Introduction to Relational Databases

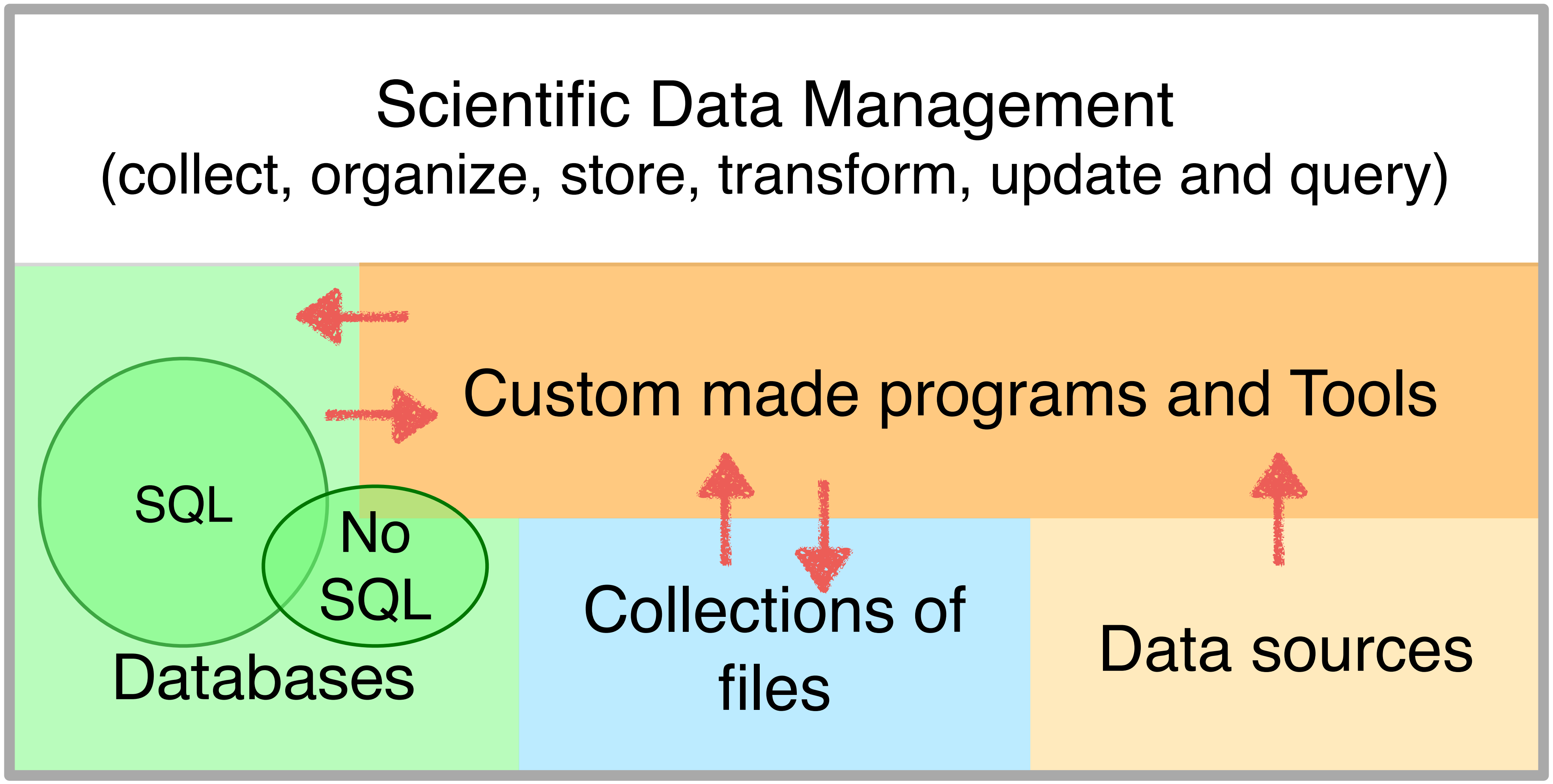
La Serena School for Data Science:
Applied Tools for Data-driven Sciences
August 2019

Mauro San Martín
msmartin@userena.cl
Universidad de La Serena

Contents

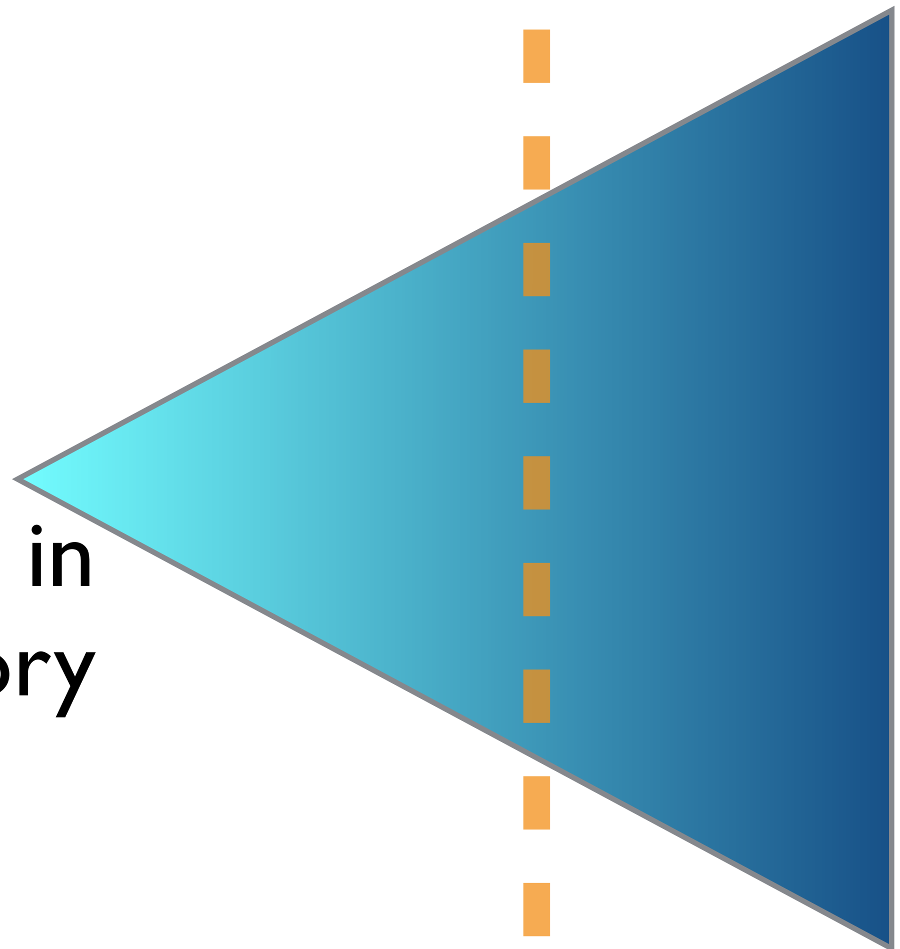
Introduction

Scientific Data Management



A key choice…

- Single user
- Static data
- Dataset fits in main memory



SQL

- Centralized storage (and updates)
- Complete and consistent answers

NoSQL

- Distributed storage (and updates)
- Partial answers and eventual consistency

Collection of Files | Database Systems

Why Databases?

Because is not trivial to satisfy our information-needs under our current computing and storage models and resources.

But…

How an information-need is fulfilled?

Two steps

- Locate.

  We need at least a notion of where each piece of data/information item should be.

- Combine and Select.

  We must combine several pieces of information and choose among these items.

This might be easy and fast (if we have a system) or VERY time consuming (if not).

But…

How an information-need is stated?

When formulating an information need. What would you prefer?

- Elaborate a detailed retrieval plan in terms of the organization of the storage

  e.g. file locations and formats

or

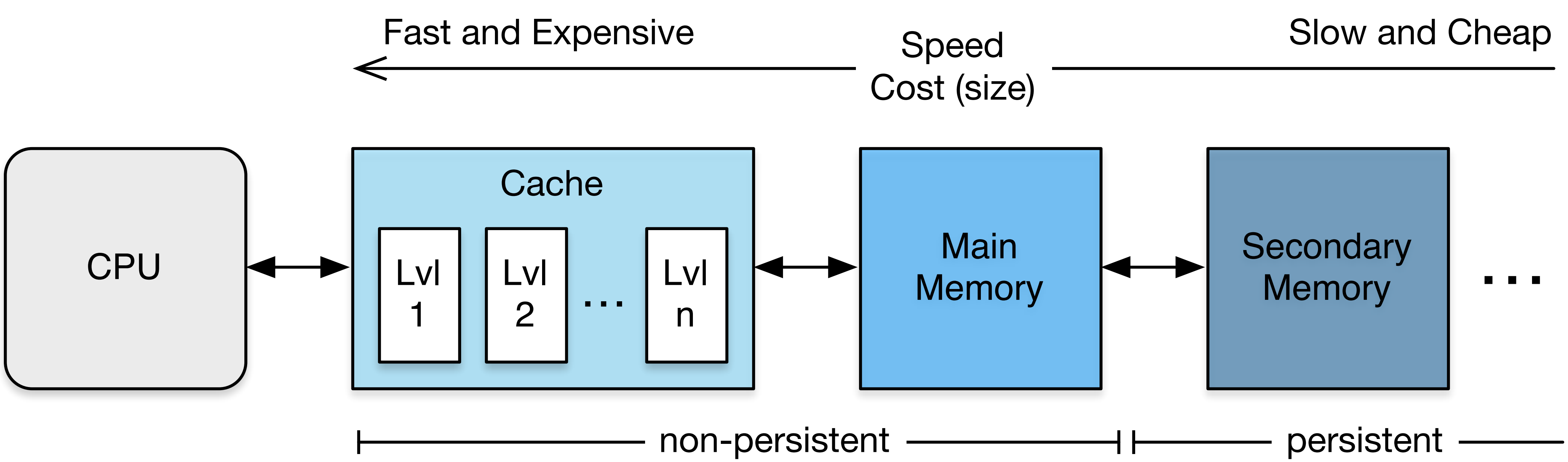
- Express it in terms of the information entities from the problem domain

  e.g. conditions data elements must satisfy to be part of the answer

A relevant technical remark

Not all memory is made equal

The memory in a computer system memory is organized as a hierarchy (this constrains our storage and retrieval models)



Databases

Requirements

- To query and keep updated a shared and meaningful collection of data,
- which is too big to fit in main memory and requires persistence.

Definitions

- Database: An organized and self-describing collection of data, with an intended meaning, and maintained with a purpose.
- Database Management System (DBMS): Software system designed and implemented to define, maintain

There are several types of DBMS

Each one addressing different use cases: types of data and information needs.

- Relational / SQL
- Graph
- NOSQL and NewSQL (column stores, key-value stores, hstore, etc.)

Interesting example:

- Qserv (LSST)

1
Relational Databases Concepts
(an extremely brief introduction)

RDBs at a glance

- E. F. Codd 1970

  "A Relational Model of Data for Large Shared Data Banks"

- Main characteristics
  - One simple data structure: relation (table)
  - Solid mathematical foundations
  - Several comprehensive implementations available

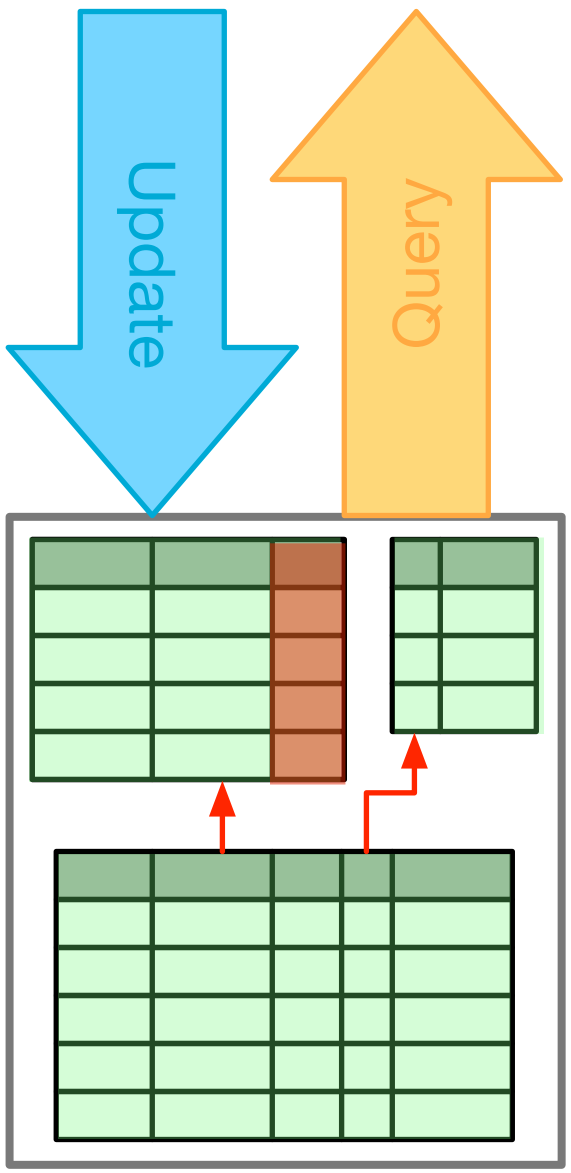
    (PostgreSQL, MySQL, Oracle, SQL Server, etc.)

- Industry standard since the 80's

Relational Data Model

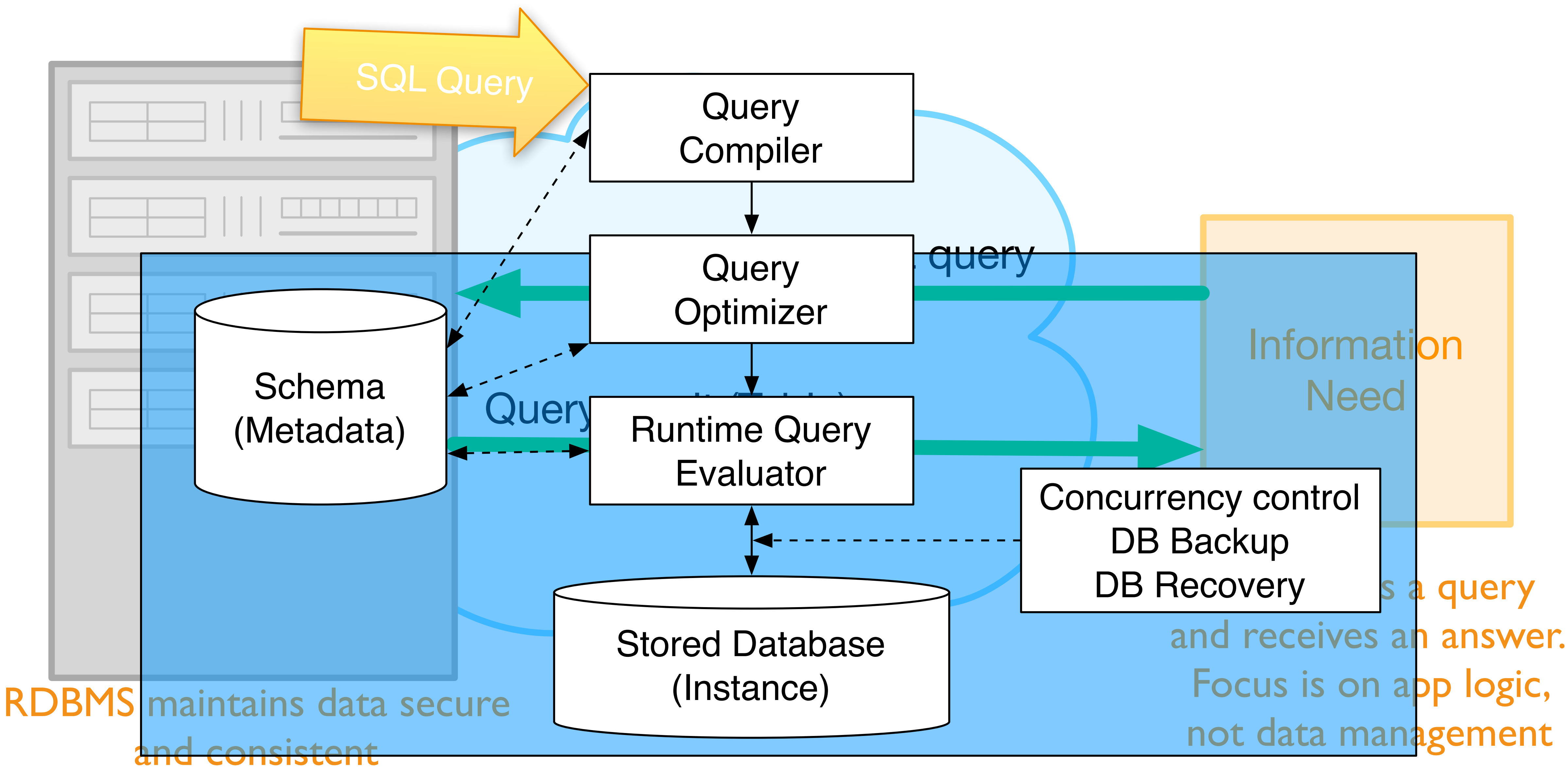
The relational data model

- data structure
  relations/tables: collections of tuples
- operations (update + query)
  Structured Query Language (SQL),
  based on Relational Algebra and Calculus
- integrity constraints
  Data type, not null, referential integrity

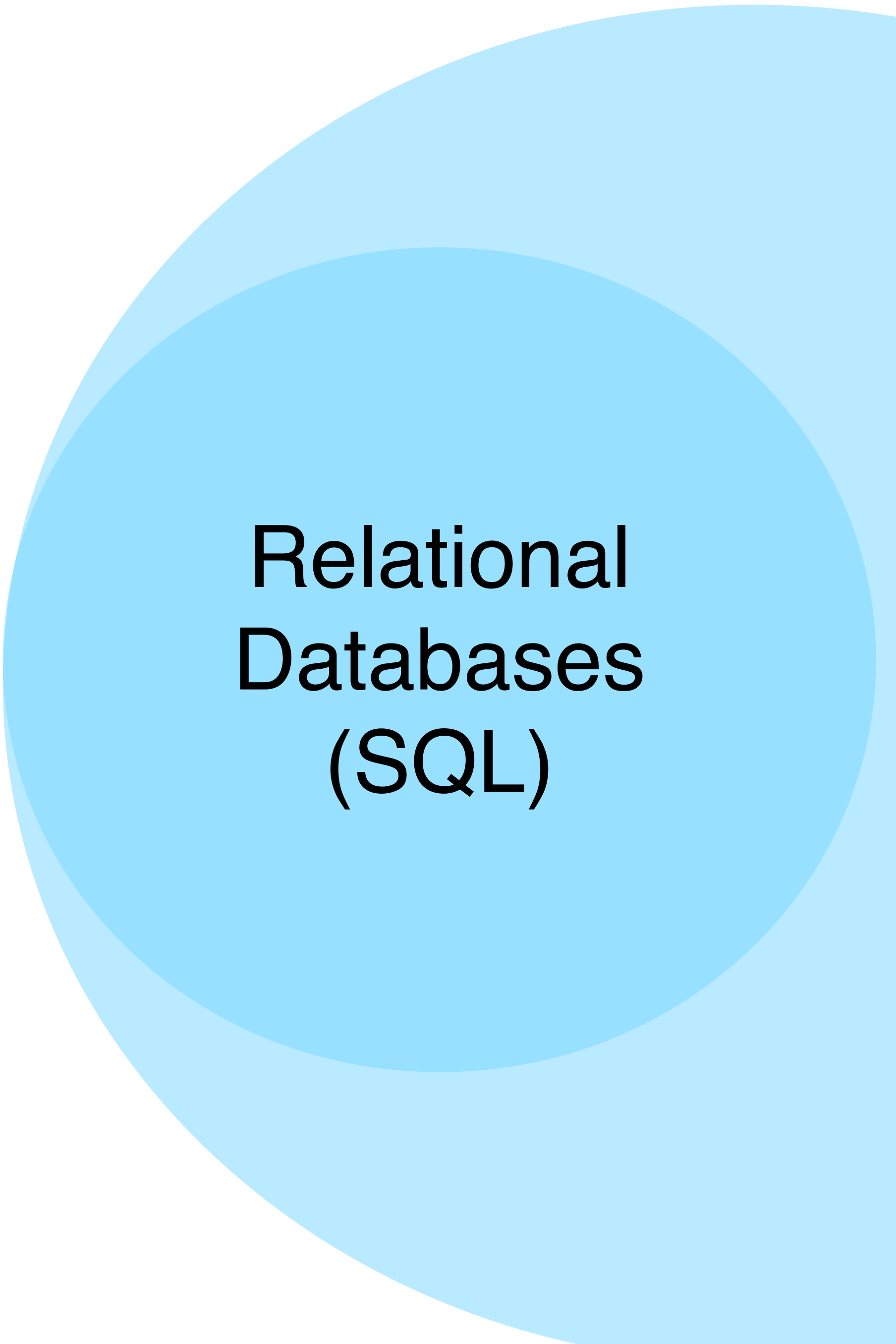


What is an RDBMS?

A Relational DataBase Management System is the software that implements a Relational Database



RDBMS Comfort Zone



An RDBMS performs better when ...

- Data is complete, homogeneous and well defined.
- All data is together (in the same database).
- Answers must be complete and fully consistent.
- Vertical scaling is possible.

RDBMS Objects

- Tables

  Represent data: collection of records
  Record: set of attributes (columns)
  that represents a fact in the real world.

ObjectID	A	B
ID1	3.4	a
ID2	4.0	b
ID2	2.1	c

- Views: named queries
- Indices: improve search and access time
- Functions: extend query language

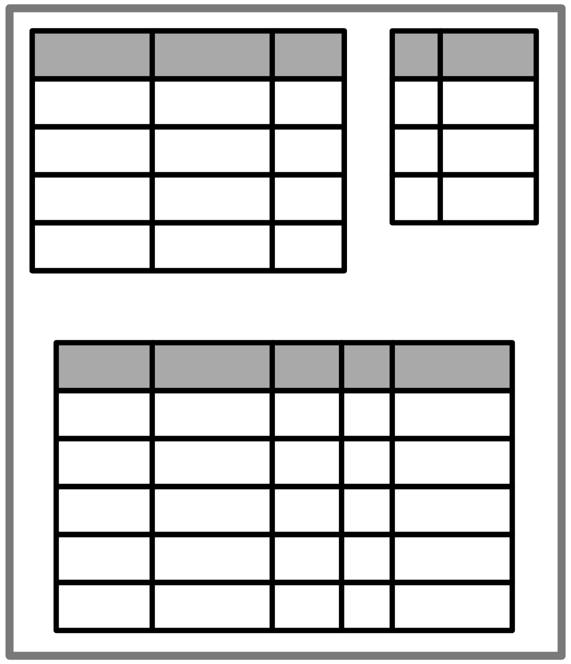
Building a DB

- Design a Schema

  Tables (columns, types, and keys), integrity constraints, and other objects. Avoid data duplication, null values, and update anomalies.

  - SQL as Data Definition Language

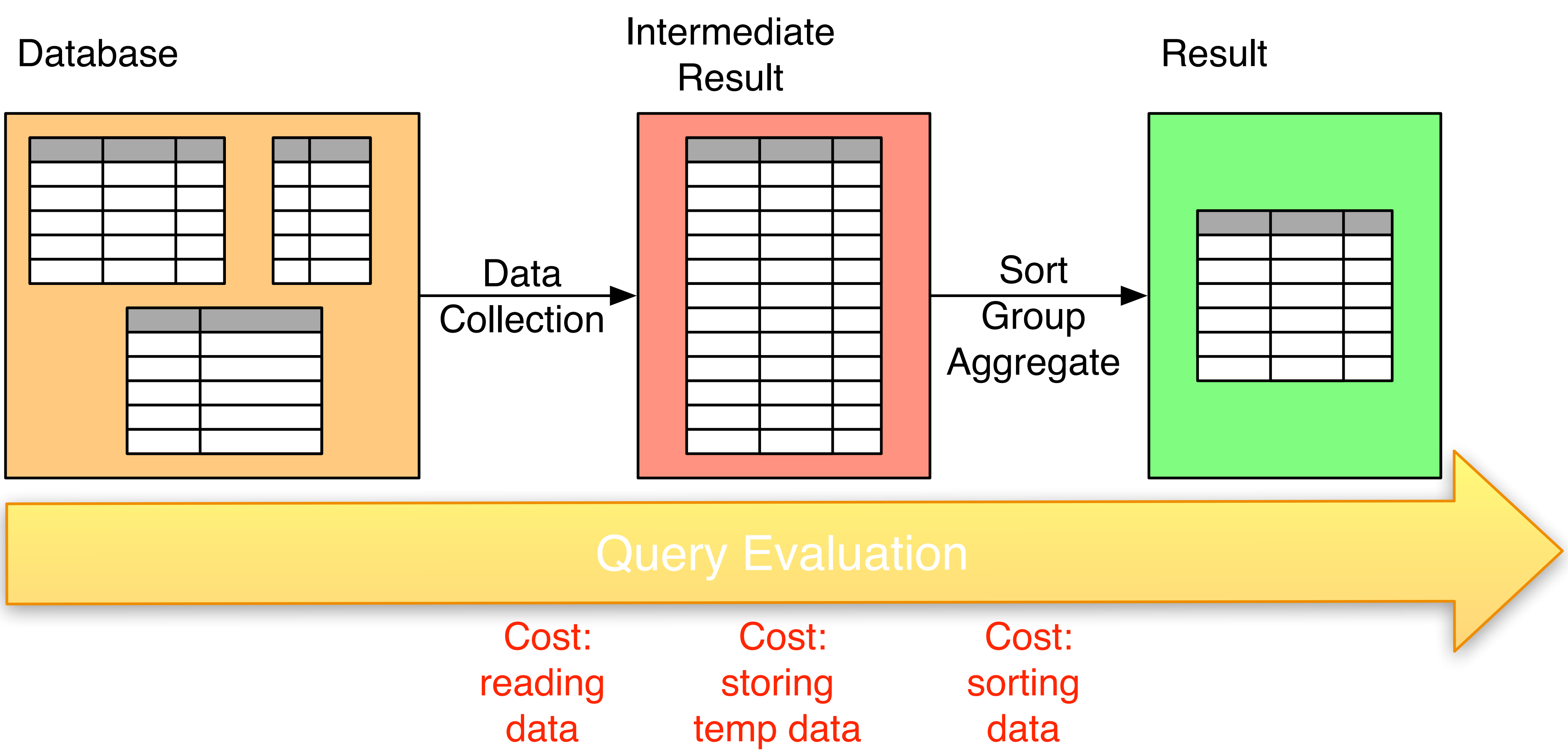
```
create table myTable(number int, letter char)
drop table myTable
```



- Load data into the DB:
  - Bulk loading from SQL dumps, csv files, etc.
  - Insert individual records (SQL)

Querying the DB

Map data from DB to the information needed



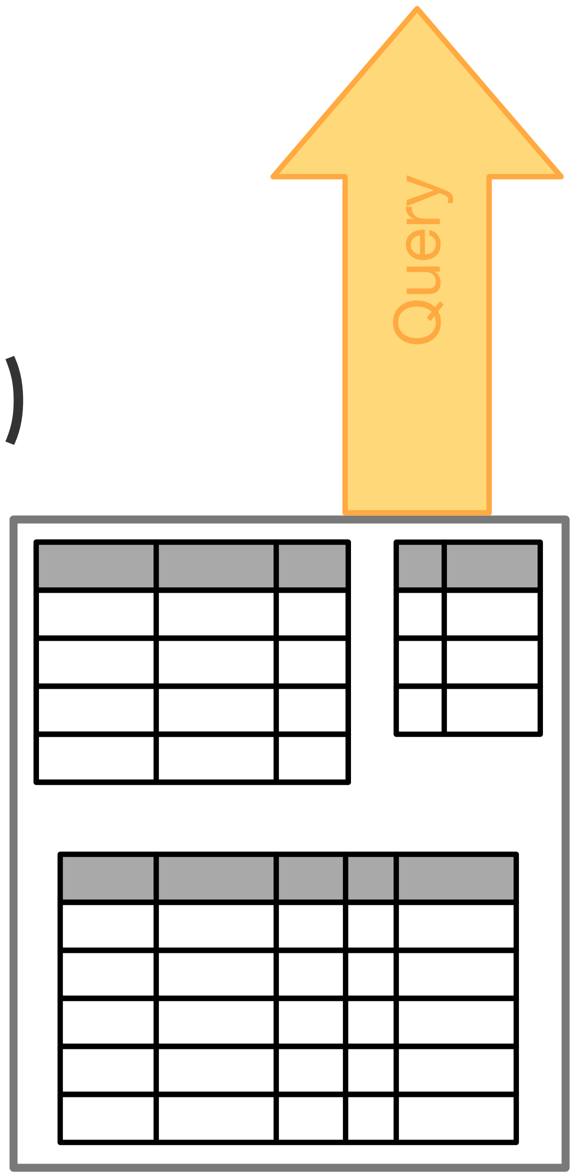
SQL: Querying the DB

- Basic Query Structure

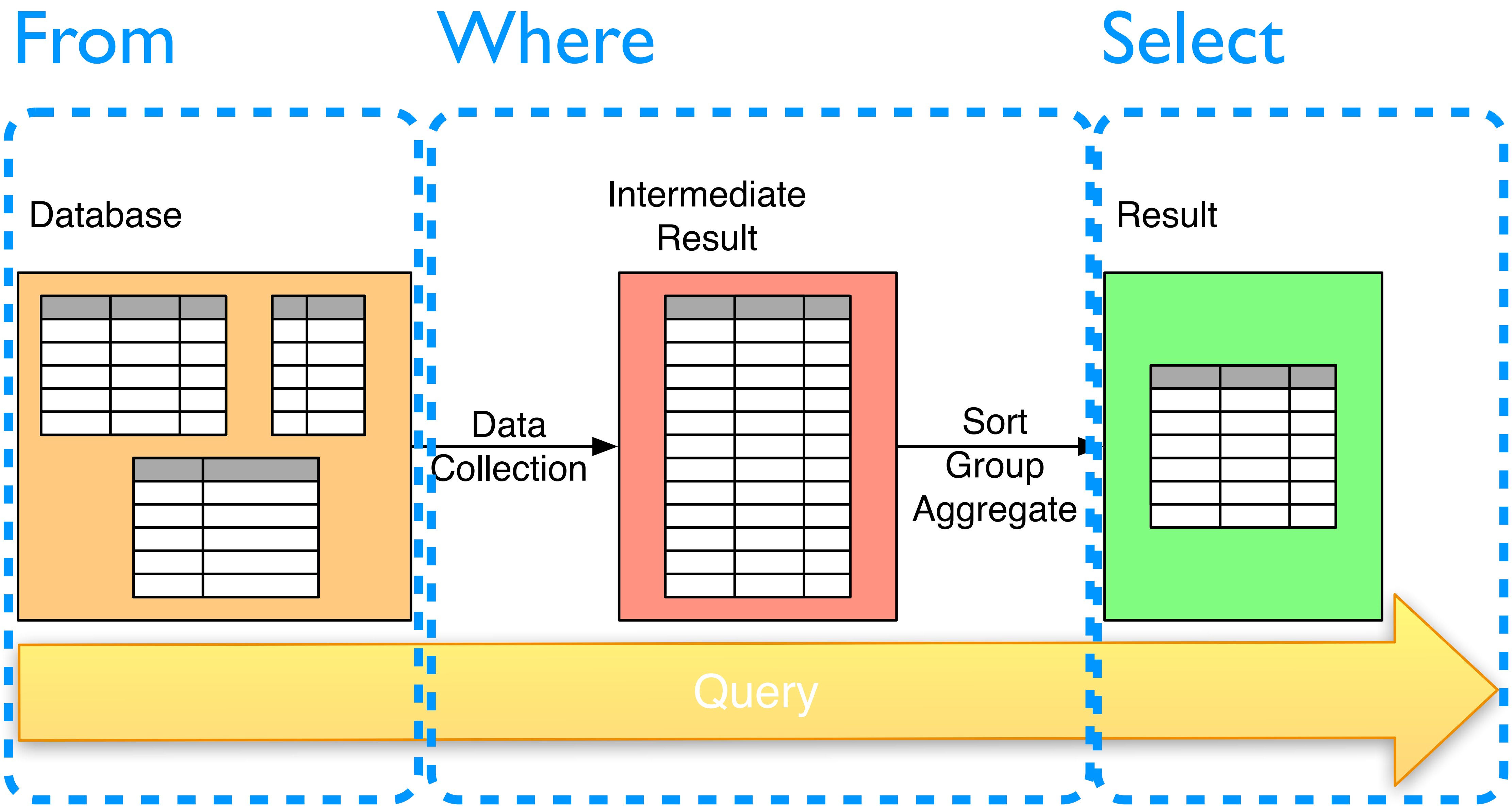
  SELECT: definition of the output table
  FROM: identification of source tables
  WHERE: optional condition (filter or join)

- Additional blocks

  GROUP BY: group defining criteria
  HAVING: optional condition on aggregate values
  ORDER BY: sorting criteria for the result



Query Evaluation



Note that query results are also tables ⇒ query composition

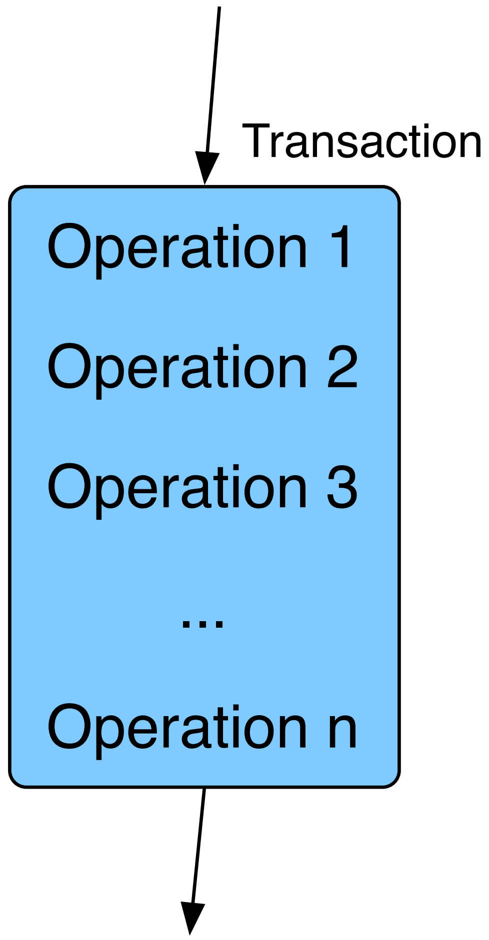
Query Complexity (cost)

- Data Volume
  - I/O based cost model

    number of reads from and writes to persistent storage
- Query Complexity
  - table size: n, number of tables: k
  - projections, and selections (search): O(1) to O(log n) to O(n)
  - joins: O(n) to $O(n^k)$
  - group, and aggregates (sort): O(n log n)

Updates

- Update: add and modify data.
  - Warning: Updates may render the database inconsistent
- Transactions and ACID
  - Atomicity
  - Consistency
  - Isolation
  - Durability



SQL: Updating the DB

- SQL as Data Manipulation Language
  - Inserting new records in tables

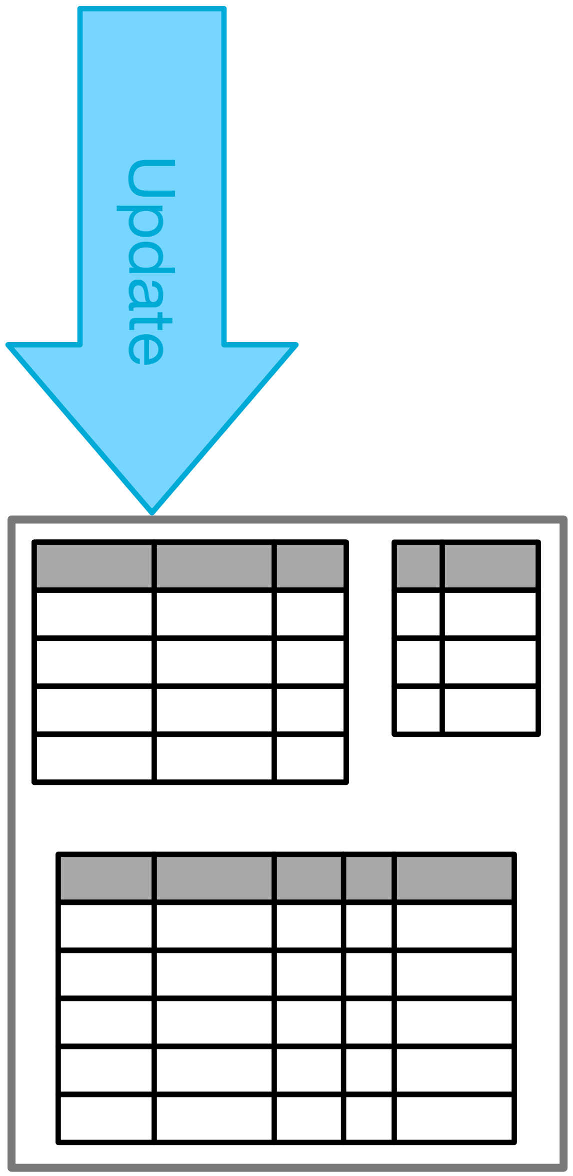
    ```
    insert into myTable values(1, 'a')
    ```

  - Updating data in existing records

    ```
    update myTable set letter = 'b'
           where number = 1
    ```

  - Removing records from tables

    ```
    delete from myTable where number = 1
    ```



Update Complexity (cost)

- Data Volume
  - I/O based cost model

    number of reads from and writes to persistent storage

- Update Complexity
  - table size: n
  - search: O(1) to O(log n) to O(n)
  - integrity constraints must be checked, referential integrity constraints may propagate the task across the database

II
Relational Databases
Practice

Executing Queries

- Parametric
- SQL
  - System console
  - Applications and web interfaces
- From code
  - Parametric from programmer's perspective
  - Languages + libraries

Sloan Web Interface

- Example Database
  - Source:

    Sloan Digital Sky Survey, DR15 (dozens of tables and views, millions of records),

    see their SQL tutorial:

    http://skyserver.sdss.org/dr12/en/help/howto/search/searchhowtohome.aspx
  - Interactive web interface:

    Small answers, exploratory purposes.

    http://skyserver.sdss.org/dr15/en/tools/search/sql.aspx
  - CasJobs: Web interface for batch jobs

    http://skyserver.sdss.org/casjobs/

Query Practice

- Practice Database
  - Data collected in the last hours from two devices:

    BME680: temperature, humidity, barometric pressure and air quality.

    TSL2561: luminosity sensor (broadband, infrared, and illuminance - lux).
  - Example Schema:

    Two tables (one per device) and several thousands of records.

    bme680(time, temperature, voc, humidity, pressure, altitude)

    tsl2561(time, broadband, infrared, lux)
  - You can follow the examples in the notebook provided (Update server IP address!!)

Building a DB (1/2)

- Design the Schema
  - Tables: columns, types and primary keys
  - Good design: avoid data duplication and NULLs
  - Basic design improving strategy: divide offending tables (new groups of columns)
- Implement the schema

```
create table myTable(number int
primary key,letter char)
```

Building a DB (2/2)

- Insert and remove a record from a table:

```
insert into myTable values(1, ‘a’)
delete from myTable where number = 1
```

- Bulk load data into the schema
  - A sequence of insertions is slow, specially if integrity constraints are present.
  - Prefer bulk loading functions like `copy`

Part III.
NoSQL
Not only SQL

Beyond RDBMS

Maybe a RDBMS is not a good match to my problem ...

- RDBMS limitations
  - Cost of ACID
  - Horizontal scaling
- Relaxing DBMS requirements
  - NoSQL
- Direct Access to Data

NoSQL Comfort Zone

NoSQL

- Data is massive, heterogeneous, and distributed.
- Partial and eventually consistent answers are acceptable.
- Data must be always available.
- Horizontal scaling is preferred (or vertical scaling is not practical).

NoSQL Databases

- Aggregate

  Key: identify each aggregate

  Data: heterogeneous collections of attributes as name/value pairs.

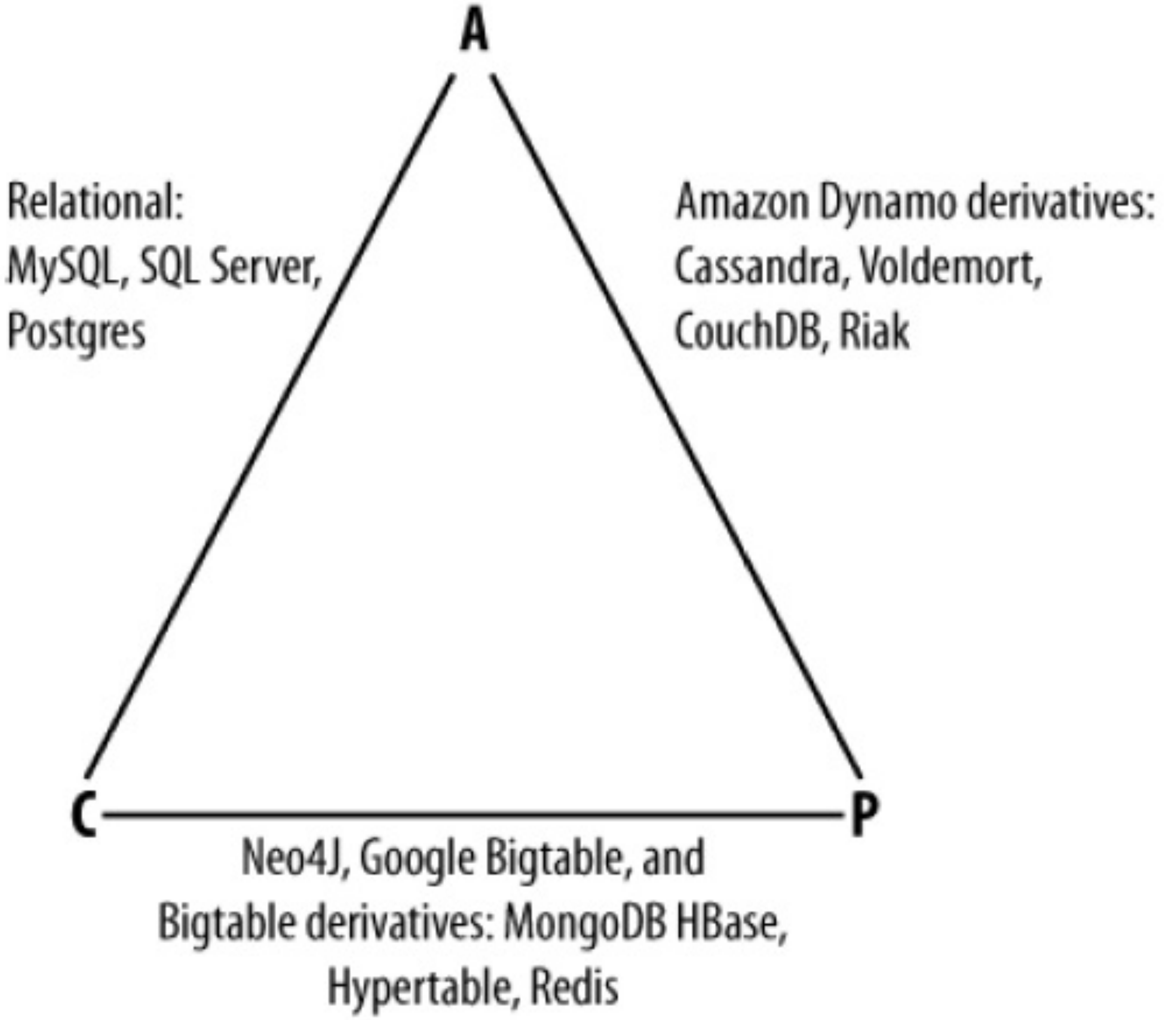
- Main Types
  - Key-Value Stores

    fast to retrieve data with unknown structure
  - Document Databases

    (mostly) tree structured data
  - Column-Family Stores

CAP *Theorem*

- Consistency
- Availability
- Partition Tolerance



Choose two!

Query Evaluation

- Map-Reduce

  Parallel (cluster) data-processing pattern.

- Two steps
  - Map

    Input is an aggregate, output is a bunch of key-value pairs.

    Each map is independent (across aggregates in all the cluster).
  - Reduce

    Map results are collected, sorted and combined.

Summary

- RDBMS
  - Tables: collections of records with keys, and integrity constraints.
  - SQL Queries: basic, join, groups and aggregates.
- An RDBMS is usually better than a collection of files.
- An RDBMS is not always the best solution ¿Management in main memory? ¿NoSQL?