Introduction to Relational Databases

La Serena School for Data Science:
Applied Tools for Data-driven Sciences
August 2017

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

Contents

- Introduction
- Part I. Relational Databases Primer
- Part II. Relational Databases Hands-On

Introduction

Why Databases?

- because we can't live without data

or

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

But…

How an information-need is answered?

Let's try an analogy

Think about packing your bag to come to La Serena

1. locate and select things to carry to LS
2. organize everything inside the bag
3. leave a lot of food for the cat

https://simonscat.com/

1. Locate and select things to carry to LS

- Locate.

  We need at least a notion of where each item should be: toothbrush in the bathroom, shoes in the bedroom, phone charger by the kitchen counter, …

- Select.

  We must choose if there are several items of the same class, v.g.: to decide which shoes?

This might be easy and fast (if we live in a tidy place) or VERY time consuming (if not).

2. Put everything inside the bag

- Organize and pack

  We should follow some practical criteria to put everything inside the bag

https://simonscat.com/

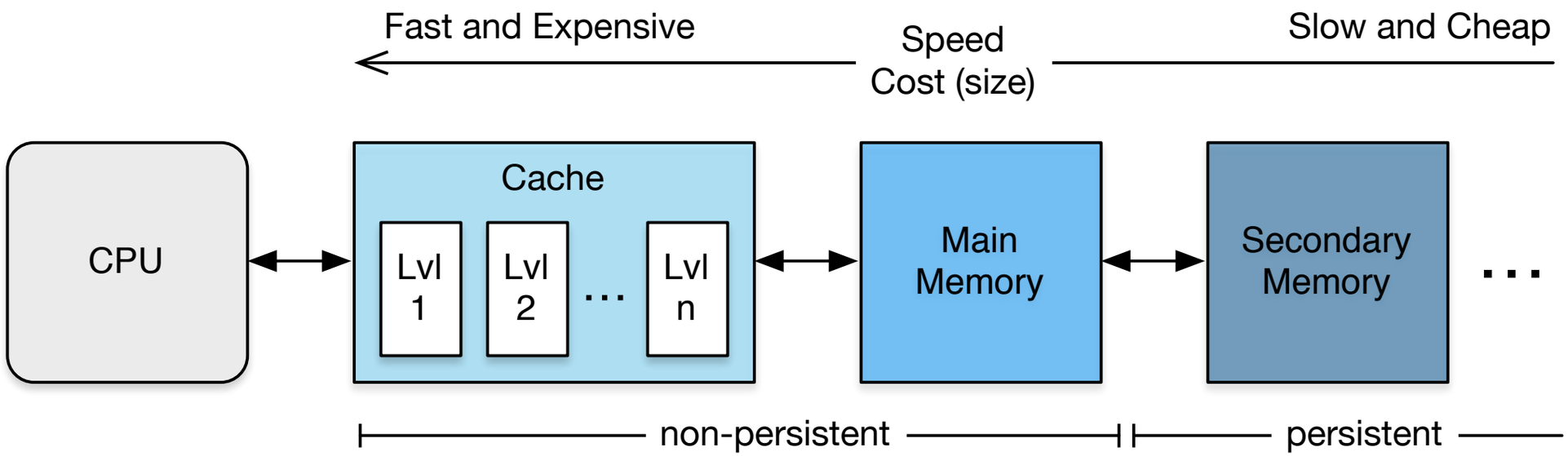
Remember, the actual **data bag** is huge…

https://simonscat.com/

Two technical remarks

1. Database Management Systems are not File Systems
2. Not all memory is made equal

In a computer system memory is a hierarchy (this constrains our storage models)



Databases

Requirements

- To query and keep updated a 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 and share a database.

There are several types of DBMS

Each one addressing different types of data and information needs.

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

Interesting example:

- Qserv (LSST)

I

Relational Databases Primer

(actually 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

Capturing the world (or the universe)

The relational data model

- data structure

  relations/tables: collections of tuples

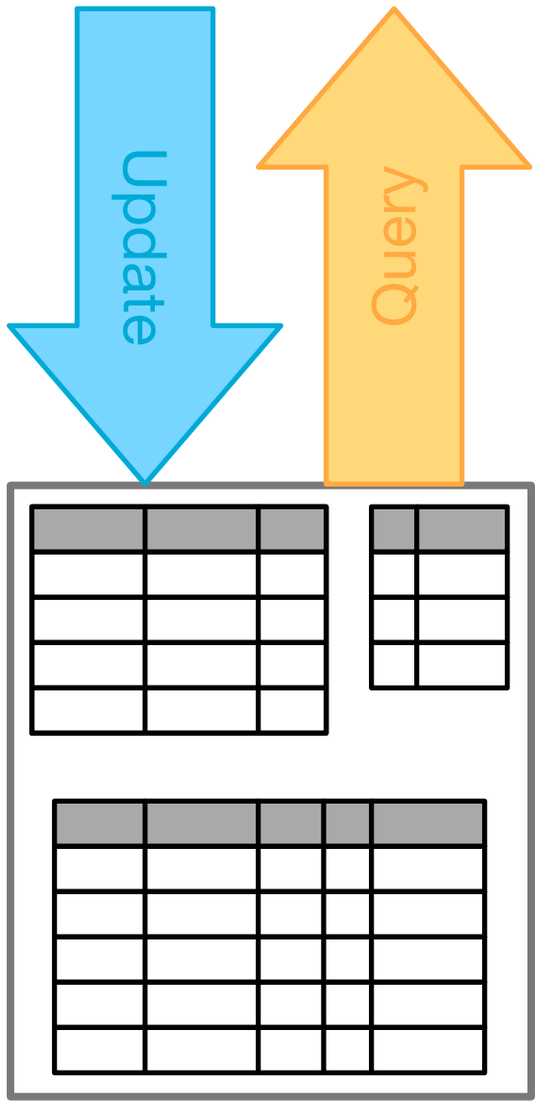
- operations (update + query)

  Structured Query Language (SQL),

  based on Relational Algebra and Calculus

- integrity constraints

  Data type, not null, referential integrity

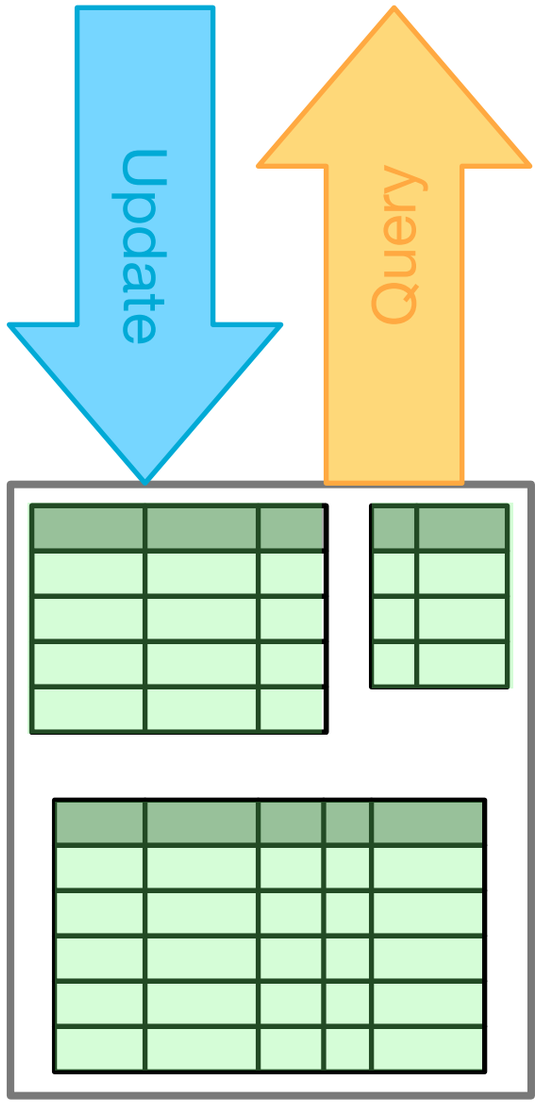


Relational Data Model

Capturing the world (or the universe)

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

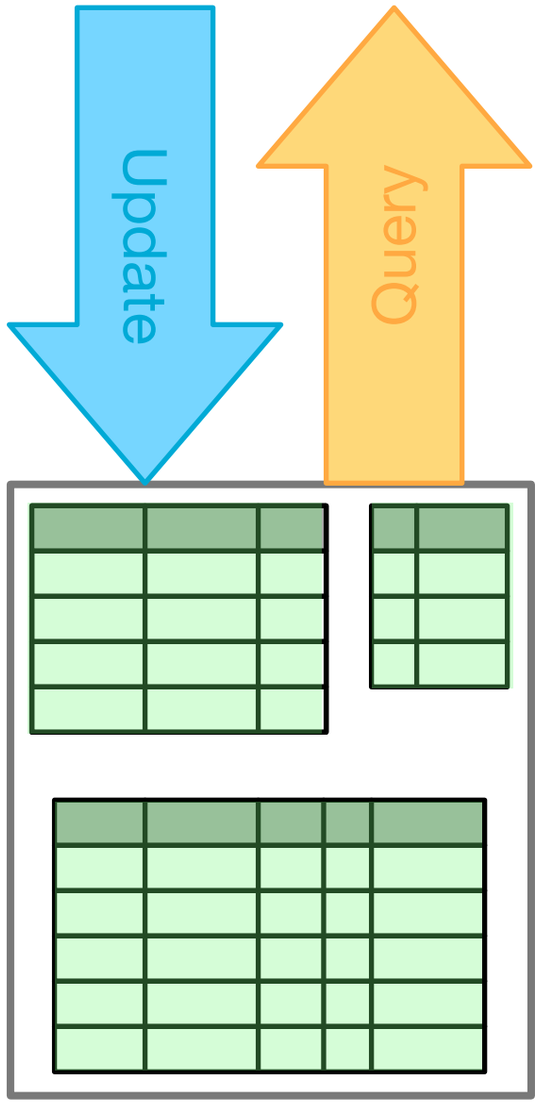


Relational Data Model

Capturing the world (or the universe)

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

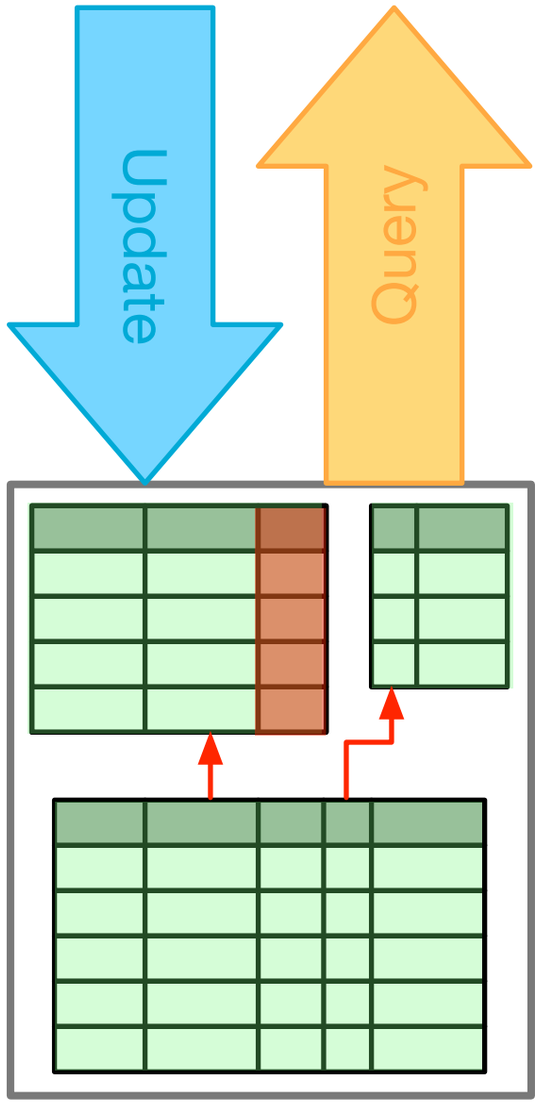


Relational Data Model

Capturing the world (or the universe)

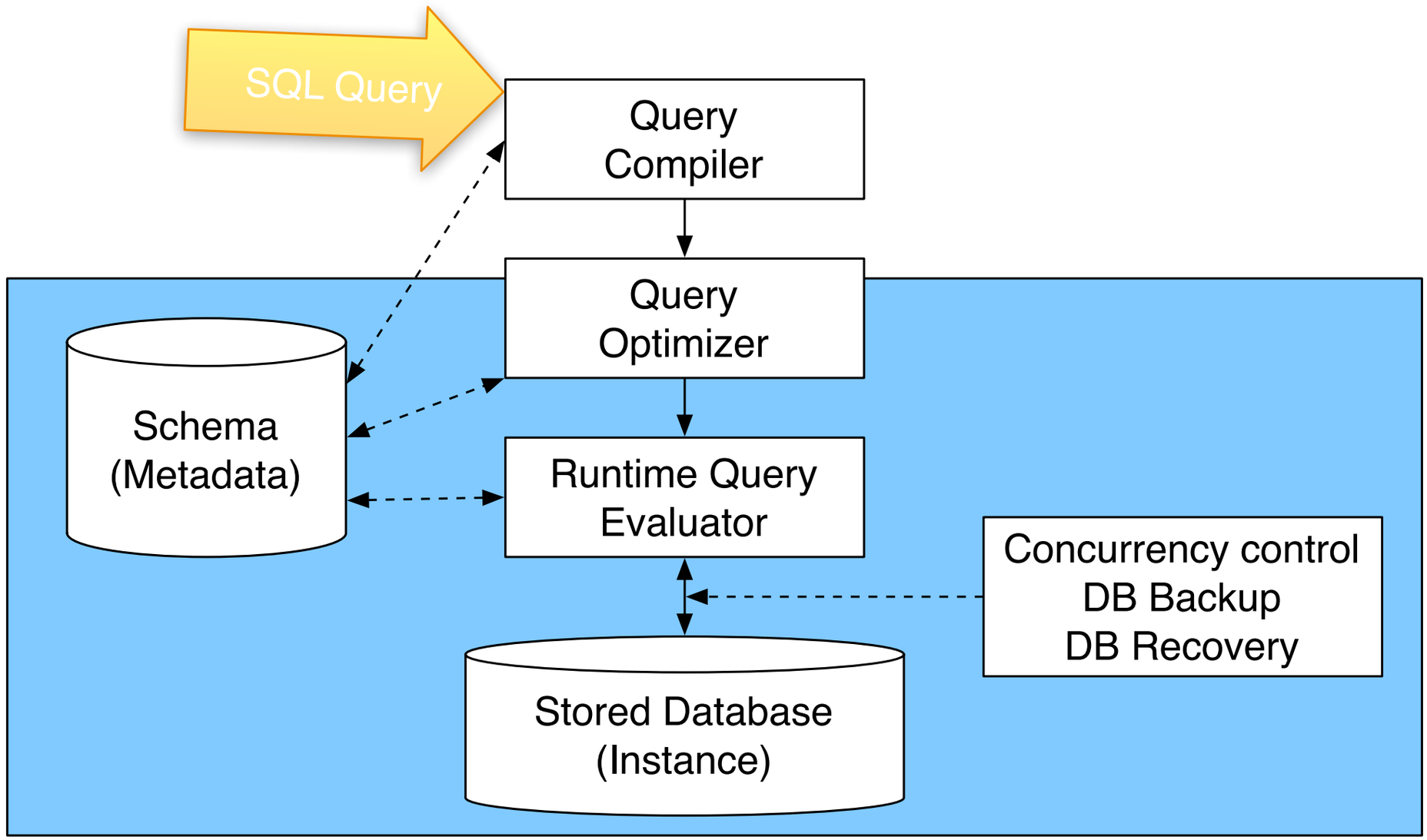
The relational data model

- 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



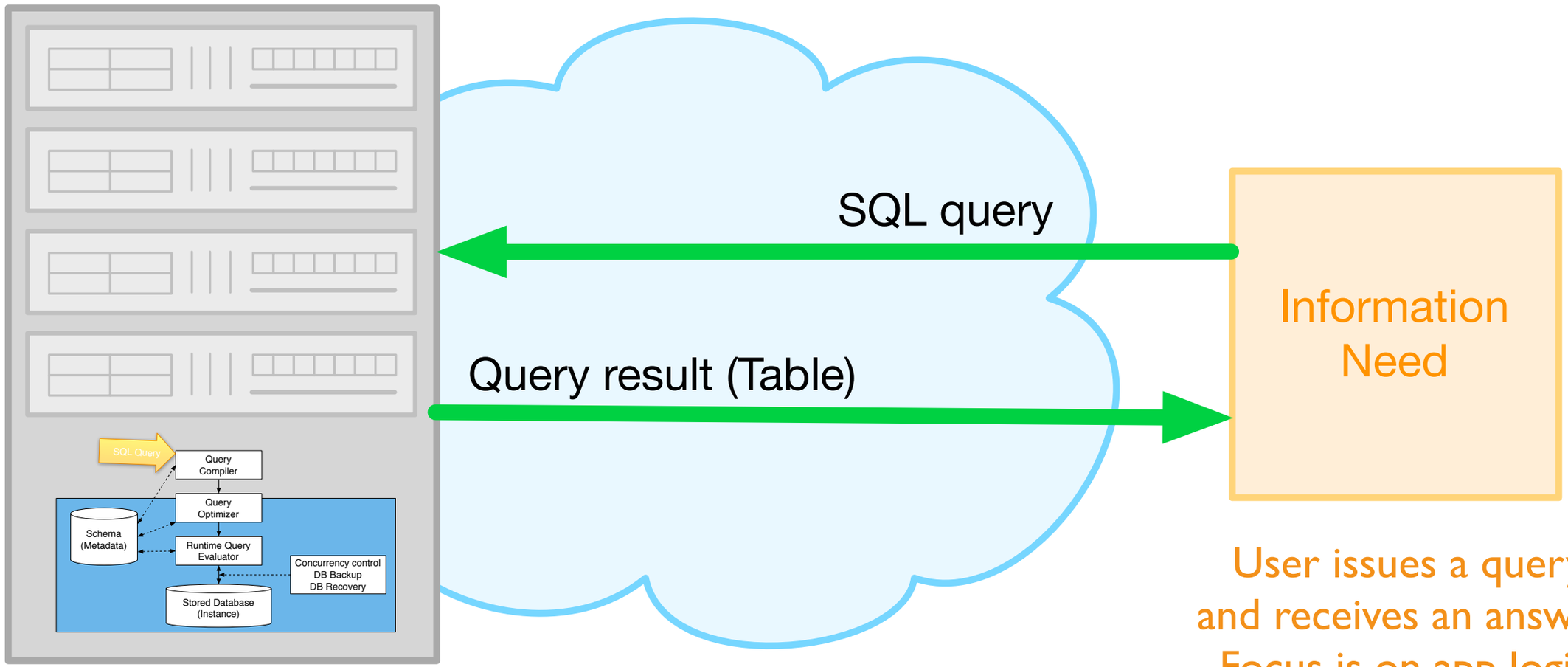
What is an RDBMS?

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



What is an RDBMS?

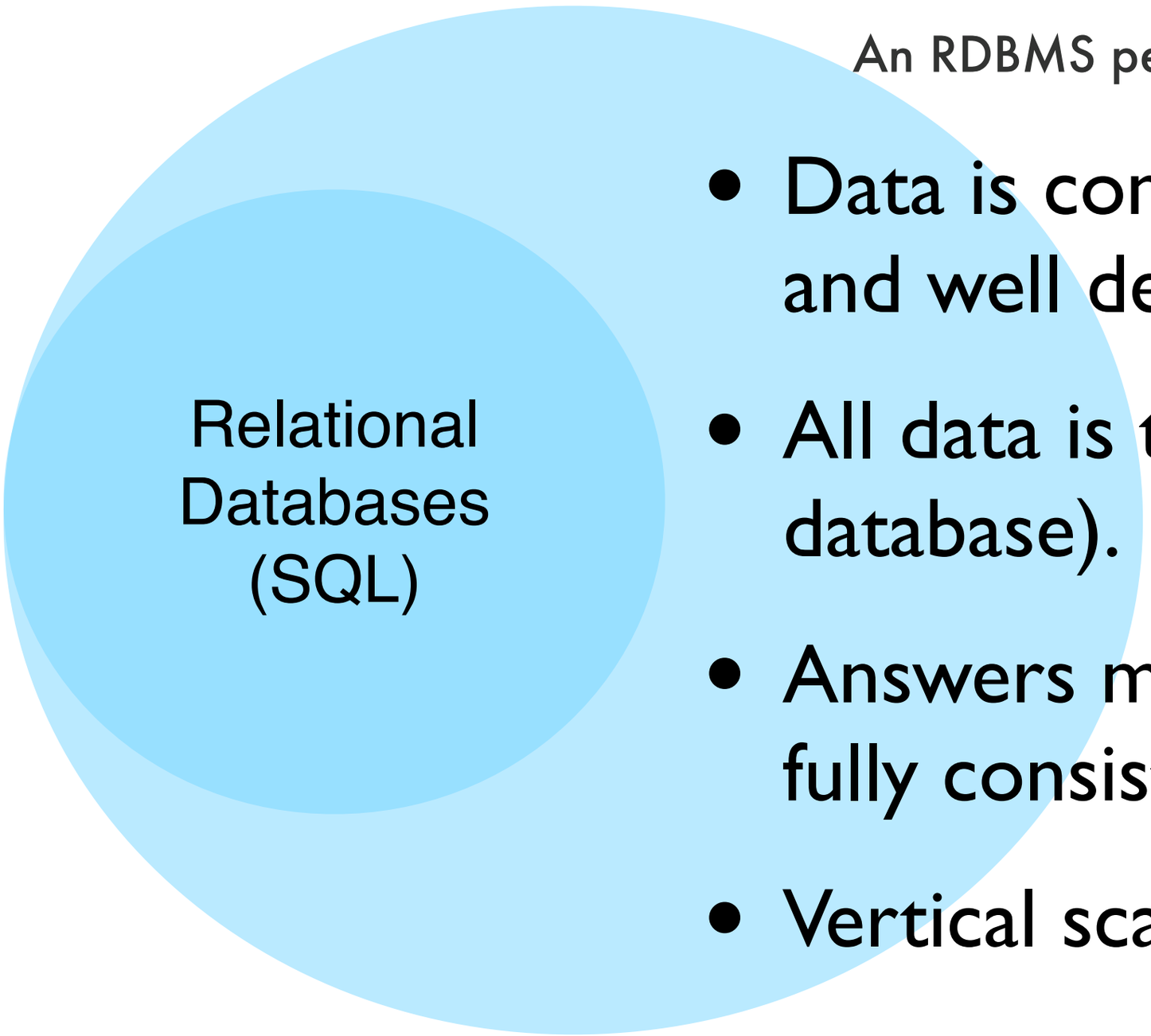
A Relational DataBase Management System is the software that implements a Relational Database



RDBMS maintains data secure and consistent

User issues a query and receives an answer. Focus is on app logic, not data management

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)

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

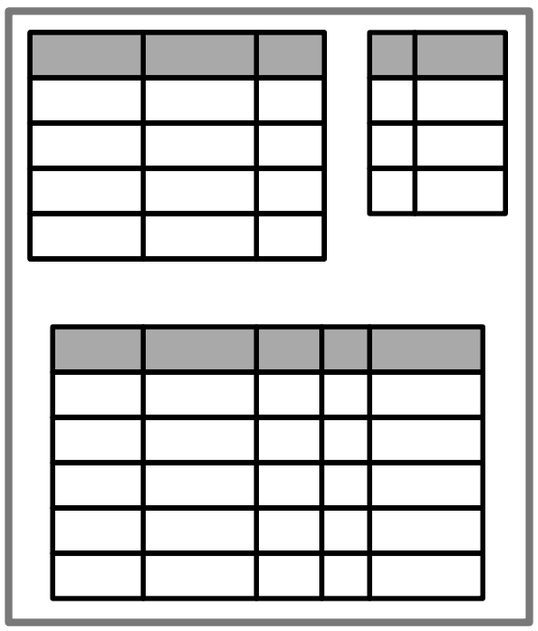
- Design a Schema

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

  - SQL as Data Definition Language

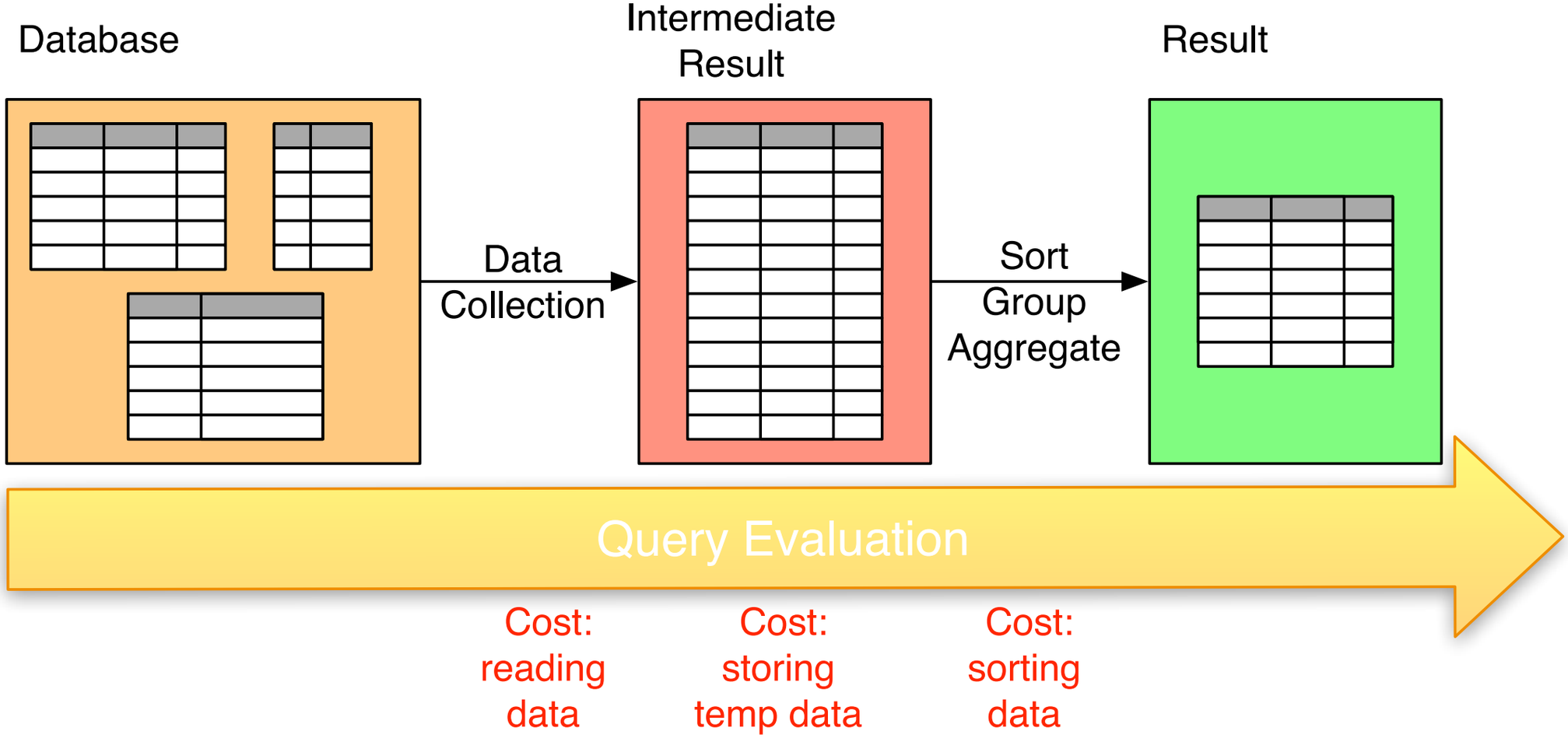
```
create table myTable(number int, letter char)
drop table myTable
```

- 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



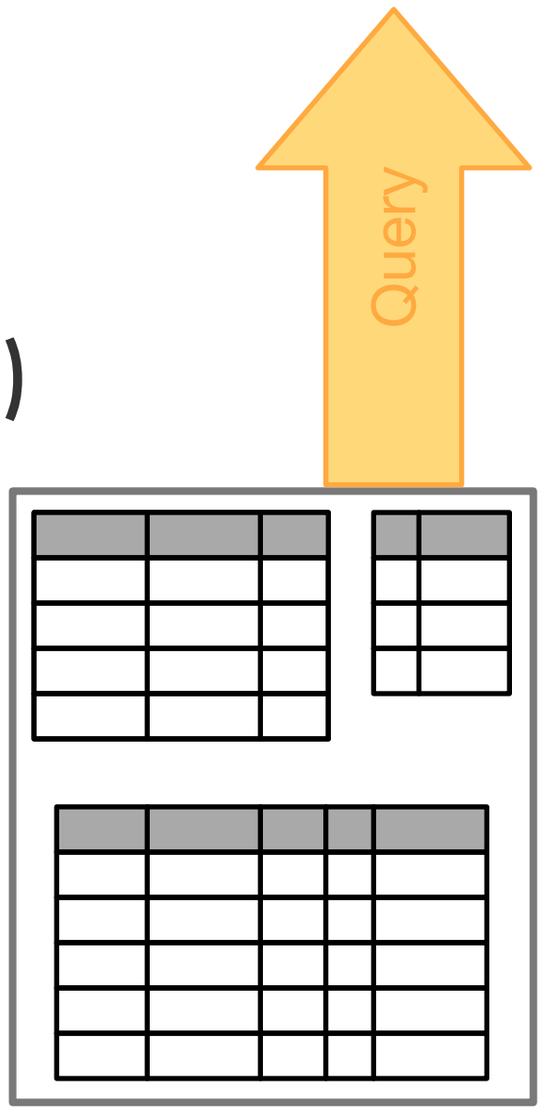
SQL: Querying the DB

- Basic Query Structure

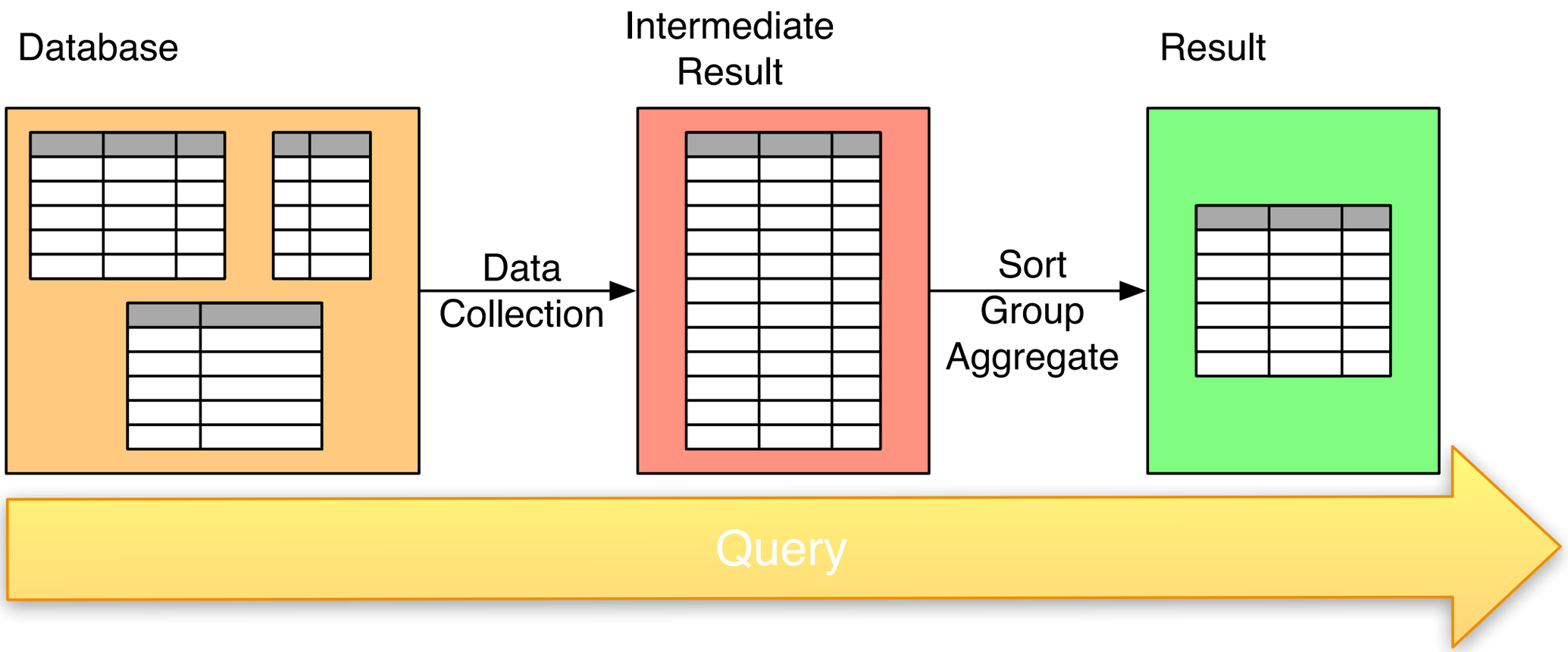
  SELECT: definition of the output table
  FROM: identification of source tables
  WHERE: optional condition (filter or join)

- Additional blocks

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

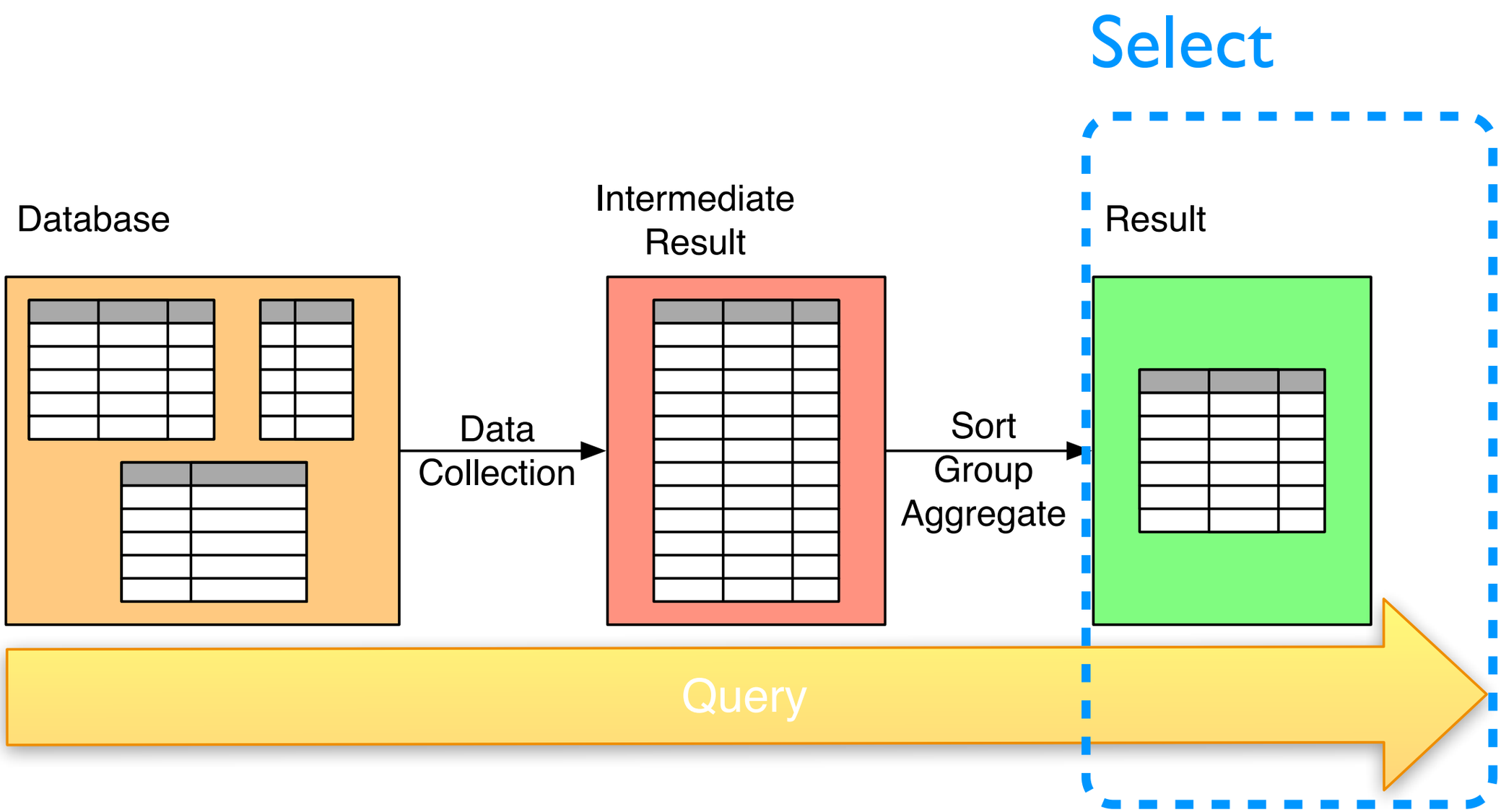


Query Evaluation



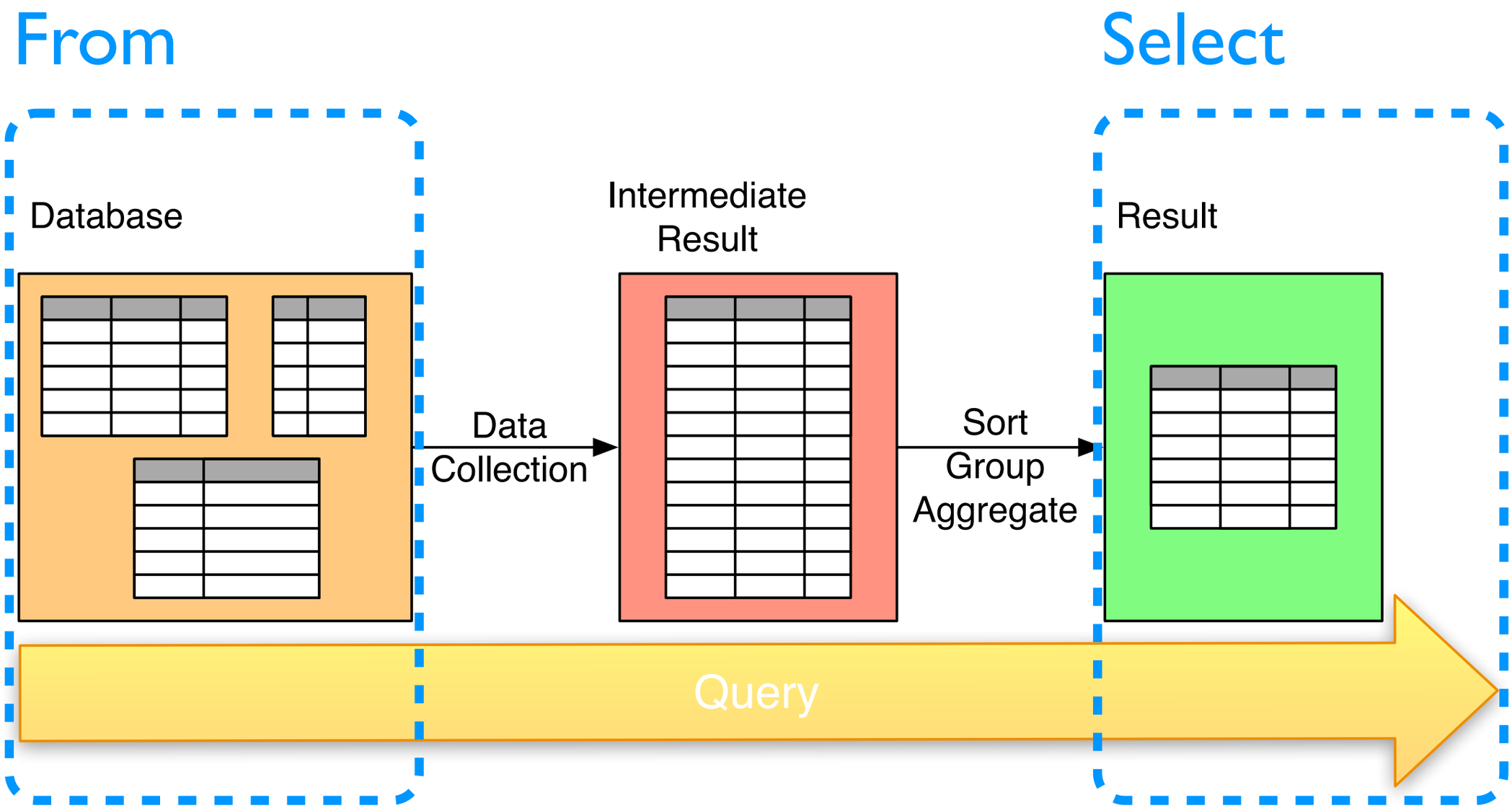
Note that query results are also tables ⇒ query composition

Query Evaluation



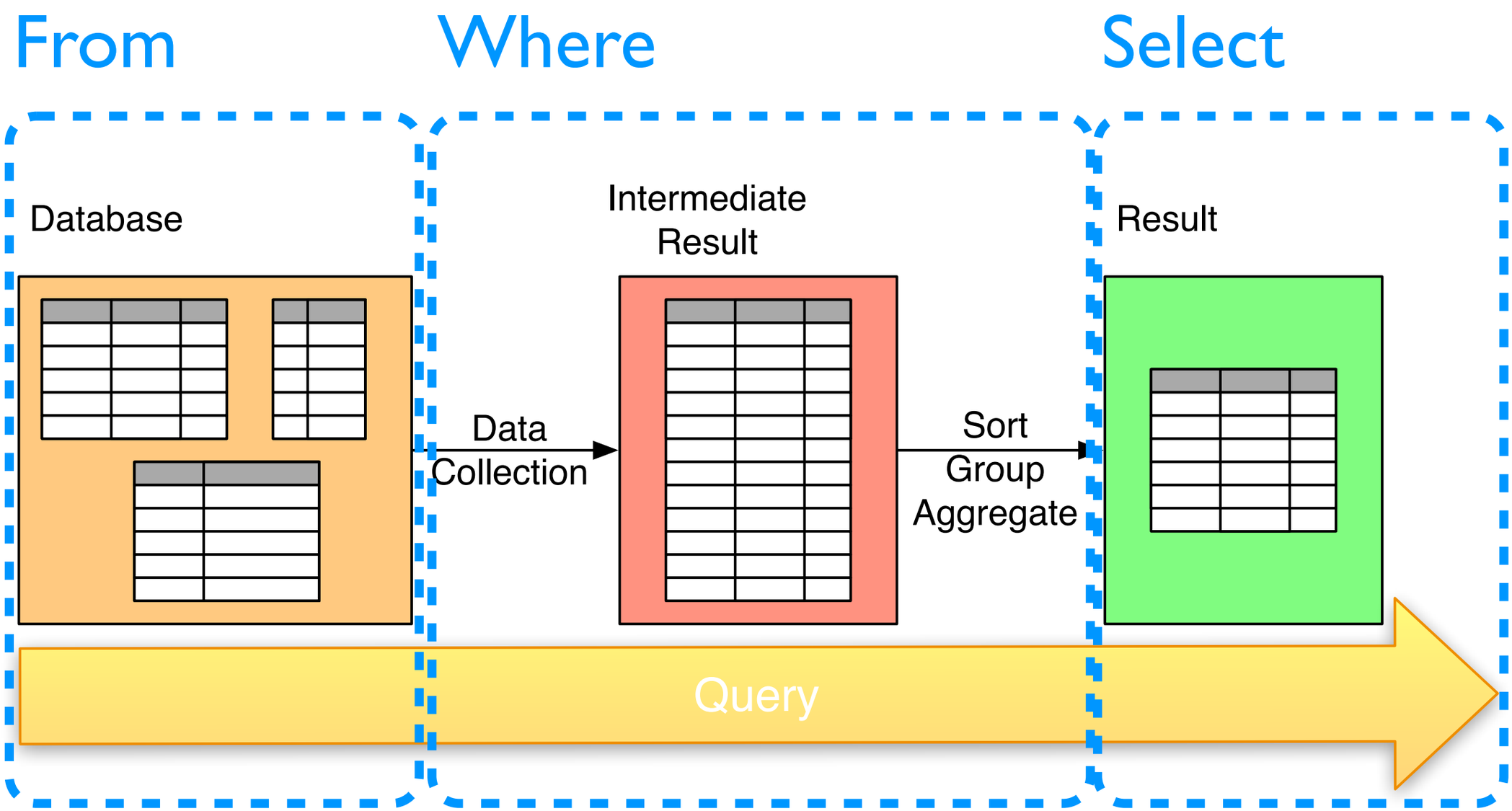
Note that query results are also tables ⇒ query composition

Query Evaluation



Note that query results are also tables ⇒ query composition

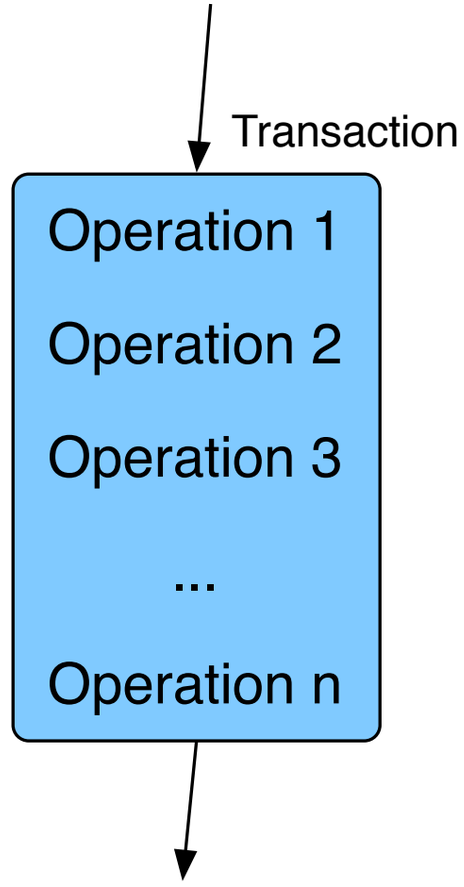
Query Evaluation



Note that query results are also tables ⇒ query composition

Updates

- Update: add and modify data.
  - 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

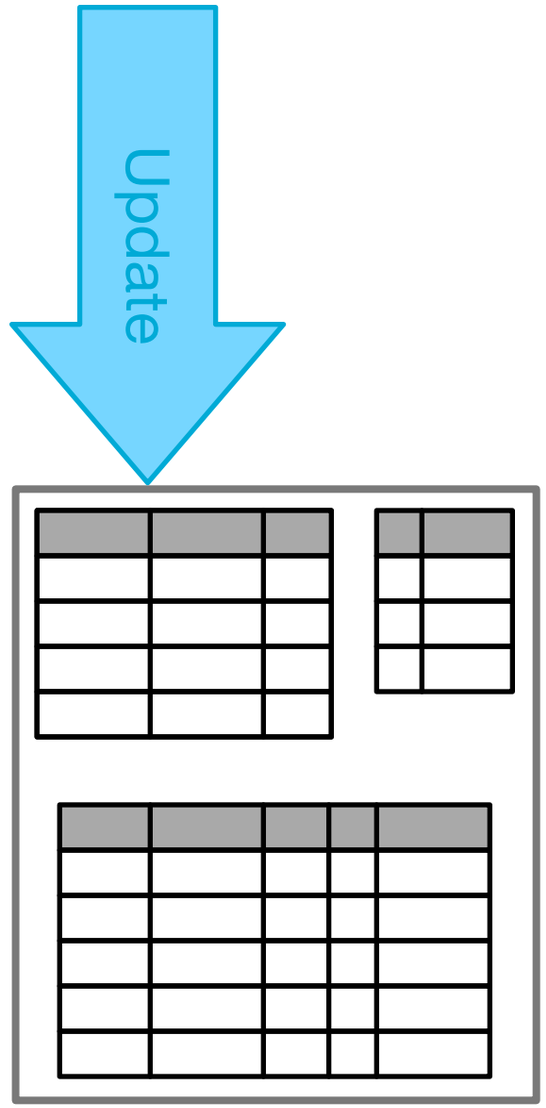
```
insert into myTable values(1, 'a')
```

  - Updating data in existing records

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

  - Removing records from tables

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



II

Relational Databases Practice

Query Examples

- Example Database
  - Source:
    SLOAN DR12 (hundreds of tables and views, millions of records), see their SQL tutorial:
    http://skyserver.sdss.org/dr12/en/help/howto/search/searchhowtohome.aspx
  - Example Schema:
    Tiny subset: 2 tables and dozens of records.

```
photoObj(oid, ra, dec, g, r)
specObj(oid, class, subclass)
```

  - You can follow the examples in the IPython notebook provided (Update de IP address!!)

Basic Queries

```
SELECT * FROM photoObj;
```

```
SELECT oid, class
FROM specObj
WHERE class = 'GALAXY';
```

Basic Queries

```
SELECT * FROM photoObj;
```

```
SELECT oid, class
FROM specObj
WHERE class = 'GALAXY';
```

Basic Queries

```
SELECT * FROM photoObj;
```

Bring me everything from this table!

```
SELECT oid, class
FROM specObj
WHERE class = 'GALAXY';
```

Basic Queries

```
SELECT * FROM photoObj;
```

Bring me everything from this table!

```
SELECT oid, class
FROM specObj
WHERE class = 'GALAXY';
```

Basic Queries

```
SELECT * FROM photoObj;
```

Bring me everything from this table!

```
SELECT oid, class
FROM specObj
WHERE class = 'GALAXY';
```

Bring me the oid of galaxies

Complex Conditions

```
SELECT oid, ra, dec
FROM photoObj
WHERE
      g < 12 and
      r < 12 and
      g - r < 0
```

Complex Conditions

```
SELECT oid, ra, dec
FROM photoObj
WHERE
      g < 12 and
      r < 12 and
      g - r < 0
```

Complex Conditions

```
SELECT oid, ra, dec
FROM photoObj
WHERE
    g < 12 and
    r < 12 and
    g - r < 0
```

What does this condition mean?

Joins

```
SELECT p.oid, p.ra, p.dec, s.subclass
FROM photoObj as p, specObj as s
WHERE
      p.oid = s.oid
      and p.g < 12 and p.r < 12
      and p.g - p.r < 0
      and s.class = 'GALAXY';
```

Joins

```
SELECT p.oid, p.ra, p.dec, s.subclass
FROM photoObj as p, specObj as s
WHERE
      p.oid = s.oid
      and p.g < 12 and p.r < 12
      and p.g - p.r < 0
      and s.class = 'GALAXY';
```

Joins

```
SELECT p.oid, p.ra, p.dec, s.subclass
FROM photoObj as p, specObj as s
WHERE
      p.oid = s.oid
      and p.g < 12 and p.r < 12
      and p.g - p.r < 0
      and s.class = 'GALAXY';
```

The records with the same oid are joined.

Groups and Aggregates

```
SELECT s.subclass, count(*)
FROM photoObj as p, specObj as s
WHERE
      p.oid = s.oid and p.g < 12
      and p.r < 12 and p.g - p.r < 0
      and s.class = 'GALAXY'
GROUP BY s.subclass;
```

Groups and Aggregates

```
SELECT s.subclass, count(*)
FROM photoObj as p, specObj as s
WHERE
      p.oid = s.oid and p.g < 12
      and p.r < 12 and p.g - p.r < 0
      and s.class = 'GALAXY'
GROUP BY s.subclass;
```

Groups and Aggregates

```
SELECT s.subclass, count(*)
FROM photoObj as p, specObj as s
WHERE
      p.oid = s.oid and p.g < 12
      and p.r < 12 and p.g - p.r < 0
      and s.class = 'GALAXY'
GROUP BY s.subclass;
```

Count how many elements are in each subclass

Summary

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