Database Management

Let's talk about how databases are really used in industry ...

Can you think of a REAL database that stores data about YOU?

- School: Registration, Classes, Grades
- Employer: Payroll
- Government: Taxes, ID Card
- Cellphone: Location, Apps, Calls, Texts
- Online: Social Media, Shopping,
  Entertainment, Google, Banking

Database Management

For each of these ...

How big is this database?
What happens if the database is DOWN?

- School: No grades, no records
- Employer: No pay
- Government: No taxes, no travel
- Cellphone: No apps, no calls, no texts
- Online: No shopping, no banking, no web searches

Database Management

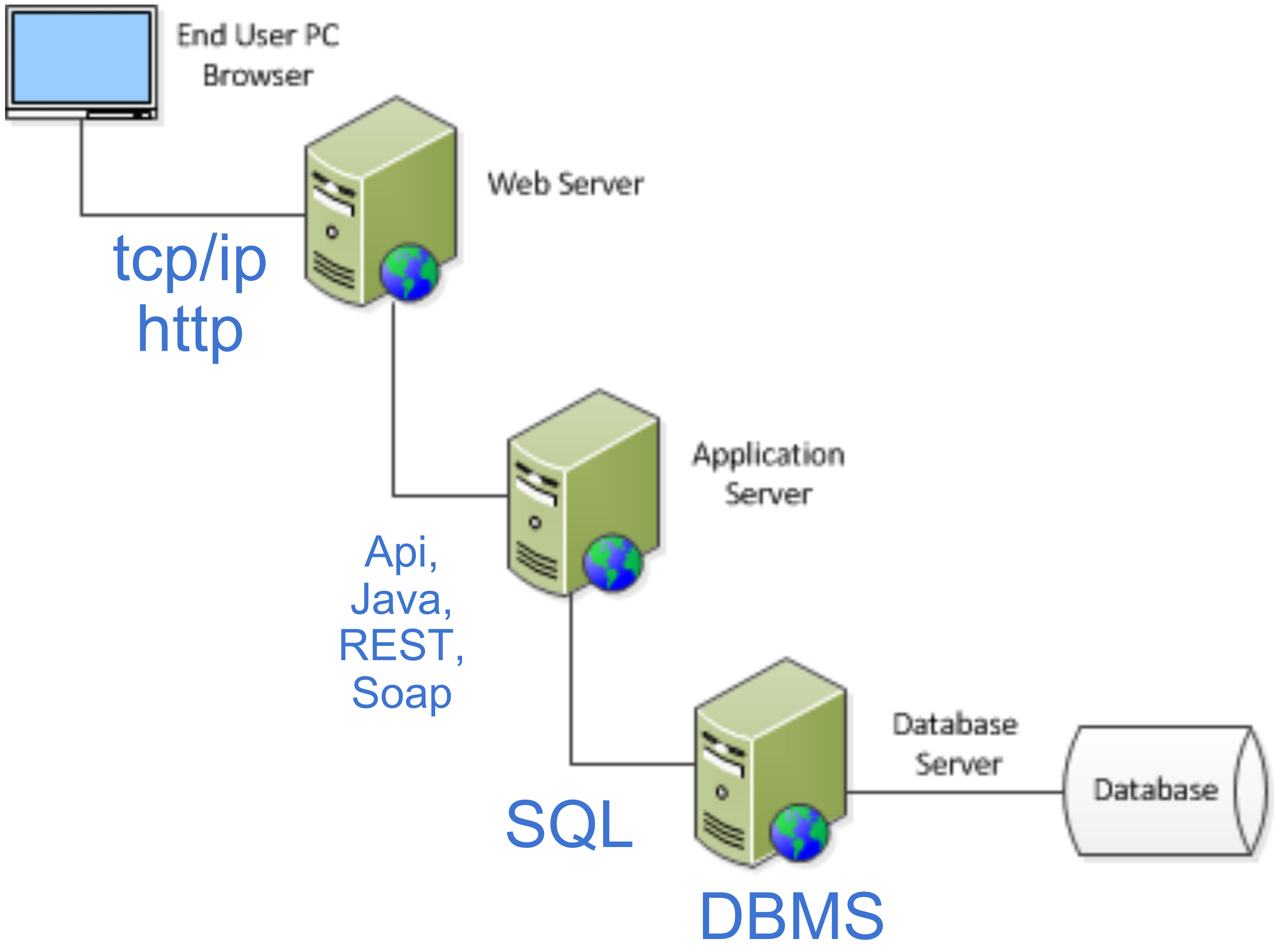
In our Modern Society:

We cannot tolerate downtime due to broken databases !

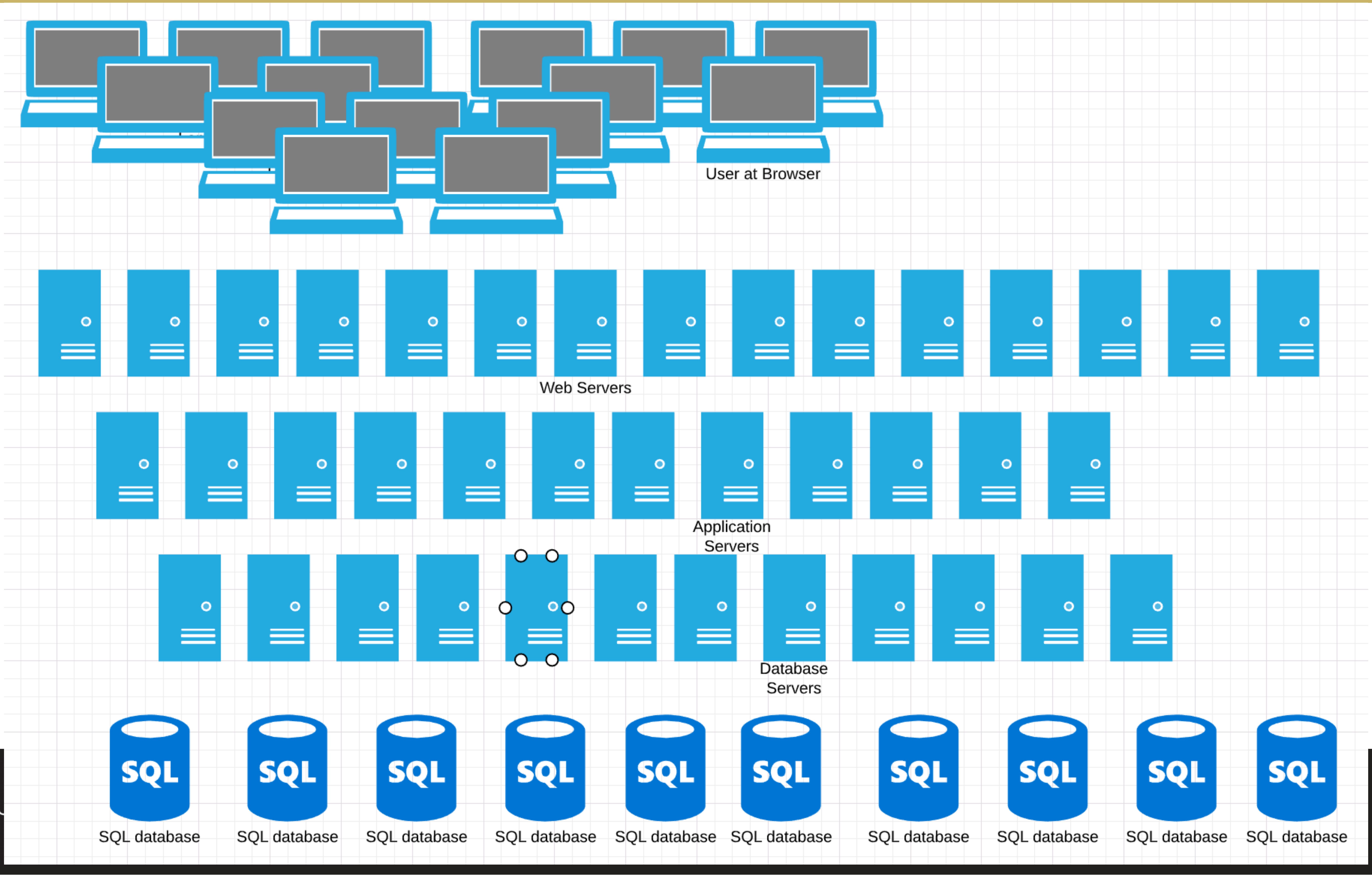
Can you do YOUR job if your organization's databases are down?

How can we make sure that our databases remain healthy and always available?

Application Architecture



Application Architecture



Database Management

Some Big Challenges

The database may be HUGE --

- Tens of thousands of tables
- Millions/Billions of rows in large tables
- Queries can take a long, long time
- Moving/Copying data can take a long time

Database Management

Some Big Challenges

Thousands/Millions of users:

- Hitting the same table at the same time
- May request concurrent updates of the same data

Privacy & Security

- How do we prevent one user's updates from overlaying another user's work?

Database Management

Consider:

What if the database is down?

What if the database is slow?

What if the database crashes?

Hospitals, Banks, StockExchanges/Brokers, Universities

What is the COST of an unavailable database?

Database Management

LEARNING OBJECTIVE

- Understand how TRANSACTION LOG processing allows the DBMS to handle such challenges

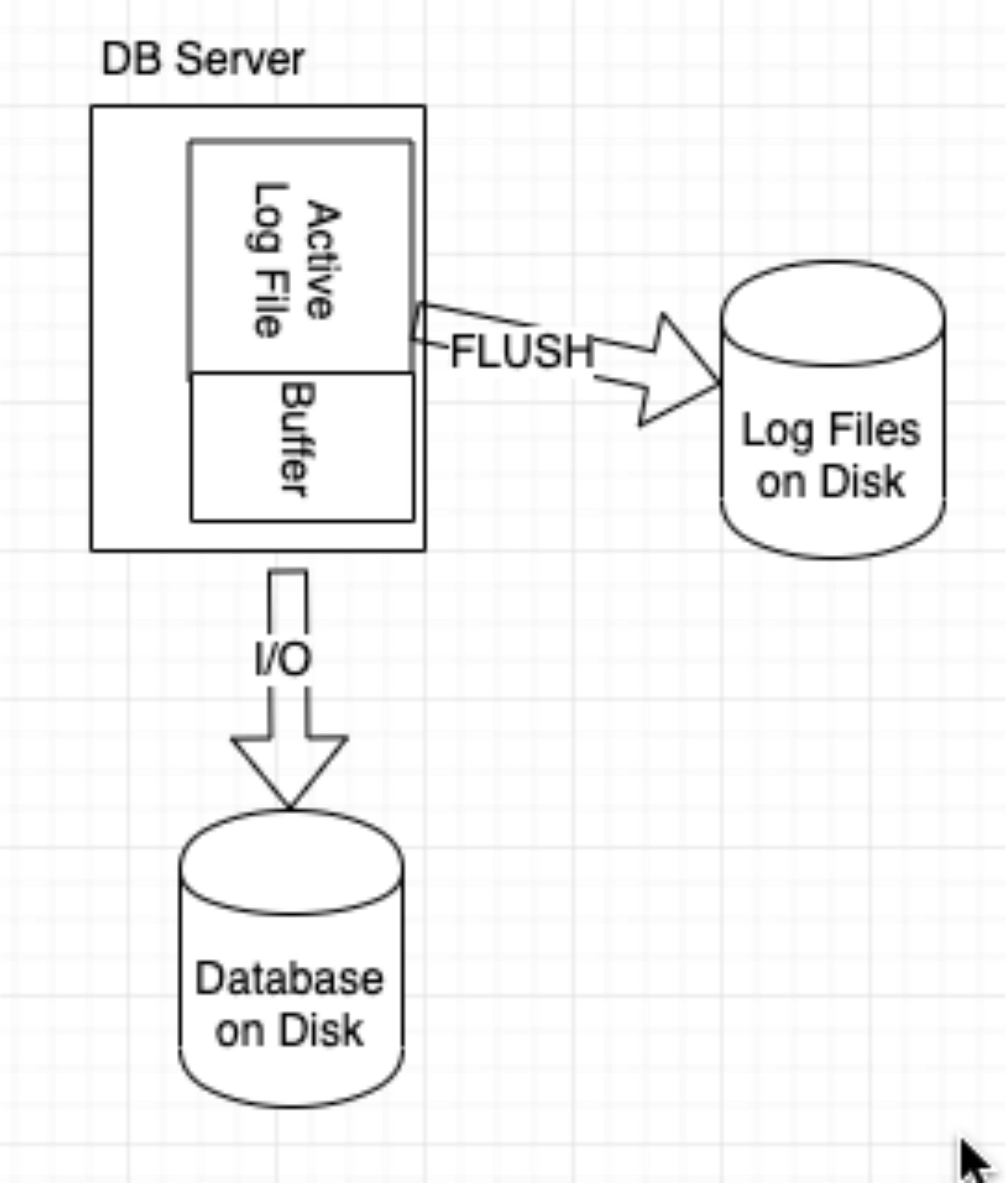
DBMS Transaction Logging

Every activity against the database that changes the database is written to a log file.

- “Binary Logs” or “BinLogs” in MySQL.
- “WAL” (write ahead log) in PostgreSQL and SQLite.
- “RedoLogs” in Oracle
- “Transaction Logs” in MS SQL Server

DBMS Transaction Logging

The log file:



DBMS Transaction Logging

The LOG file is sequential file, NOT a database table

Log files are BINARY -- Not human-readable

The DBMS software provides a utility to allow database administrators to view log contents in human-readable format

DBMS Transaction Logging

Active Log files reside in memory and are written to disk periodically. (In MySQL, this is called a “flush”.)

Transactions are written to the log file first, then written to the database. (Write to log is sequential and fast.)

Changes to database tables on disk are written only after those changes have been logged

DBMS Transaction Logging

Log files can be automatically written to disk

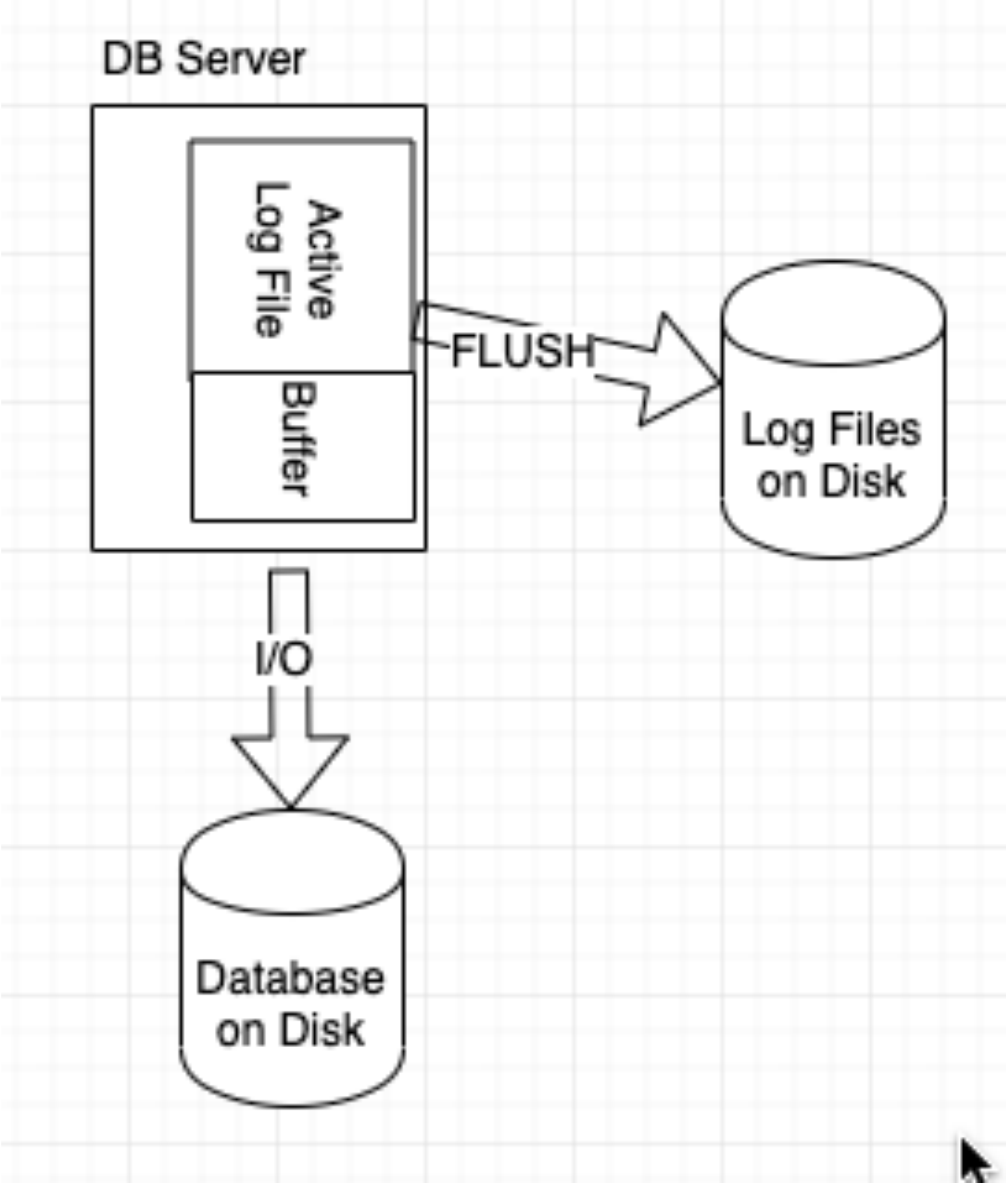
- On a timer, every *nn* minutes
- On a size limit, whenever the log file grows to *nn* MB or GB
- Manually by a DBA (executing a FLUSH LOGS command)

Closes out the old log file and opens up a new one

- Increments the log number in the filename

DBMS Transaction Logging

Define Commit, Rollback



Database changes are cached in memory

Commit = updated cache buffers are written to disk. Log is updated.

Rollback = updated cache buffers are undone by replaying the logs. Log is updated.

DBMS Transaction Logging

Sample Log File (for illustration)

Log Sequence Number ("LSN")	Tranasaction ID	Previous LSN	Object	Operation	Row Before Image	Row After Image	Start Time	End Time
00042:001	T1	00000:000		Start Transaction			MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:002	T1	00042:001	CUST 10123	MODIFY	(Before Image)	(After Image)	MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:003	T2	00000:000		Start Transaction			MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:004	T1	00042:002	INV	INSERT		(After Image)	MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:005	T1	00042:004	JRNL	INSERT		(After Image)	MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:006	T2	00042:003	INV 38596	MODIFY	(Before Image)	(After Image)	MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:007	T3	00000:000		Start Transaction			MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:008	T1	00042:005		COMMIT			MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:009	T3	00042:007	CUST 30456	MODIFY	(Before Image)	(After Image)	MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:010	T3	00042:009		COMMIT			MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:011	T2	00042:006	JRNL	INSERT		(After Image)	MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:012	T2	00042:011	INV 54987	DELETE	(Before Image)		MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS
00042:013	T2	00042:012		COMMIT			MMDDYYYYTHH:MM:SSSS	MMDDYYYYTHH:MM:SSSS

DBMS Transaction Logging

- A transaction can consist of multiple updates to multiple database objects
- Logs can be used to UNDO a transaction or REDO a transaction.
- Logs capture an image of the row BEFORE and AFTER it was changed.
- Logs capture start/stop timestamps for every activity.

Transaction Control

- The transaction, or logical unit of work (LUW), can be a series of actions taken against the database
- The steps of a transaction must occur as an atomic unit, that is:
- Either ***ALL*** actions in a transaction are committed, or ***NONE*** of them can be.

Transaction Control

Payroll Example

When we pay someone:

1. Update the CHECK REGISTER
2. Create accounting JOURNAL ENTRIES
    - 2.1 Debit CASH
    - 2.2 Credit SALARIES PAYABLE

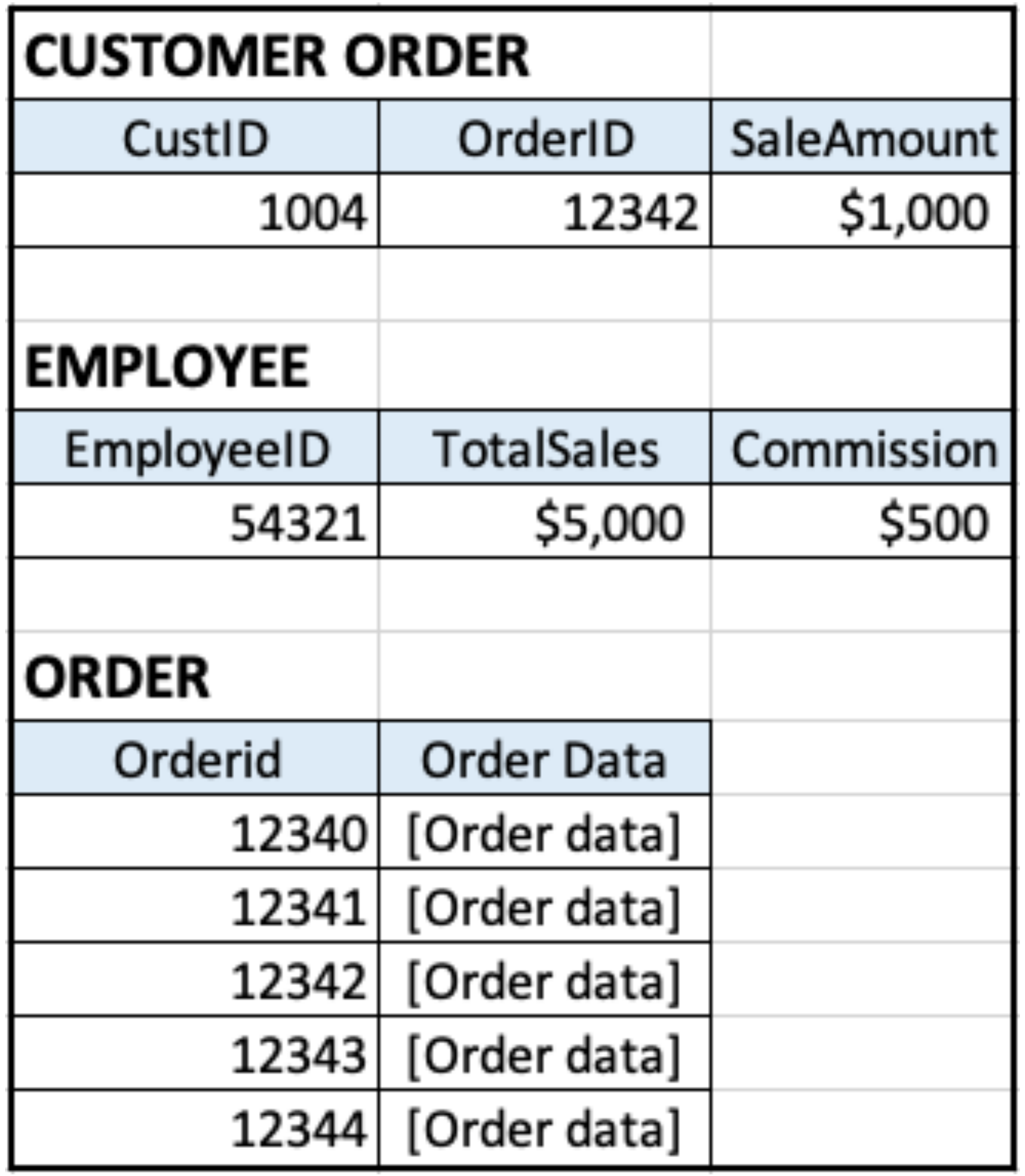
All three updates must take place together or the accounting books will be out of balance.

Transaction Control

BEFORE

What happens without transaction control

AFTER



CUSTOMER ORDER

CustID	OrderID	SaleAmount
1004	12342	$1,000

EMPLOYEE

EmployeeID	TotalSales	Commission
54321	$5,000	$500

ORDER

Orderid	Order Data
12340	[Order data]
12341	[Order data]
12342	[Order data]
12343	[Order data]
12344	[Order data]

Customer 1004 places a new order for $1,200
(INSERT a new CUSTOMER ORDER Row)

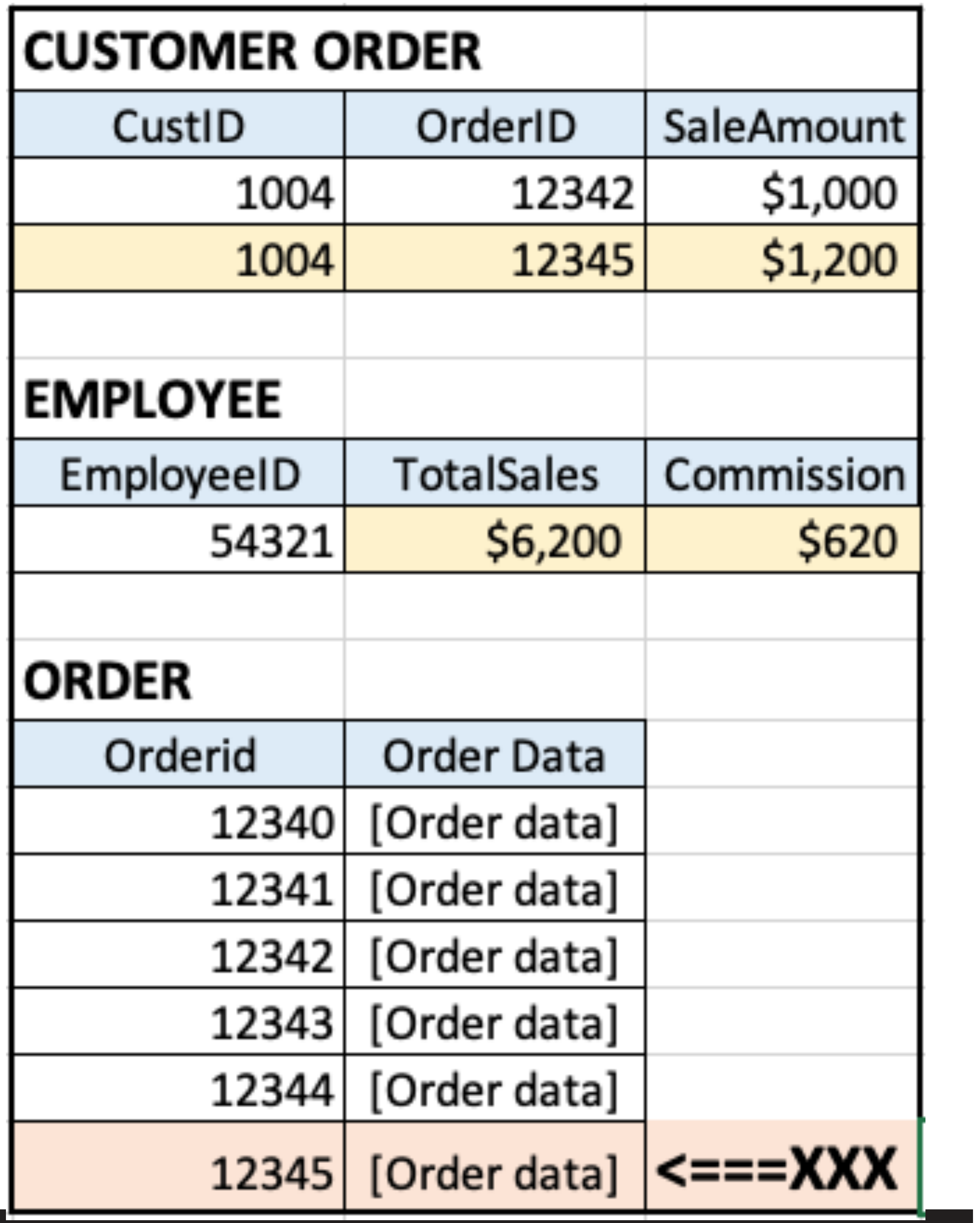
Salesperson 54321 earns a $120 commission
(UPDATE the EMPLOYEE Row)

Add the new order record
(INSERT a new ORDER Row)

BUT

The ORDER Table is full and the INSERT fails!

Customer, Employee and Order data is
out of sync !!



CUSTOMER ORDER

CustID	OrderID	SaleAmount
1004	12342	$1,000
1004	12345	$1,200

EMPLOYEE

EmployeeID	TotalSales	Commission
54321	$6,200	$620

ORDER

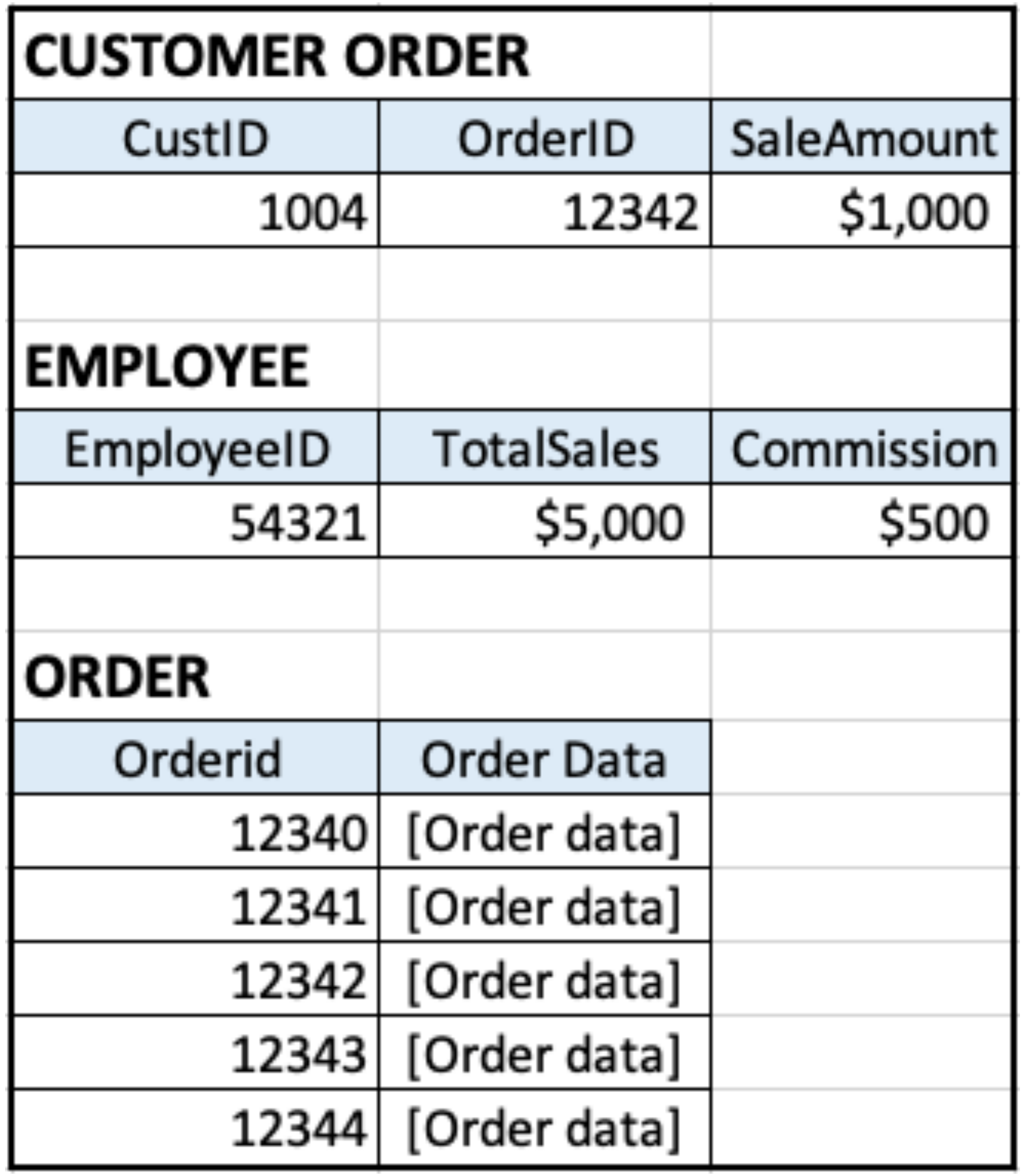
Orderid	Order Data	
12340	[Order data]	
12341	[Order data]	
12342	[Order data]	
12343	[Order data]	
12344	[Order data]	
12345	[Order data]	<===XXX

Transaction Control

BEFORE

What happens WITH transaction control

AFTER



CUSTOMER ORDER		
CustID	OrderID	SaleAmount
1004	12342	$1,000
EMPLOYEE		
EmployeeID	TotalSales	Commission
54321	$5,000	$500
ORDER		
Orderid	Order Data	
12340	[Order data]	
12341	[Order data]	
12342	[Order data]	
12343	[Order data]	
12344	[Order data]	

Customer 1004 places a new order for $1,200
(INSERT a new CUSTOMER ORDER Row)

Salesperson 54321 earns a $120 commission
(UPDATE the EMPLOYEE Row)

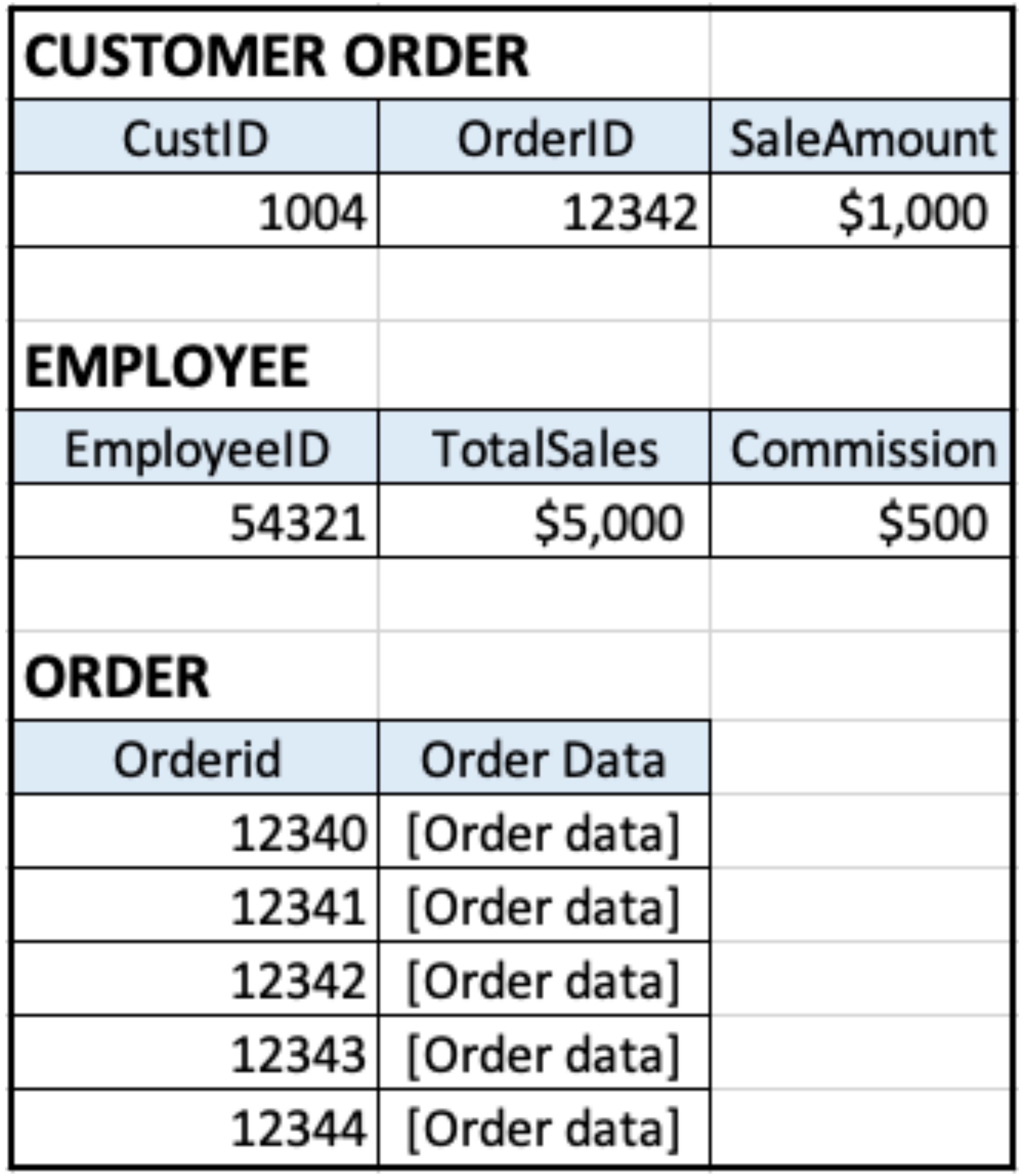
Add the new order record
(INSERT a new ORDER Row)

BUT

The ORDER Table is full and the INSERT fails!

DBMS detects the failure.

All changes are ROLLED BACK
by the DBMS replaying the LOGS



CUSTOMER ORDER		
CustID	OrderID	SaleAmount
1004	12342	$1,000
EMPLOYEE		
EmployeeID	TotalSales	Commission
54321	$5,000	$500
ORDER		
Orderid	Order Data	
12340	[Order data]	
12341	[Order data]	
12342	[Order data]	
12343	[Order data]	
12344	[Order data]	

Transaction Control

Summary

- The DBMS logs all activity
- Transactions are an atomic set (all or none)
- DBMS will COMMIT the transaction set, OR

  will ROLLBACK the transaction set