[544] SQL Databases

Tyler Caraza-Harter

Learning Objectives

- explain the motivation for using an ETL (extract transform load) process to copy data from an transactions processing system to an analytics processing system
- create database schemas with types and keys
- use database transactions to group multiple updates together
- write SQL queries with common clauses (SELECT, FROM, JOIN, WHERE, GROUP BY, HAVING, ORDER BY, and LIMIT) to answer questions about data

Outline

Databases

- tables and queries
- architecture
- transactions vs. analytics

Creating/designing tables

Transactions

Queries

Demos

Tables

tbl_purpose

id	loan_purpose
1	Home purchase
2	Home improvement
3	Refinancing

tbl_state

code	abbr	name
1	AL	Alabama
2	AK	Alaska
4	AZ	Arizona
5	AR	Arkansas
6	CA	California
8	CO	Colorado
9	CT	Connecticut
10	DE	Delaware
…	…	…

string

tbl_action

id	action_taken
1	Loan originated
2	Application approved but not accepted
3	Application denied by financial institution
4	Application withdrawn by applicant
5	File closed for incompleteness
6	Loan purchased by the institution
7	Preapproval request denied by financial
8	Preapproval request approved but not accepted

tbl_loan

id	purpose	action	state	amount	rate
1	2	1	2	20000	5.0
2	1	1	8	300000	3.0
3	1	4	10	450000	3.2
…	…	…	…	…	…

int float

Databases store a collection of tables

- schemas define the columns/types for each table
- IDs/keys let us relate multiple tables
  (for example, the first loan is in Alaska)

Queries

tbl_purpose

id	loan_purpose
1	Home purchase
2	Home improvement
3	Refinancing

tbl_state

code	abbr	name
1	AL	Alabama
2	AK	Alaska
4	AZ	Arizona
5	AR	Arkansas
6	CA	California
8	CO	Colorado
9	CT	Connecticut
10	DE	Delaware
…	…	…

tbl_action

id	action_taken
1	Loan originated
2	Application approved but not accepted
3	Application denied by financial institution
4	Application withdrawn by applicant
5	File closed for incompleteness
6	Loan purchased by the institution
7	Preapproval request denied by financial
8	Preapproval request approved but not accepted

tbl_loan

id	purpose	action	state	amount	rate
1	2	1	2	20000	5.0
2	1	1	8	300000	3.0
3	1	4	10	450000	3.2
…	…	…	…	…	…

Queries let us

- ask questions about the data
  (like, what is the name of the state with "WI" as an abbreviation)
- make changes to the data
  (like insert Puerto Rico as a row in tbl_state)

SQL

tbl_purpose

id	loan_purpose
1	Home purchase
2	Home improvement
3	Refinancing

tbl_state

code	abbr	name
1	AL	Alabama
2	AK	Alaska
4	AZ	Arizona
5	AR	Arkansas
6	CA	California
8	CO	Colorado
9	CT	Connecticut
10	DE	Delaware
…	…	…

tbl_action

id	action_taken
1	Loan originated
2	Application approved but not accepted
3	Application denied by financial institution
4	Application withdrawn by applicant
5	File closed for incompleteness
6	Loan purchased by the institution
7	Preapproval request denied by financial
8	Preapproval request approved but not accepted

tbl_loan

id	purpose	action	state	amount	rate
1	2	1	2	20000	5.0
2	1	1	8	300000	3.0
3	1	4	10	450000	3.2
…	…	…	…	…	…

Structure Query Language (SQL)

- most popular/famous query language
- ask questions about the data: SELECT
- make changes to the data: INSERT, UPDATE, DELETE

SQL

tbl_purpose

id	loan_purpose
1	Home purchase
2	Home improvement
3	Refinancing

tbl_state

code	abbr	name
1	AL	Alabama
2	AK	Alaska
4	AZ	Arizona
5	AR	Arkansas
6	CA	California
8	CO	Colorado
9	CT	Connecticut
10	DE	Delaware
...	...	...

tbl_action

id	action_taken
1	Loan originated
2	Application approved but not accepted
3	Application denied by financial institution
4	Application withdrawn by applicant
5	File closed for incompleteness
6	Loan purchased by the institution
7	Preapproval request denied by financial
8	Preapproval request approved but not accepted

tbl_loan

id	purpose	action	state	amount	rate
1	2	1	2	20000	5.0
2	1	1	8	300000	3.0
3	1	4	10	450000	3.2
...	...	...	...	...	...

Structure Query Language (SQL)

- most popular/famous query language
- ask questions about the data: SELECT
- make changes to the data: INSERT, UPDATE, DELETE

```
SELECT AVG(rate) FROM tbl_loan;
```

analytics (calculate over many/all rows, few colums)

```
SELECT amount, rate FROM tbl_loan WHERE id = 544;
INSERT INTO tbl_loan (...) VALUES (...);
```

transactions (working with whole row or few rows at a time)

Outline

Databases

- tables and queries
- architecture
- transactions vs. analytics

Creating/designing tables

Transactions

Queries

Demos

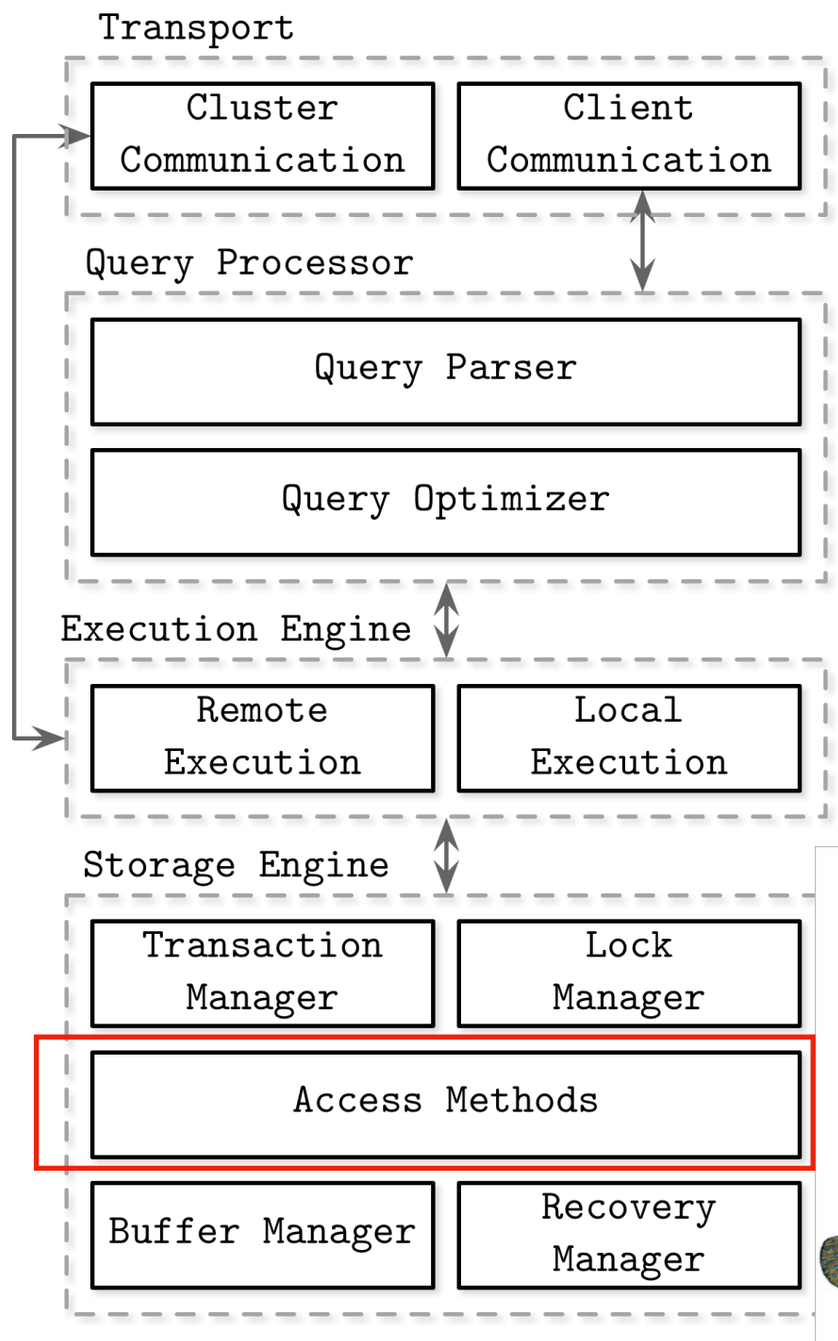
Data Base Management Systems

Architecture: *big picture of a system's components/subsystems*

Databases manage all the resources we've learned about:

- storage
- memory
- network
- compute

example database architecture:



storage structures in files

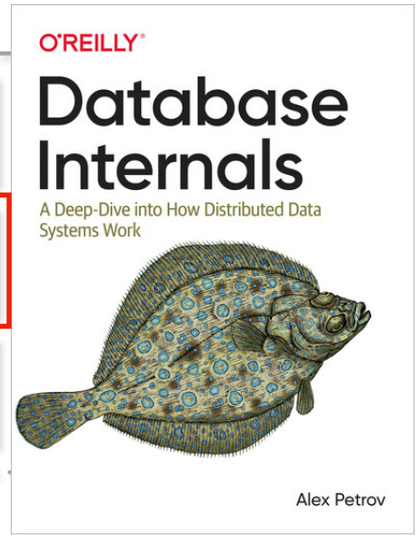


Figure 1-1. Architecture of a database management system (Chapter 1 of Database Internals, by Petrov)

Data Base Management Systems

Architecture: *big picture of a system's components/subsystems*

Databases manage all the resources we've learned about:

- storage
- memory
- network
- compute

example database architecture:

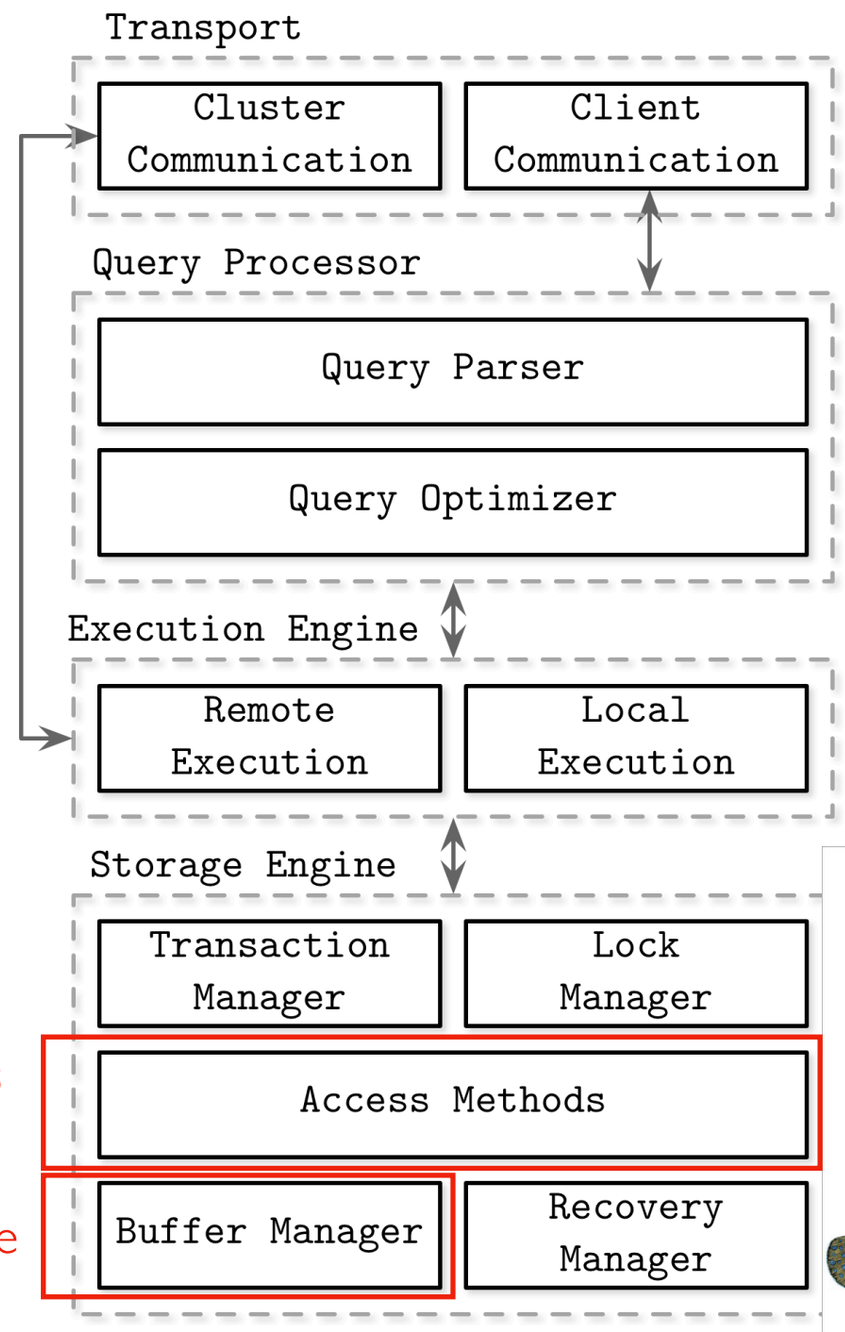
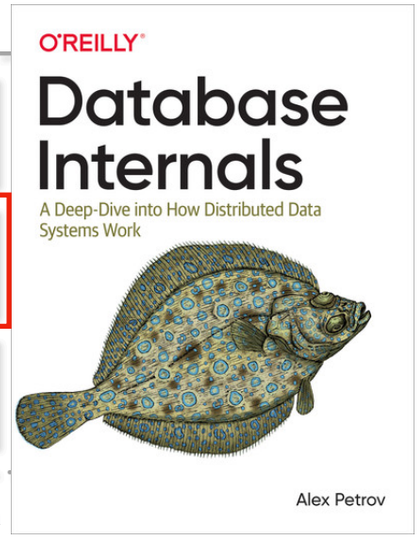


Figure 1-1. Architecture of a database management system (Chapter 1 of Database Internals, by Petrov)



Data Base Management Systems

Architecture: *big picture of a system's components/subsystems*

Databases manage all the resources we've learned about:

- storage
- memory
- network
- compute

example database architecture:

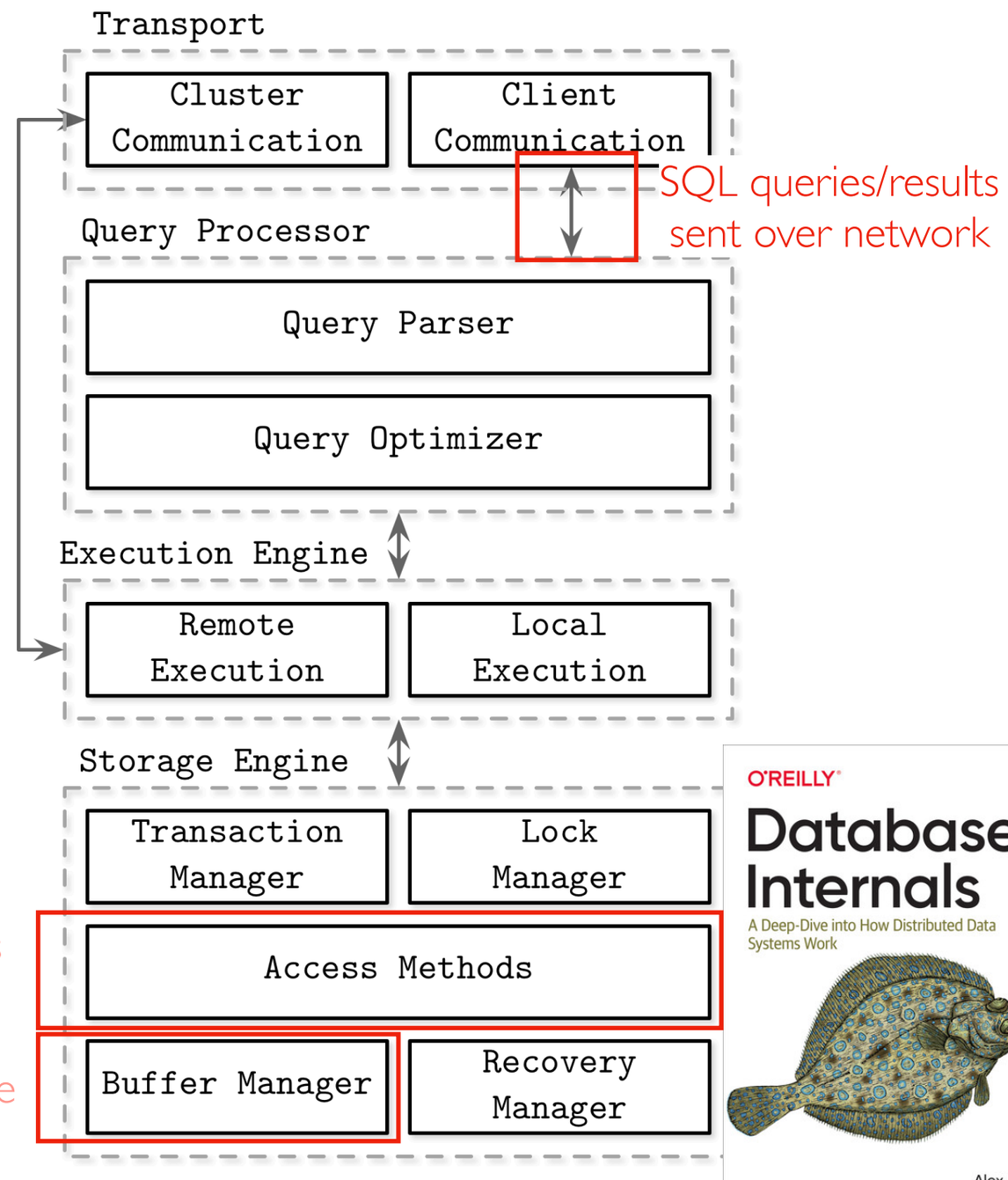


Figure 1-1. Architecture of a database management system (Chapter 1 of Database Internals, by Petrov)

Data Base Management Systems

Architecture: *big picture of a system's components/subsystems*

Databases manage all the resources we've learned about:

- storage
- memory
- network
- compute

example database architecture:

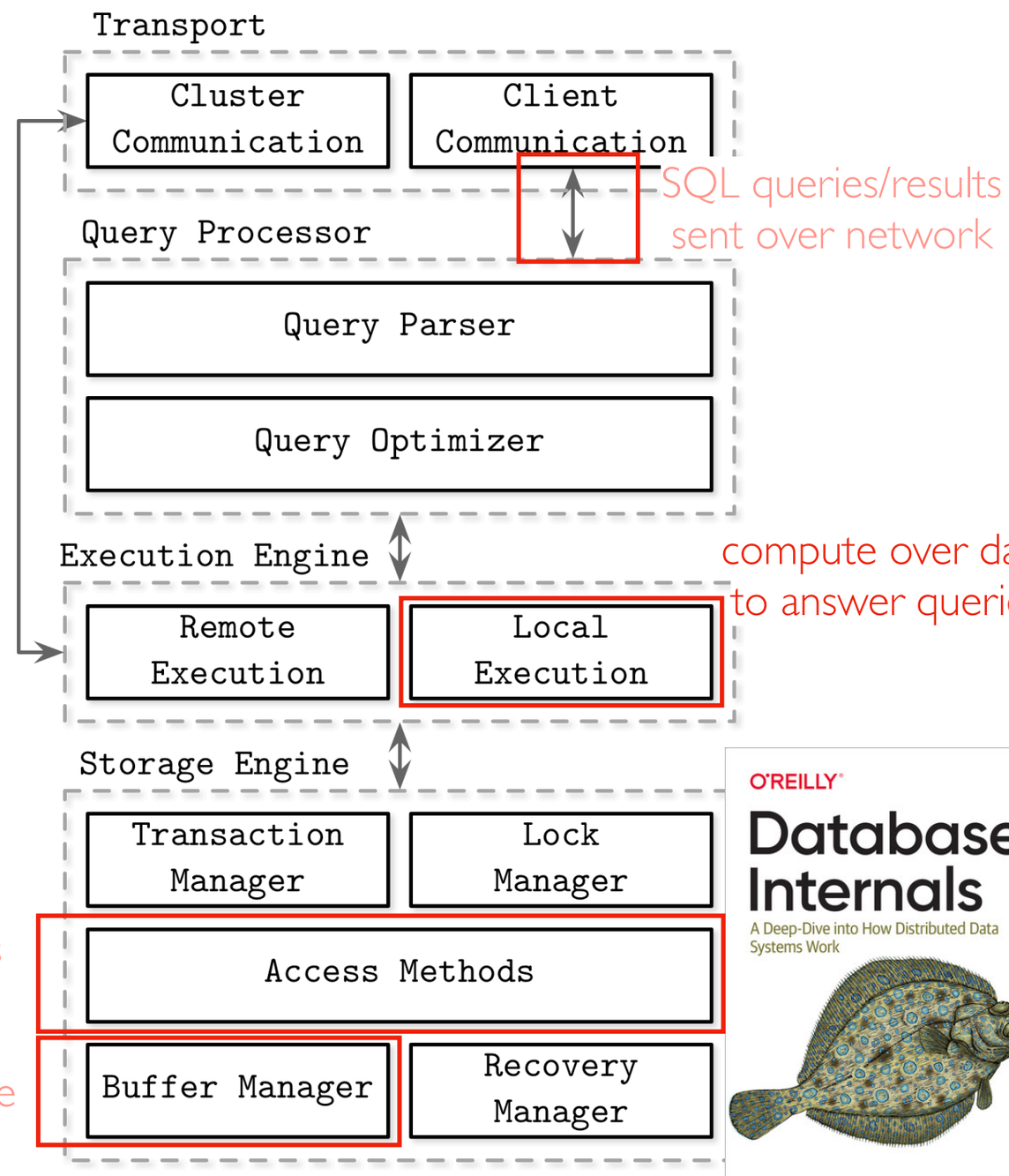
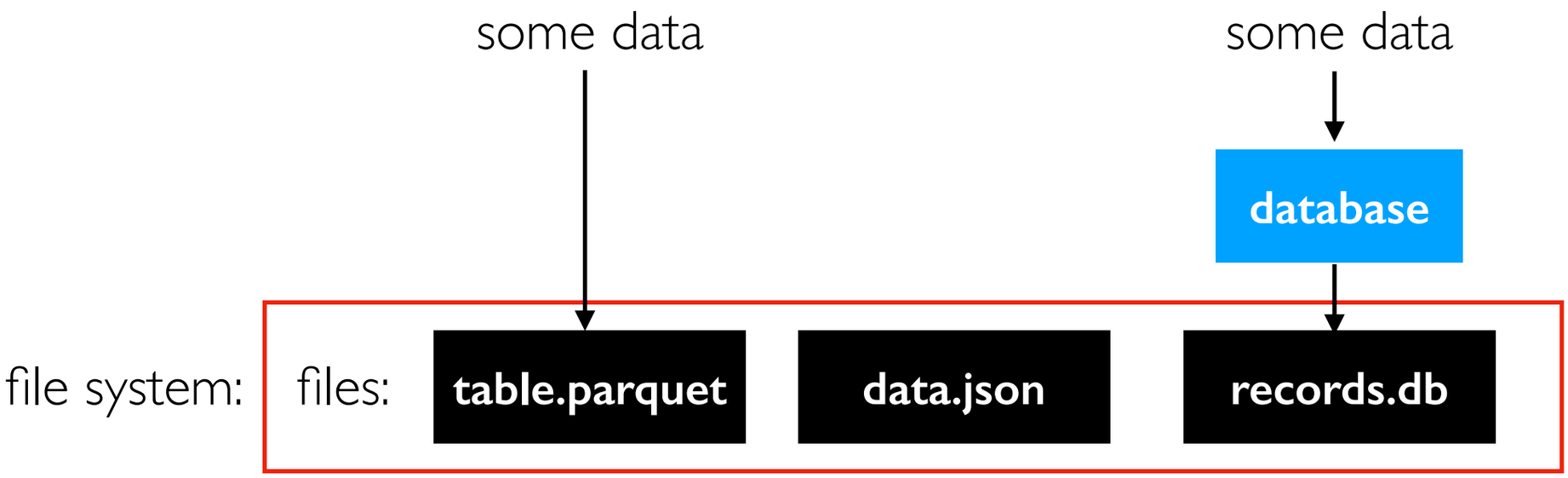


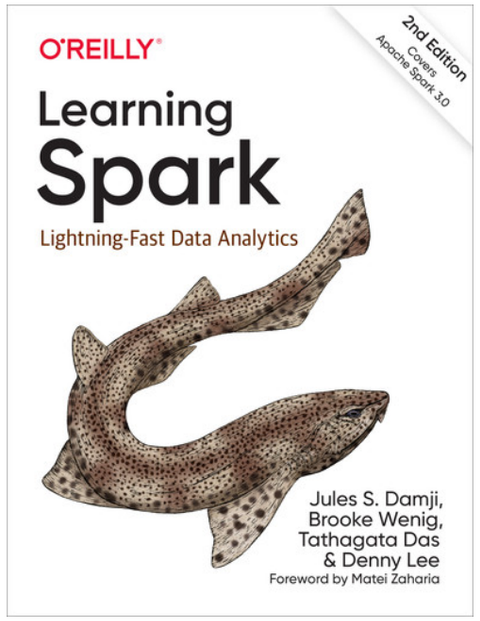
Figure 1-1. Architecture of a database management system (Chapter 1 of Database Internals, by Petrov)

Files vs. Databases (storage+compute coupling)



Databases pros/cons (relative to just using files):

- "*[databases] tightly couple their internal layout of the data and indexes in on-disk files with their highly optimized query processing engines, thus providing very fast computations on the stored data...*"
- "*Databases store data in complex (often proprietary) formats that are typically highly optimized for only that database's SQL processing engine to read. This means other processing tools, like machine learning and deep learning systems, cannot efficiently access the data (except by inefficiently reading all the data from the database).*"



Outline

Databases

- tables and queries
- architecture
- transactions vs. analytics

Creating/designing tables

Transactions

Queries

Demos

Transactions vs. Analytics

```
SELECT AVG(rate) FROM tbl_loan;
```

analytics (calculate over many/all rows, few columns)

```
SELECT amount, rate FROM tbl_loan WHERE id = 544;
INSERT INTO tbl_loan (...) VALUES (...);
```

transactions (working with whole row or few rows at a time)

SQL (as a language) works great for both transactions and analytics

Problem: it's hard for a single database (SQL or otherwise) to be good at both

Main database types:

- OL**T**P (online transactions processing)
- OL**A**P (online analytics processing)

"*The meaning of online in OLAP is unclear; it probably refers to the fact that queries are not just for predefined reports, but that analysts use the OLAP system interactively for explorative queries.*" ~ Kleppmann.

Transactions vs. Analytics

example database architecture:

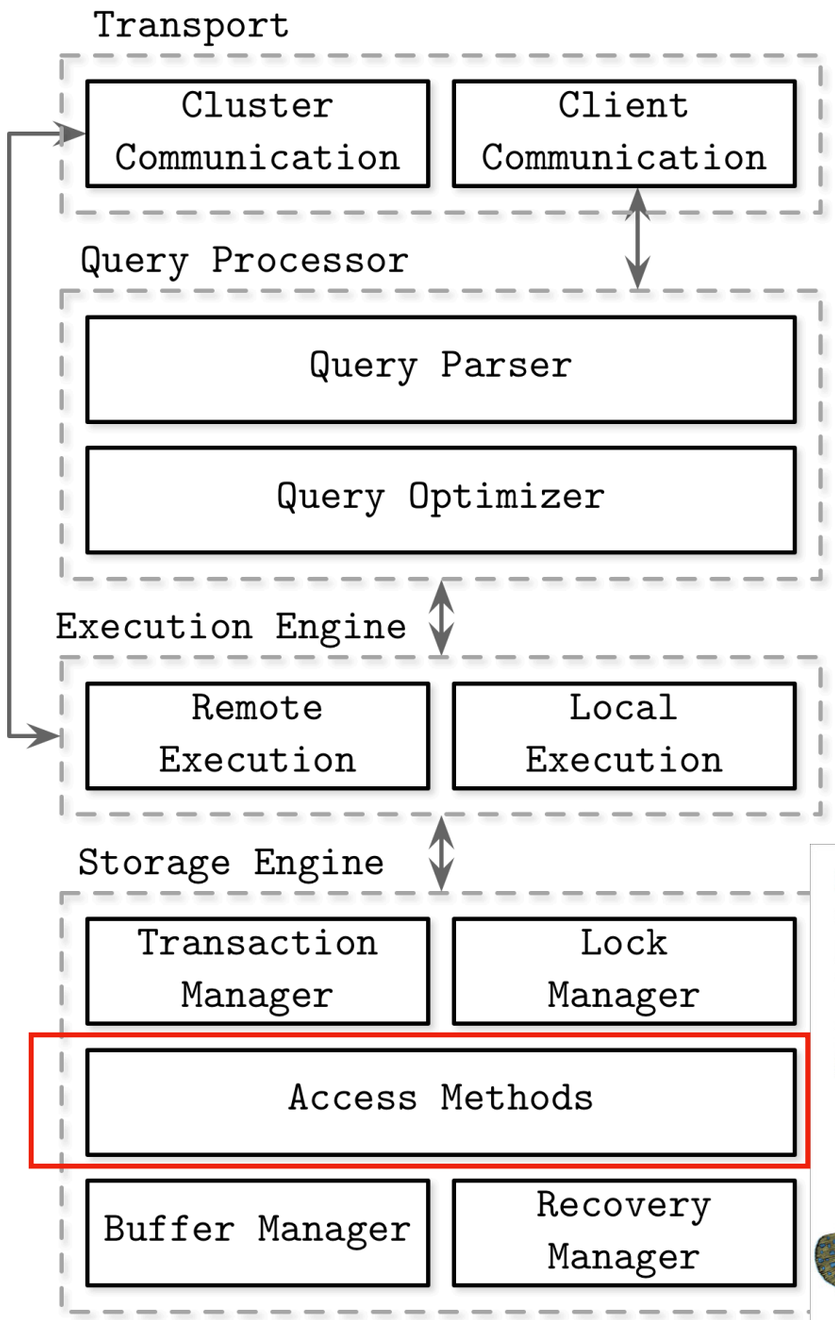
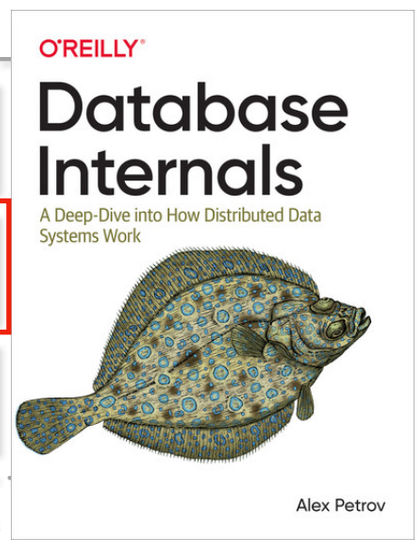


Figure 1-1. Architecture of a database management system (Chapter 1 of Database Internals, by Petrov)

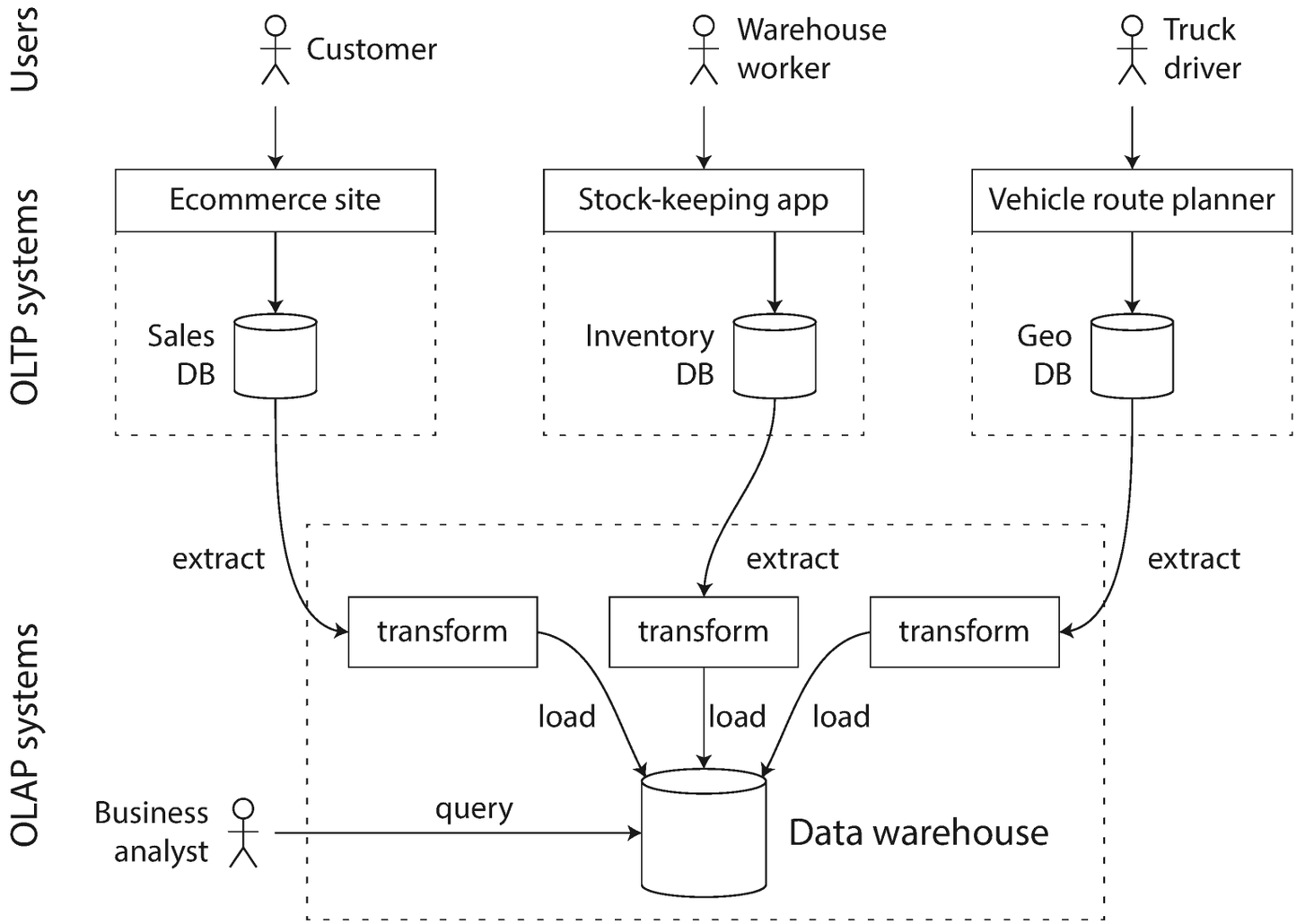


Typical storage design

OLTP: *row oriented data layout*

OLAP: *col oriented data layout*

What if you need transactions AND analytics?



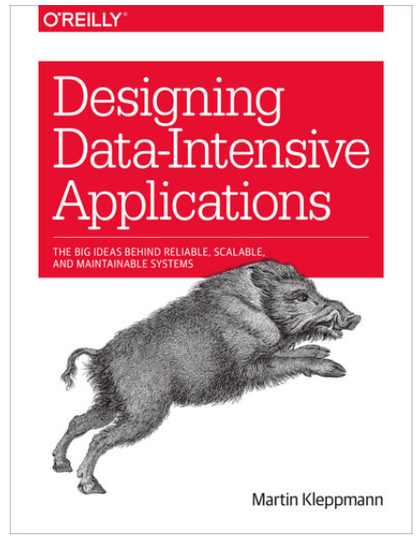


Figure 3-8. Simplified outline of ETL into a data warehouse.
(Chapter 3 of Data-Intensive Applications, by Kleppmann)

Vocab

- Data warehouse: *the OLAP database where we combine data from many sources*
- ETL: *extract-transform-load (process for getting data out of OLTP DBs and into OLAP DB)*

Outline

Databases

Creating/designing tables

- data modeling
- primary/foreign keys

Transactions

Queries

Demos

Data Modeling

Data modeling: deciding how to represent something in an underlying system.

Low-level example (protobufs): how will we represent numbers as bytes being sent over a network?

Traditional Databases: how will we represent things/people/events/etc as rows in tables?

option 1:

tbl_orders

name	book	amount	county	state
Tyler Harter	Designing Data-Intensive Applications	23	Dane	WI
Tyler Harter	Learning Spark	38	Dane	WI
Tyler Harter	Cassandra: The Definitive Guide	39	Dane	WI

Keys and Normalization

SQL keys:

- primary key: uniquely identify a row ("id" in tbl_counties)
- foreign key: reference a primary key ("county_id" in tbl_orders)

In database theory we would say option 2 is "more normalized" (note: there are well-defined normalization levels with formal rules -- we won't get into that in 544)

option 1:

tbl_orders

name	book	amount	county	state
Tyler Harter	Designing Data-Intensive Applications	23	Dane	WI
Tyler Harter	Learning Spark	38	Dane	WI
Tyler Harter	Cassandra: The Definitive Guide	39	Dane	WI

option 2:

tbl_orders

name	book	amount	county_id
Tyler Harter	Designing Data-Intensive Applications	23	1
Tyler Harter	Learning Spark	38	1
Tyler Harter	Cassandra: The Definitive Guide	39	1

tbl_counties

id	county	state
1	Dane	WI
2	Milwaukee	WI
3	La Crosse	WI

Normalization Tradeoffs

Benefits of more normalization:

- avoid inconsistencies
- changes in the real world correspond to fewer changes in the DB
- often save space

Downsides of more normalization:

- queries are sometimes slower
- historical record keeping (for example, if you need to reproduce an invoice prior to somebody's name change, you might want the name at time of purchase)

tbl_orders

name	book	amount	county_id
Tyler Harter	Designing Data-Intensive Applications	23	1
Tyler Harter	Learning Spark	38	1
Tyler Caraza-Harter	Cassandra: The Definitive Guide	39	1

tbl_counties

id	county	state_id
1	Dane	55
2	Milwaukee	55
3	La Crosse	55

tbl_states

id	state
55	WI
...	...

Outline

Databases

Creating/designing tables

Transactions

Queries

Demos

Definitions of Transactions

Definition 1, regarding access patterns

- analytics: calculate over many/all rows, few colums (corresponding DB: OLAP)
- transactions: work with whole row or few rows at a time (corresponding DB: OLTP)

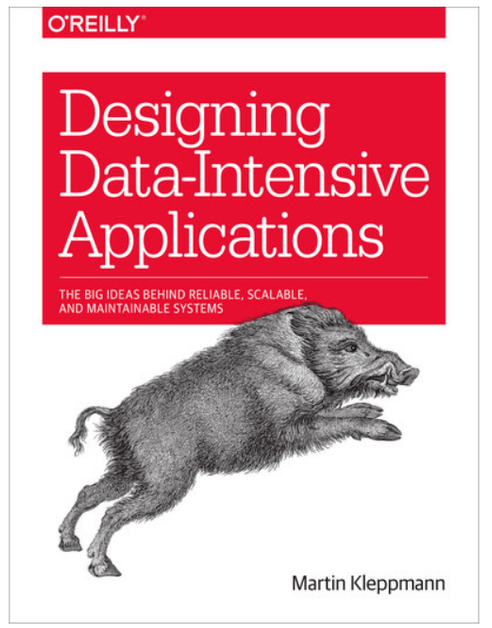
Definition 2, regarding guarantees for a collection of DB operations (often changes).
Common guarantees:

- atomicity: it all happens or nothing happens (partial progress is rolled back upon failure)
- consistency: application invariants (like no negative bank accounts) are supported
- isolation: others cannot see a transaction in progress (aka *atomicity* when talking about locks)
- durability: once finished, it persists (even if machine crashes+restarts)

Transactions in a DB are similar to critical sections in a multi-threaded process:

```
 8    if bank_accounts[src] >= dollars:
 9        bank_accounts[src] -= dollars
10        bank_accounts[dst] += dollars
```

critical section
(example from "locks" lecture)



"The Meaning of ACID"

"NoSQL" databases often have weaker transactions (not ACID) in order to achieve other goals (e.g., performance, scalability, availability, etc).

Outline

Databases

Creating/designing tables

Transactions

Queries

Demos

SQL Query: General Structure

SELECT

FROM

JOIN (optional)

WHERE (optional)

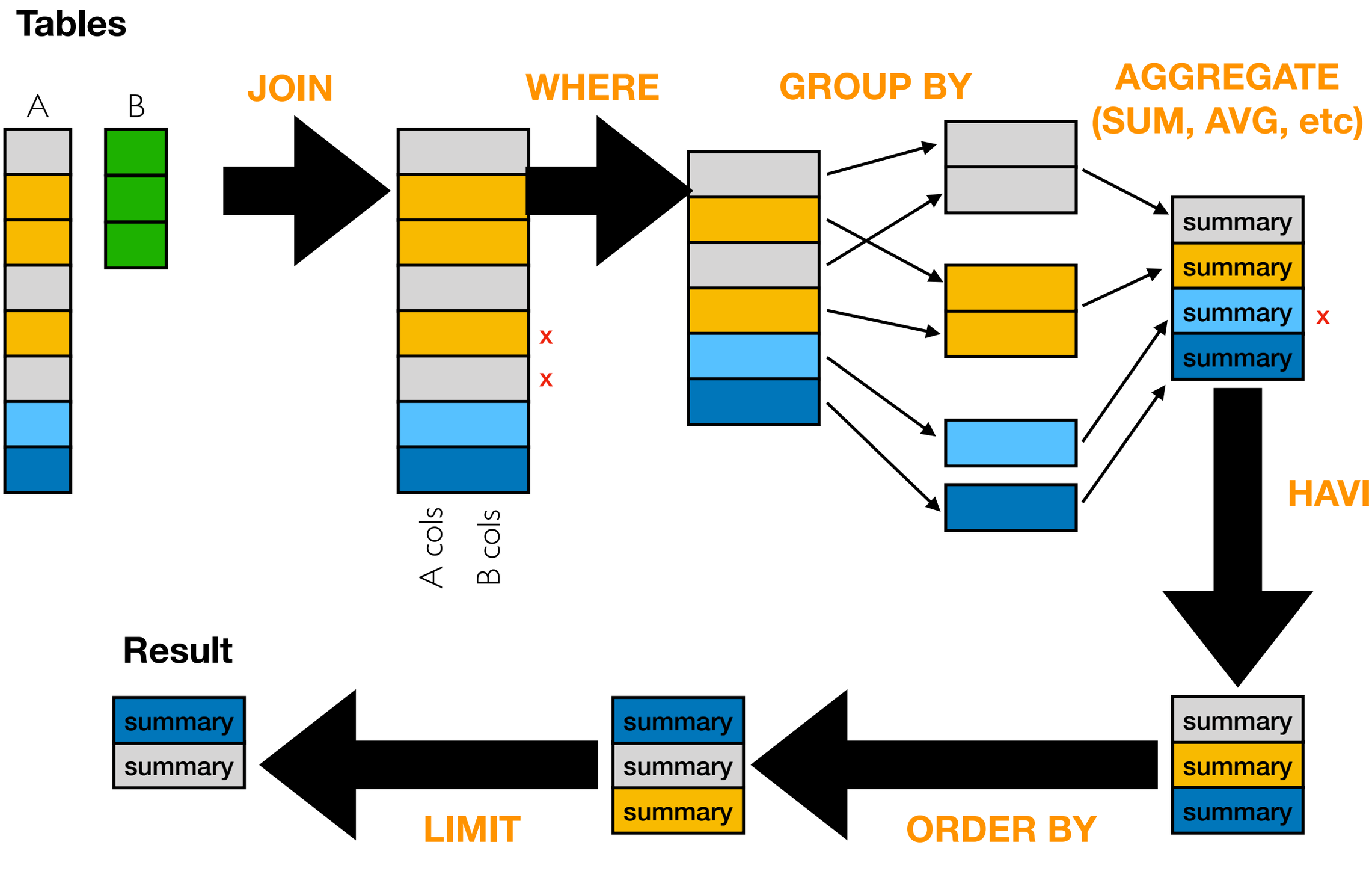
GROUP BY (optional)

HAVING (optional)

ORDER BY (optional)

LIMIT (optional) ;

Query: a series of transformations



Outline

Databases

Creating/designing tables

Transactions

Queries

Demos

Banking Demos