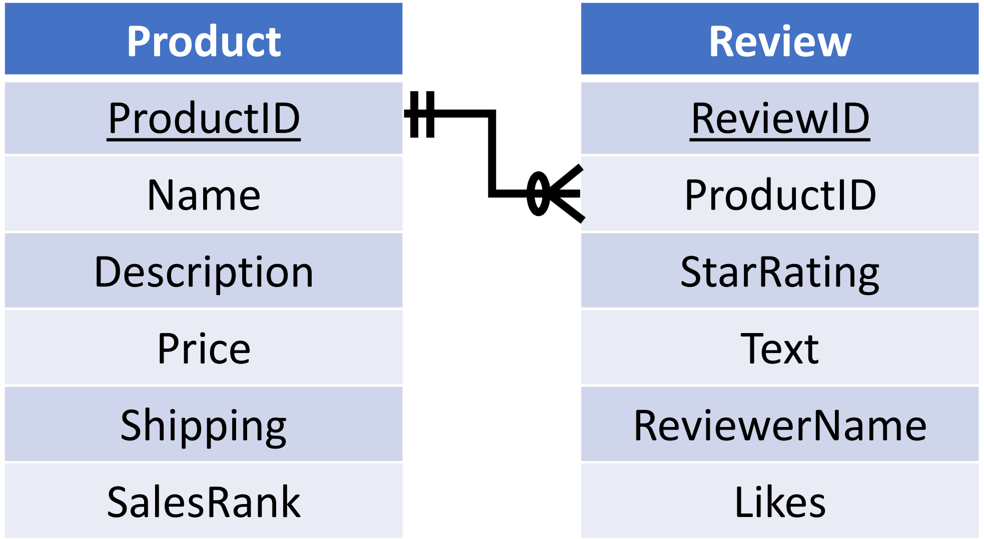
For databases, a transaction is an action performed in a database management system.

transactional databases deal with both: they store information about business transactions using database transactions

- Examples of transactions
  - ✓Purchase a product
  - ✓Enroll in a course
  - ✓Hire an employee
- Data is in real-time
  - ✓Reflects current state
  - ✓How things are “now”

The Relational Database

- Most popular transactional database to collect and store transaction data
- Two primary goals: Minimize redundancy
  - ✓Reduce errors
  - ✓Less space required
- An example
  - ✓A series of tables with logical associations between them
  - ✓The associations (relationships) allow the data to be combined
-



Product
<u>ProductID</u>
Name
Description
Price
Shipping
SalesRank

Review
<u>ReviewID</u>
ProductID
StarRating
Text
ReviewerName
Likes

Analyzing Transactional Data in Relational Database

- Can be difficult to do from a relational database

- Having multiple tables is good for storage and data integrity, but bad for analysis
  - ✓ Tables must be “joined” together before analysis can be done

- The solution is the Analytical Data Store

Relational databases are optimized for storage efficiency, not retrieval

Analytical data stores are optimized for retrieval and analysis, not storage efficiency and data integrity

Comparing Transactional and Analytical Data Stores

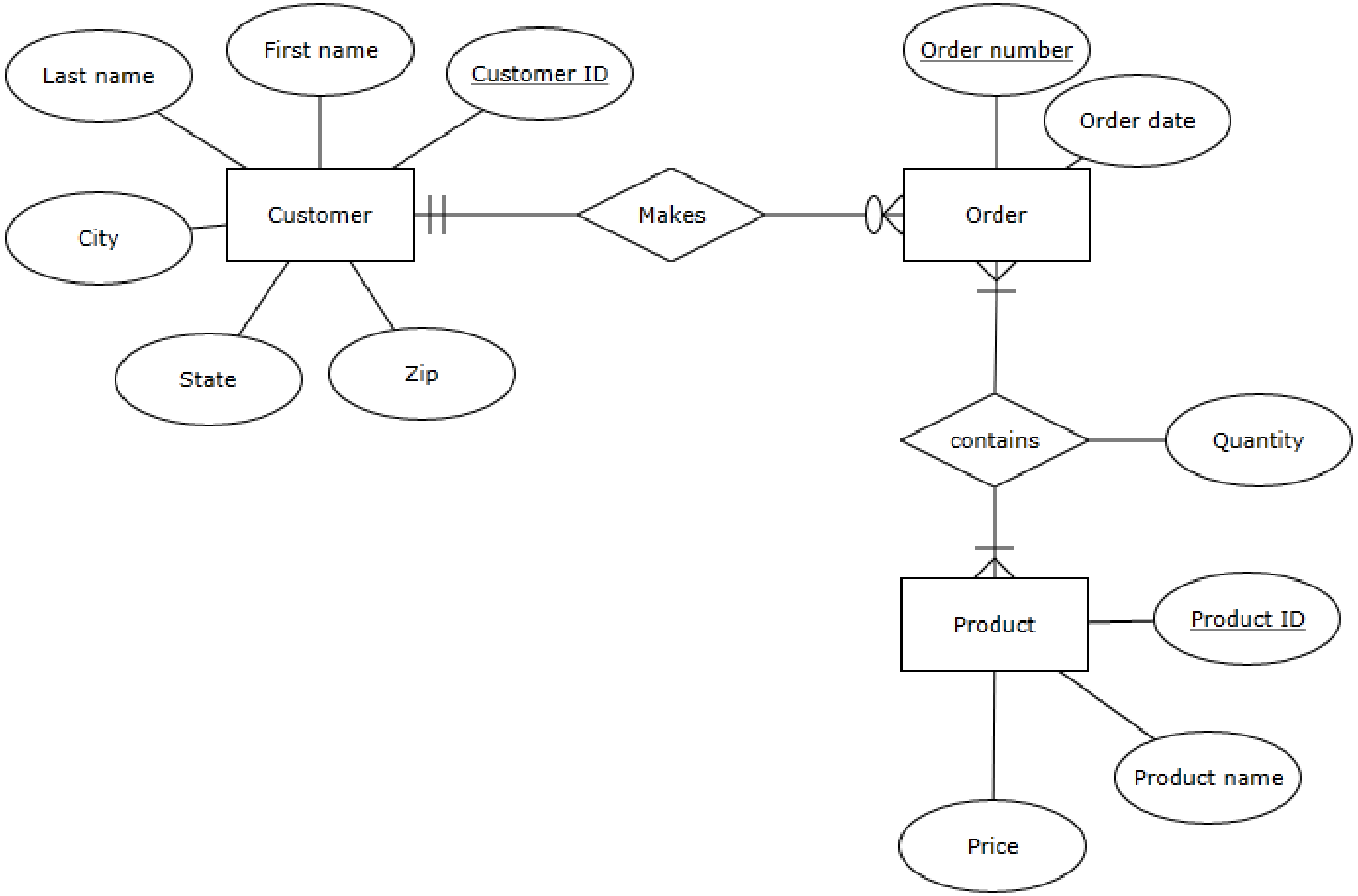
Transactional Database	Analytical Data Store
Based on Relational paradigm	Based on Dimensional paradigm
Storage of real-time transactional data	Storage of historical transactional data
Optimized for storage efficiency and data integrity	Optimized for data retrieval and summarization
Supports day-to-day operations	Supports periodic and on-demand analysis

Relational Data Modeling

ERD and Schema

The Entity Relationship Diagram (ERD)

- The primary way of modeling a relational database
- An Example of ERD



How to Draw ERD

- Step 1: Identify all entities and the corresponding attributes
  - ✓Entities are nouns
  - ✓Entity must have attributes (database is not an entity)
  - ✓Entity must be connected to (at least) other entity
  - ✓Entity must have primary key to uniquely identify it
    - Last Name, Street, Product Name cannot be primary key
- Step 2: Identify relationship attributes/Implement relationships
  - ✓Relationship is inferred from problem statement
  - ✓Relationship attributes depends on both entities
- Step 3: Implement cardinality
  - ✓Maximum cardinality: 1:1, 1:m, m:m
  - ✓Minimum cardinality: 1 or 0

Normalization

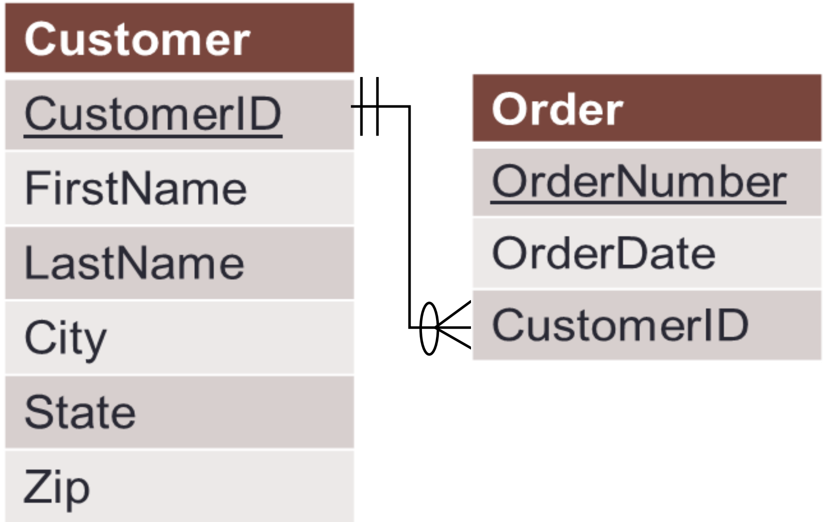
- Normalization
  - ✓Organizing data to minimize redundancy (repeated data)
  - ✓It’s easier to make changes to the data

- If an entity has multiple sets of related attributes, split them up into separate entities
  - ✓Vender phone, Vender name, Vender Address → Vender (entity)

- Each attribute should be atomic – you can’t (logically) break it up any further
  - ✓Address → State, City, Street, Zip

Draw Schema from an ERD

- Schema is a serials of connected tables



Customer

- CustomerID
- FirstName
- LastName
- City
- State
- Zip

Order

- OrderNumber
- OrderDate
- CustomerID

# Tables = # Entities + # m:m relationships

1. Each entity becomes a table

2 Each entity’s attribute becomes a field (column)

3. Implement relationships between the tables

1:many relationships

- Primary key field of “1” table put into “many” table as foreign key field

many:many relationships

- Create new table
- 1:many relationships with original tables

1:1 relationships

- Primary key field of one table put into other table as foreign key field

Join Tables / De-normalized data

Customer Table

CustomerID	FirstName	LastName	City	State	Zip
1001	Greg	House	Princeton	NJ	09120
1002	Lisa	Cuddy	Plainsboro	NJ	09123
1003	James	Wilson	Pittsgrove	NJ	09121
1004	Eric	Foreman	Warminster	PA	19111

Order Table

Order Number	OrderDate	Customer ID
101	3-2-2011	1001
102	3-3-2011	1002
103	3-4-2011	1001
104	3-6-2011	1004

Order Table			Customer Table					
Order Number	OrderDate	Customer ID	Customer ID	FirstName	LastName	City	State	Zip
101	3-2-2011	**1001**	**1001**	Greg	House	Princeton	NJ	09120
102	3-3-2011	**1002**	**1002**	Lisa	Cuddy	Plainsboro	NJ	09123
103	3-4-2011	**1001**	**1001**	Greg	House	Princeton	NJ	09120
104	3-6-2011	**1004**	**1004**	Eric	Foreman	Warminster	PA	19111

Extracting Data with SQL

Query a Single Table

```
SELECT [DISTINCT] expression(s)
FROM schema_name.table_name(s)
[WHERE condition(s)]
[GROUP BY expression(s)]
[ORDER BY expression(s) [ ASC | DESC ]] ;
```

The [] means the element is optional

Element	Description
expression(s)	The column(s) or function(s) that you wish to retrieve.
schema_name.table_name(s)	The table(s) that you wish to retrieve records from.
DISTINCT	Optional. Return unique values.
WHERE condition(s)	Optional. The conditions that must be met for the records to be selected.
GROUP BY expression(s)	Optional. Organize the results by column values.
ORDER BY expression(s)	Optional. Sort the records in your result set

Refer to Columns and Tables

- Refer to Columns
  - ✓When query from a SINGLE table, *ColumnName* only will be ok
  - ✓When query from MULTIPLE tables, column must be specified as *Tablename.columnName*
  - ✓It’s always safe to use *Tablename.columnName*
  - ✓Columns should be separated with commas “,”

- Refer to Tables
  - ✓In SQL, tables should always be specified as *SchemaName.TableName*

WHERE conditions

- The following list of operators that can be used in the WHERE clause

- We can use AND/OR to connect those operators

Operator	Description
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

SQL Functions

- Five SQL functions
  - COUNT() - Returns the number of rows
  - MAX() - Returns the largest value
  - MIN() - Returns the smallest value
  - AVG() - Returns the average value
  - SUM() - Returns the sum
- Tips about SQL functions
  - The SQL functions return ONE value
  - The SQL functions return MULTIPLE values (rows) when used with GROUP BY
  - A SQL function is “similar to” a column when used after SELECT. It means, we can treat a SQL function [e.g., COUNT(), AVG()] as a column
  - COUNT(*) returns all the rows

GROUP BY

- GROUP BY is used in conjunction with the aggregate functions (COUNT, MAX, MIN, AVG, SUM), to group the results by one or more columns
- Examples:

```
SELECT State, COUNT(*)
FROM  orderdb.Customer
GROUP BY State;
```

State	COUNT(*)
NJ	3
PA	1

```
SELECT ProductID, SUM(Quantity)
FROM  orderdb.`Order-Product`
GROUP BY ProductID;
```

ProductID	SUM(Quantity)
2251	7
2282	5
2505	12

Quotes and ORDER BY

- Simple (double) quotes
  - ✓ For strings
  - ✓Often refer to the value of a certain cell
    - Column FirstName → ‘Alvin’
    - Column ProductName →”Iphone 8 Plus”
- Back quotes
  - ✓For schema name, table name and column name
  - ✓When contains blank space or special characters such as `Order-Product`, `Last Name`
  - ✓ Or crash with SQL reserved words such as `Order`