Data Wrangling and Data Analysis

Indexing + Data Integration

Hakim Qahtan

Department of Information and Computing Sciences
Utrecht University

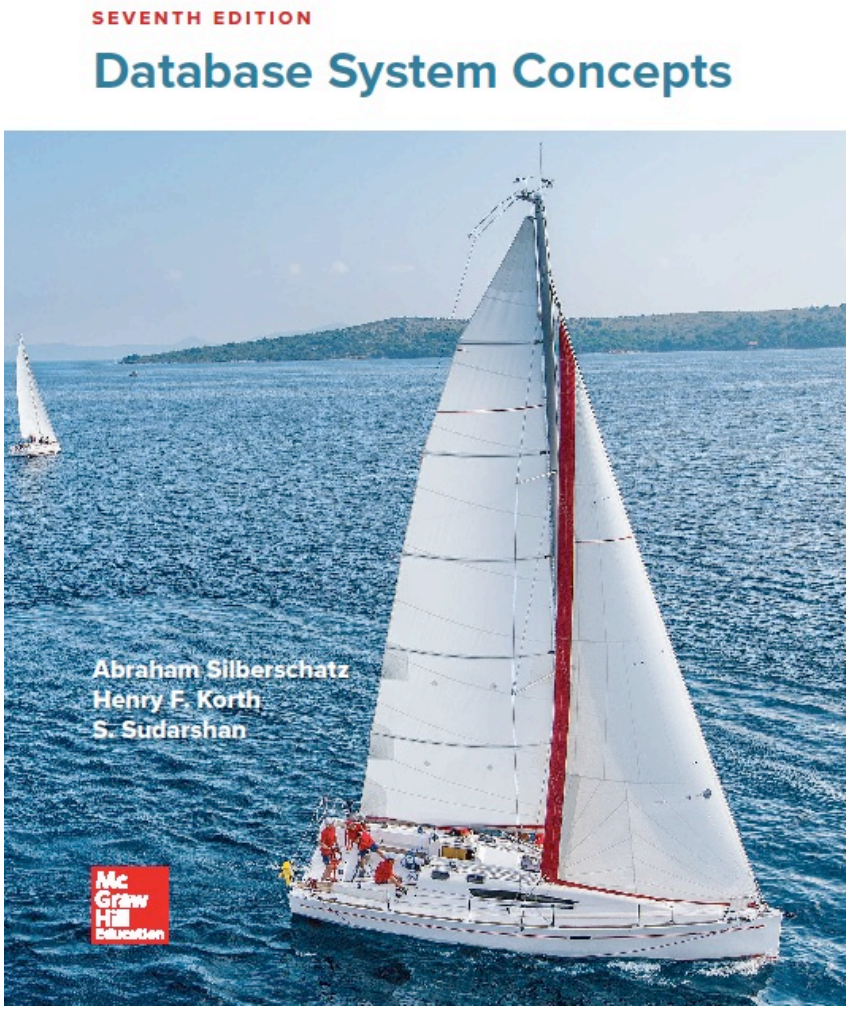
Utrecht University

Reading Material for Today

Database System Concepts (7th Edition)

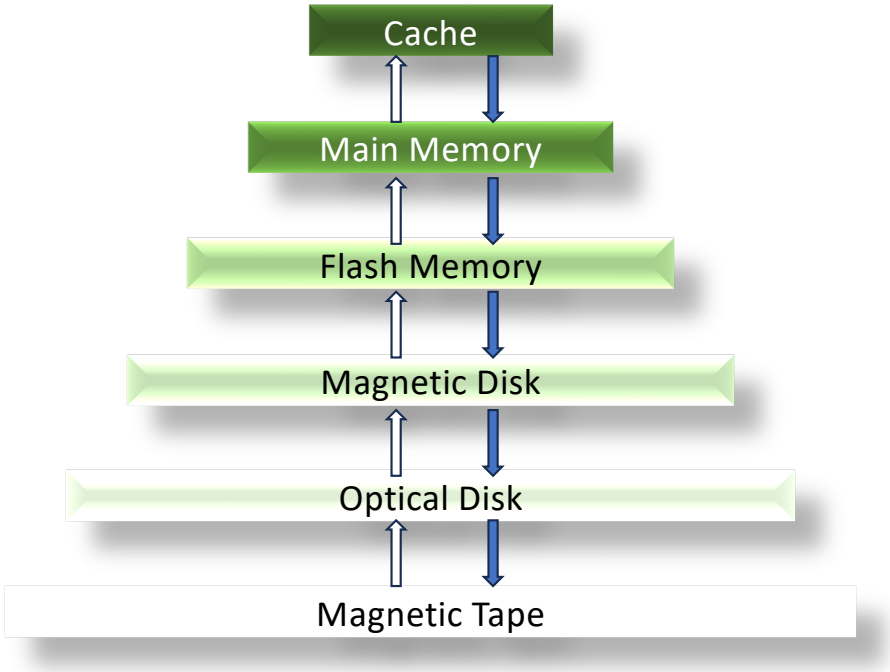
CH 14.1 – 14.7

Office Hours: Every Friday (10:00 – 11:45)



Overview of Storage and Indexing

Storage hierarchy

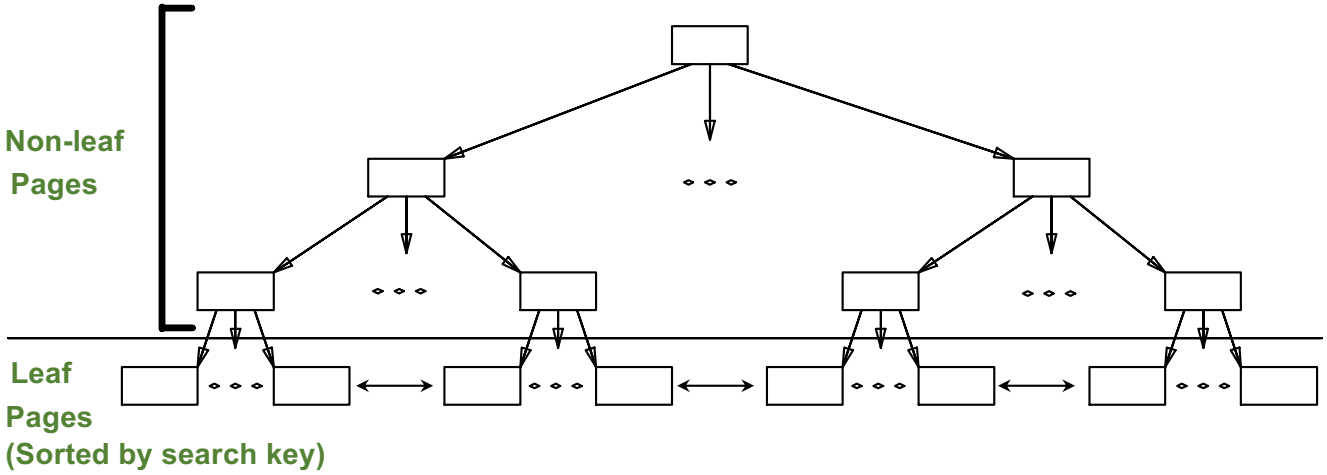


Source: Database System Concepts – 7th Edition

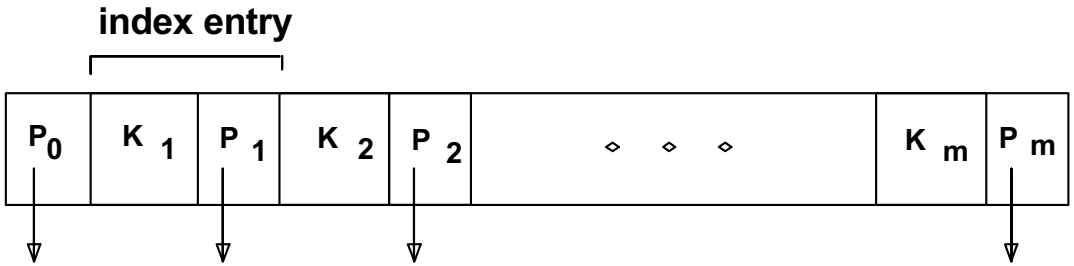
Indexes

- An *index* on a file speeds up selections on the *search key fields* for the index.
  - Any subset of the fields of a relation can be the search key for an index on the relation.
  - *Search key* is NOT the same as *key* (minimal set of fields that uniquely identify a record in a relation).
- An index contains a collection of *data entries*, and supports efficient retrieval of all data entries **k*** with a given key value **k**.
  - Given data entry k*, we can find record with key k in at most one disk I/O. (Details soon …)

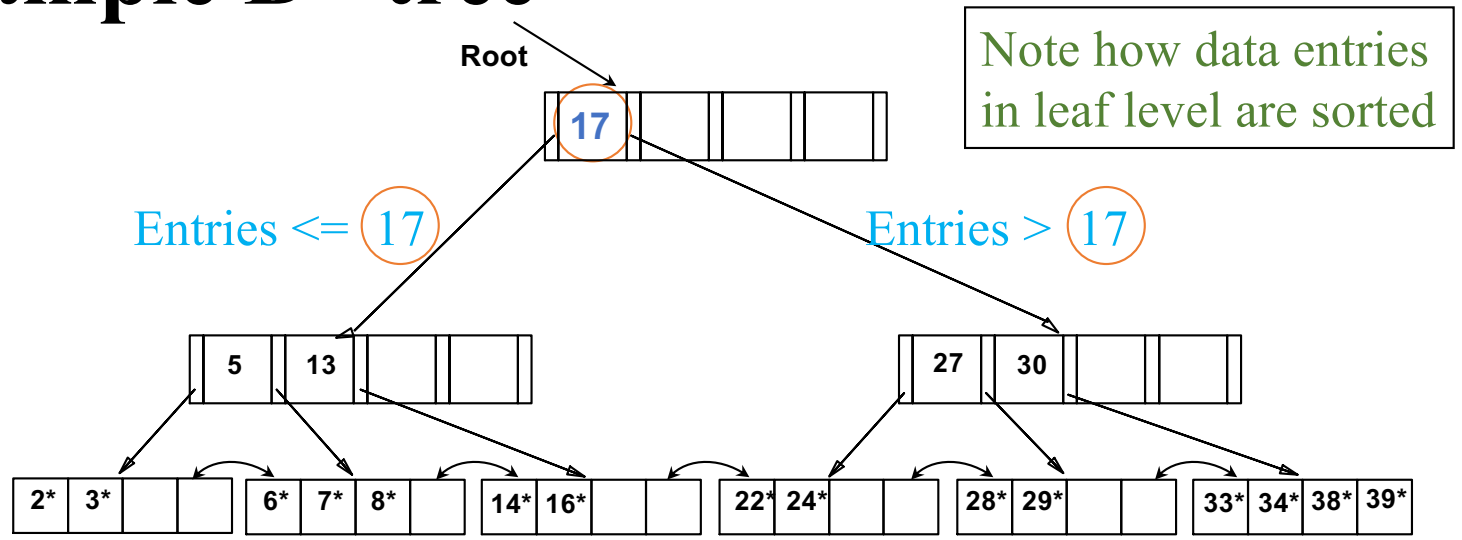
B+ Tree Indexes



- Leaf pages contain *data entries*, and are chained (prev & next)
- Non-leaf pages have *index entries*; only used to direct searches:



Example B+ tree

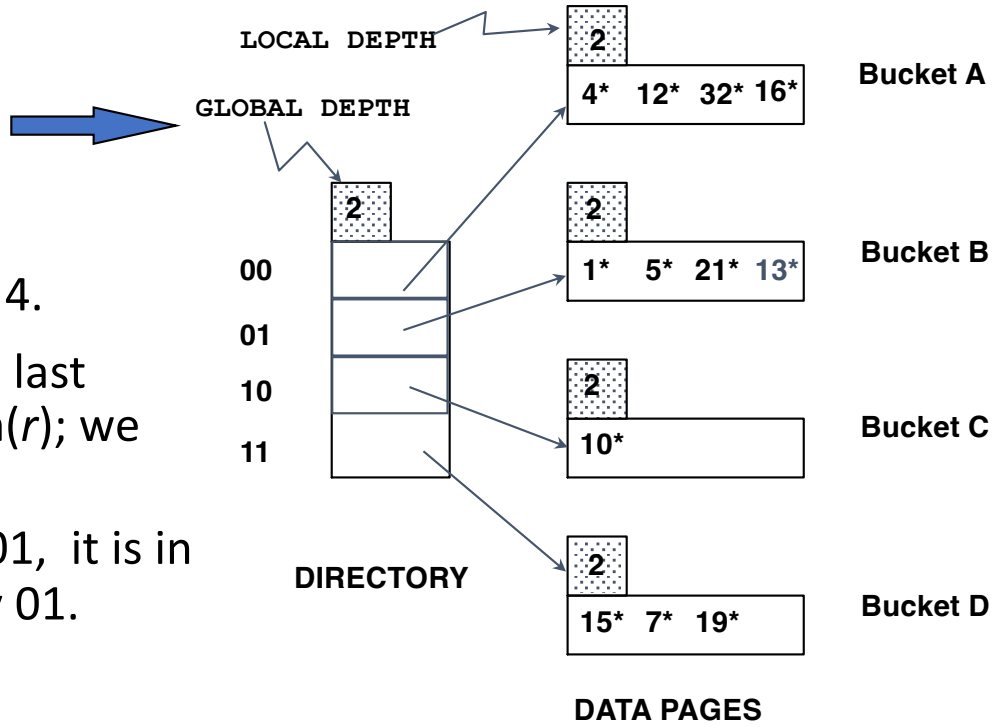


- Find 28*? 29*? All > 15* and < 30*
- Insert/delete: Find data entry in leaf, then change it. Need to adjust parent sometimes.
  - And change sometimes bubbles up the tree

Hash-based indexes

- Good for equality selections.
- Index is a collection of *buckets*.
  - Bucket = *primary* page plus zero or more *overflow* pages.
  - Buckets contain data entries.
- *Hashing function* **h**: **h**(*r*) = bucket in which (data entry for) record *r* belongs. **h** looks at the *search key* fields of *r.*
  - *No need for "index entries" in this scheme.*

Example



- Directory is array of size 4.
- To find bucket for *r*, take last `*global depth*' # bits of **h**(*r*); we denote *r* by **h**(*r*).
  - If **h**(*r*) = 5 = binary 101, it is in bucket pointed to by 01.

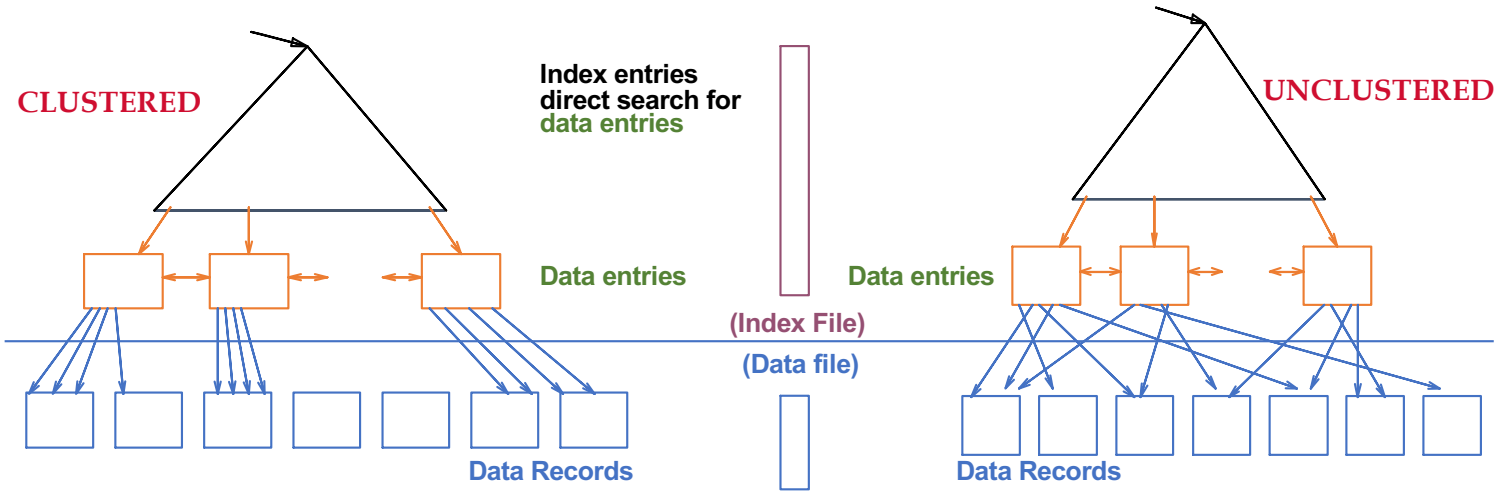
❖ **Insert**: If bucket is full, *split* it (*allocate new page, re-distribute*).

❖ *If necessary*, double the directory. (Splitting a bucket does not always require doubling; we can tell by comparing *global depth* with *local depth* for the split bucket.)

Index classification

- *Primary* vs. *secondary*: If search key contains primary key, then called primary index.
  - *Unique* index: Search key contains a candidate key.
- *Clustered* vs. *unclustered*:
  - Some database systems allow declaring one of the indices to be clustered.
  - The system stores the relation sorted by the search-key of the clustered index

Clustered vs. unclustered index



Index creation/deletion

- Single column index

```
CREATE INDEX <index_name> ON table_name(column_name)
```

  - Example

```
CREATE INDEX idxInstSalary ON instructor(salary)
```

- Use CREATE UNIQUE INDEX to indirectly enforce the condition that the search key is a candidate key
- To drop an index

```
DROP INDEX <index_name>
```

Which index to use?

- B+ tree index on E.age can be used to get qualifying tuples.
  - Is better to use clustered index?
- Consider the GROUP BY query.
  - If many tuples have *E.age* > 10, using *E.age* index and sorting the retrieved tuples may be costly.
  - Clustered *E.dno* index may be better!
- Equality queries and duplicates:
  - Clustering on *E.hobby* helps!

```
SELECT  E.dno
FROM  Emp E
WHERE  E.age>40
```

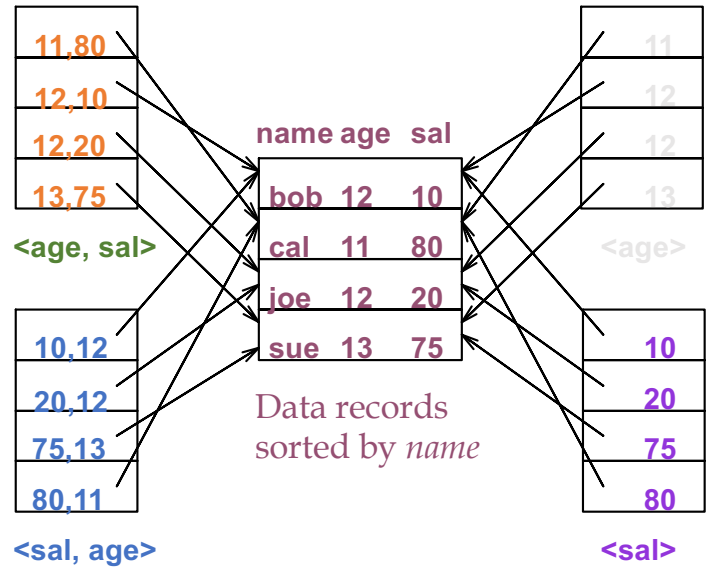
```
SELECT  E.dno,  COUNT (*)
FROM  Emp E
WHERE  E.age>10
GROUP BY E.dno
```

```
SELECT  E.dno
FROM  Emp E
WHERE  E.hobby=Stamps
```

Indexes with composite search keys

- *Composite Search Keys*: Search on a combination of fields.
- Data entries in index sorted by search key to support range queries.
  - E.g. Lexicographic order

Examples of composite key indexes using lexicographic order.



Data entries in index sorted by <*sal,age*>

Data entries sorted by <*sal*>

Composite search keys

- To retrieve Emp records with *age*=30 AND *sal*=4000, an index on <*age,sal*> would be better than an index on *age* or an index on *sal*.
- If condition is 20<*age*<30 AND 3000<*sal*<5000:
  - Clustered tree index on <*age,sal*> or <*sal,age*> is best.
- If condition is *age*=30 AND 3000<*sal*<5000:
  - Clustered <*age,sal*> index much better than <*sal,age*> index!
- Composite indexes are larger, updated more often
- Can be created using:

```
CREATE INDEX <index_name> ON table_name(column1, column2)
```

Composite indexes for single attribute search

- An index on <*age,sal*> can be used to retrieve records with *age*=30 (or age>10) but is NOT of any help for retrieving records with sal=20 (or sal>30)

Index-only plans

- A number of queries can be answered without retrieving any tuples from one or more of the relations involved if a suitable index is available.

<*E.dno*>

```
SELECT  E.dno, COUNT(*)
FROM  Emp E
GROUP BY  E.dno
```

<*E.dno,E.sal*>

Tree index!

```
SELECT  E.dno, MIN(E.sal)
FROM  Emp E
GROUP BY  E.dno
```

<*E. age,E.sal*>
or
<*E.sal, E.age*>

Tree index!

```
SELECT AVG(E.sal)
FROM  Emp E
WHERE  E.age=25 AND
  E.sal BETWEEN 3000 AND 5000
```

Index-only plans (contd.)

- Index-only plans are possible if the key is <dno,age> or we have a tree index with key <age,dno>
  - Which is better?
  - What if we consider the second query?

```
SELECT  E.dno,  COUNT (*)
FROM  Emp E
WHERE  E.age=30
GROUP BY E.dno
```

```
SELECT  E.dno,  COUNT (*)
FROM  Emp E
WHERE  E.age>30
GROUP BY E.dno
```

Choice of indexes

- What indexes should we create?
  - Which relations should have indexes? What field(s) should be the search key? Should we build several indexes?
- For each index, what kind of an index should it be?
  - Clustered? Hash/tree?

Choice of indexes (contd.)

- One approach: Consider the most important queries in turn. Consider the best plan using the current indexes and see if a better plan is possible with an additional index. If so, create it.
  - Obviously, this implies that we must understand how a DBMS evaluates queries and creates query evaluation plans!
  - For now, we discuss simple 1-table queries.
- Before creating an index, must also consider the impact on updates in the workload!
  - Trade-off: Indexes can make queries go faster, updates slower. Require disk space, too.

Operations to compare

- Scan: Fetch all records from disk
- Equality search
- Range selection
- Insert a record
- Delete a record

Understanding the workload

- For each query in the workload:
  - Which relations does it access?
  - Which attributes are retrieved?
  - Which attributes are involved in selection/join conditions? How selective are these conditions likely to be?
- For each update in the workload:
  - Which attributes are involved in selection/join conditions? How selective are these conditions likely to be?
  - The type of update (INSERT/DELETE/UPDATE), and the attributes that are affected.

Index selection guidelines

- Attributes in WHERE clause are candidates for index keys.
  - Exact match condition suggests hash index.
  - Range query suggests tree index.
    - Clustering is especially useful for range queries; can also help on equality queries if there are many duplicates.
- Multi-attribute search keys should be considered when a WHERE clause contains several conditions.
  - Order of attributes is important for range queries.
  - Such indexes can sometimes enable index-only strategies for important queries.
    - For index-only strategies, clustering is not important!
- Try to choose indexes that benefit as many queries as possible. Since only one index can be clustered per relation, choose it based on important queries that would benefit the most from clustering.

Data Integration

Data integration

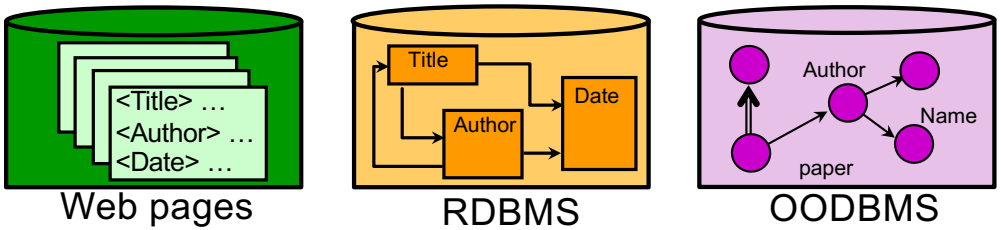
- Organizations use databases: mainframes, workstations, servers
  - Built from scratch each time… usually
  - Many models of the same object
- Need for information sharing across databases
  - Exploit advances in distributed computing and networking

Data integration

- Fact: Databases are managed by different persons
- Challenge 1: Provide uniform access to the users
- Challenge 2: Allow DB to interoperate but …
  - Autonomous
  - Different OS
  - Different purposes
  - Different data modeling
  - Different data formats
  - Different access and communication protocols

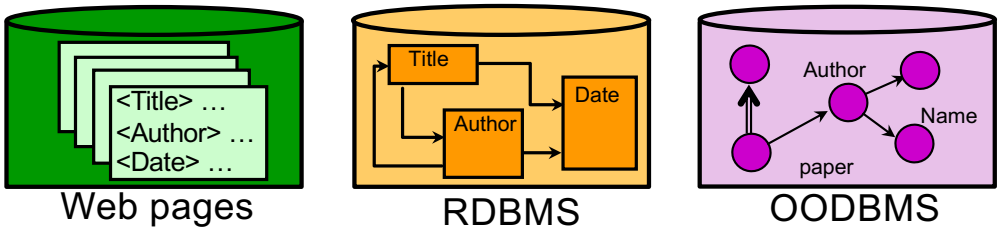
Information is everywhere

- Databases .. but not only
- Mail messages
- Web pages
- Spreadsheets
- Text documents
- Multimedia annotations
- XML Documents



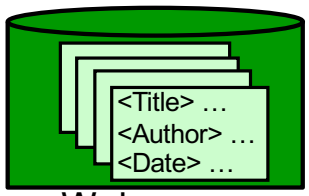
There are many needs

- Exchanged
- Replicated
- Migrated
- Integrated
- Adapted

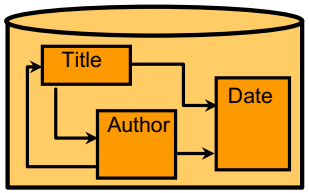


Data in different forms

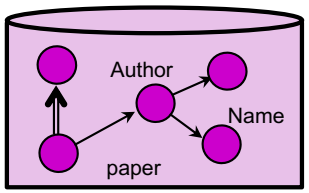
- Personalization: content adapted to user
  - upon system's decision
  - upon user's request
- Customization: structure adapted to user
  - according to the user's role
  - upon user's request
- System Dependent
  - Performance Reasons
- Context dependence
  - User, Device, Network, Place, Time, Rate



Web pages



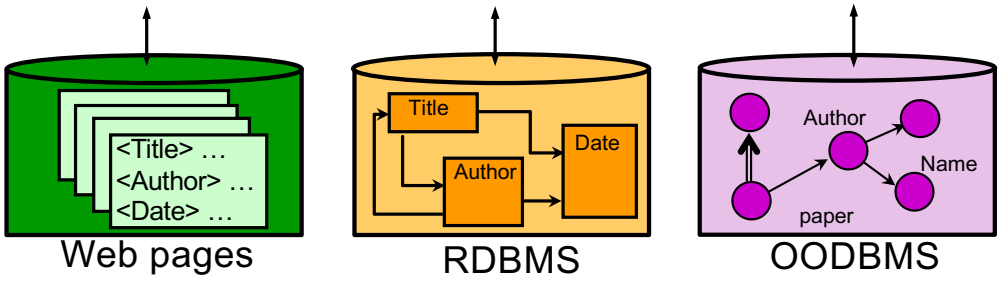
RDBMS



OODBMS

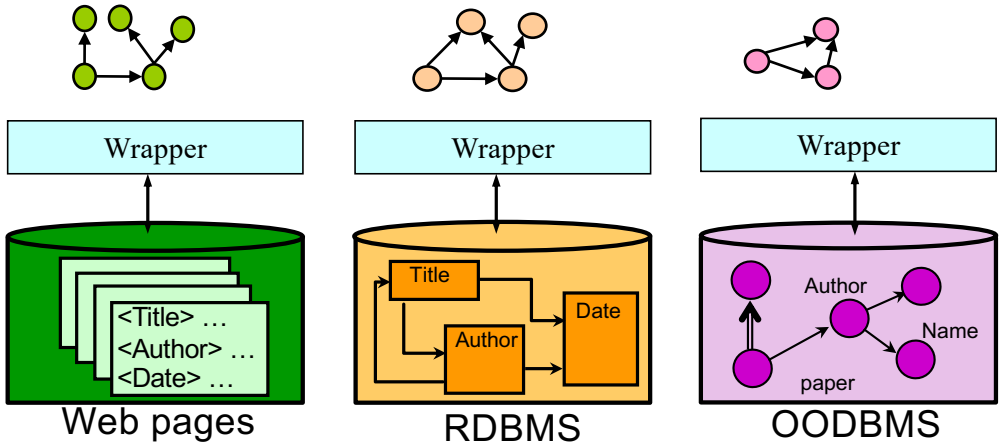
System level

- Protocols
  - Predefined methods of communication

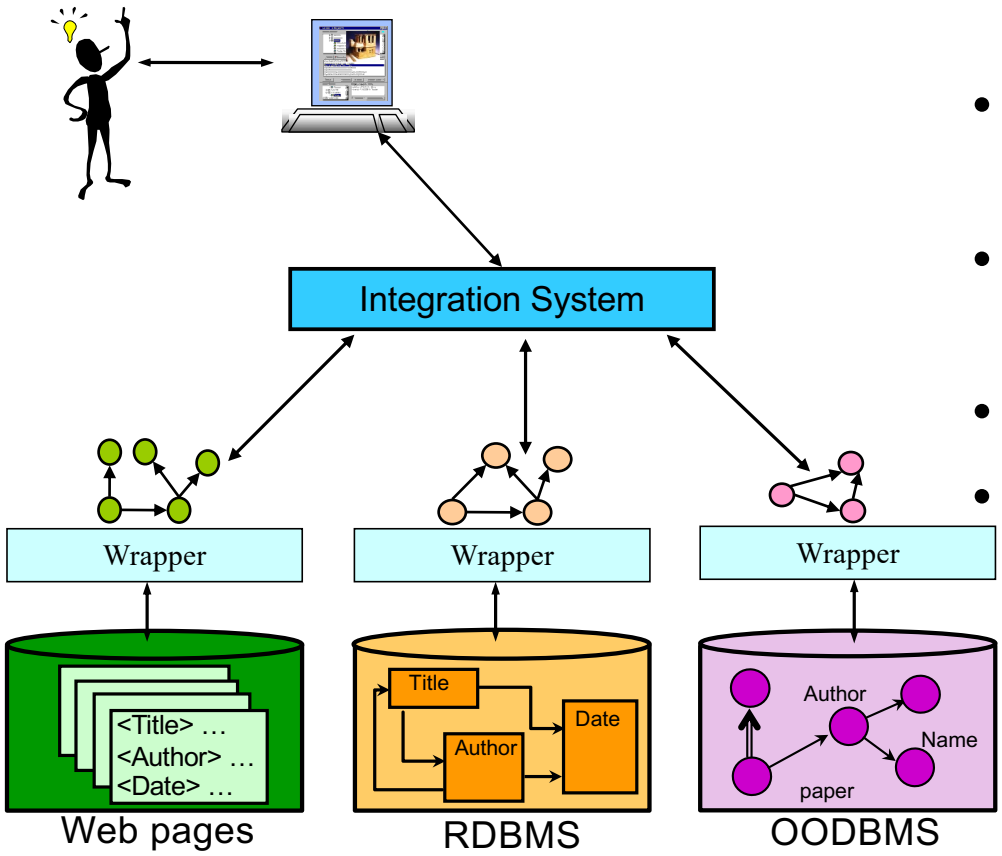


Model level

- Wrappers
  - Wrap the sources into a common data model



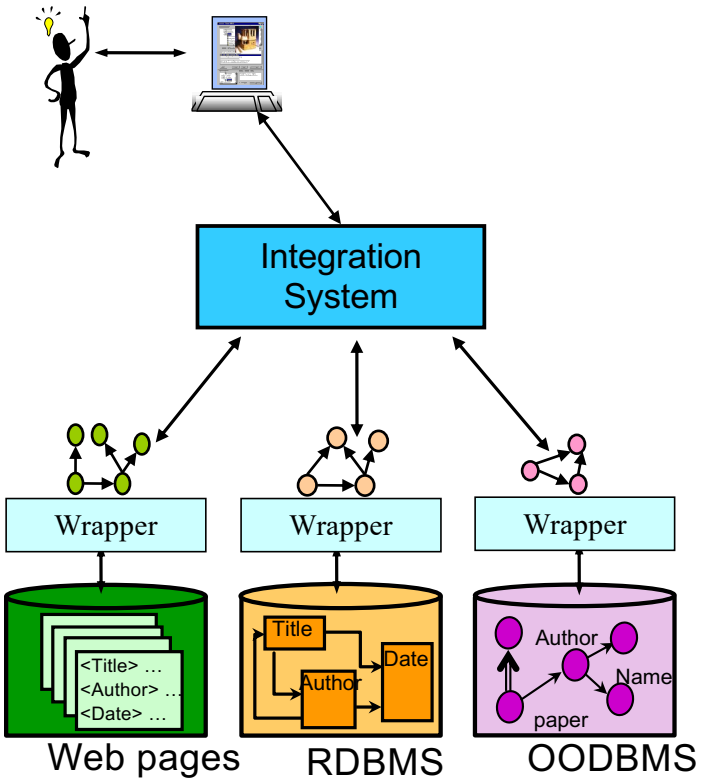
Structural and semantic level



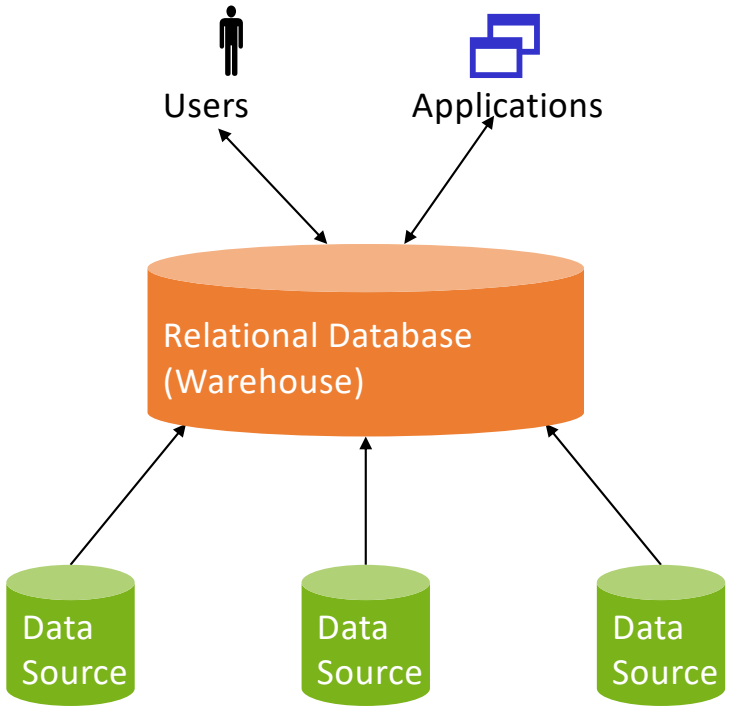
- Structural
  - Different structures for the same data
- Semantic
  - Different semantics of the same structure
- Integrated View
- Many architectures for this basic idea

An information integration system

- A set of Local Databases
  - Each with a Local Schema & a Local Instance
- A Global Integrated Schema
- A set of Mappings
  - Between the Global and the Local Schemas
  - Describe the relationship between the data in the sources and the data as the user “sees” them



Data warehouse architecture



OLAP / Decision Support
Data Cubes / Data Mining

ETL Tools
(Extract-Transform-Load)

Data Cleaning

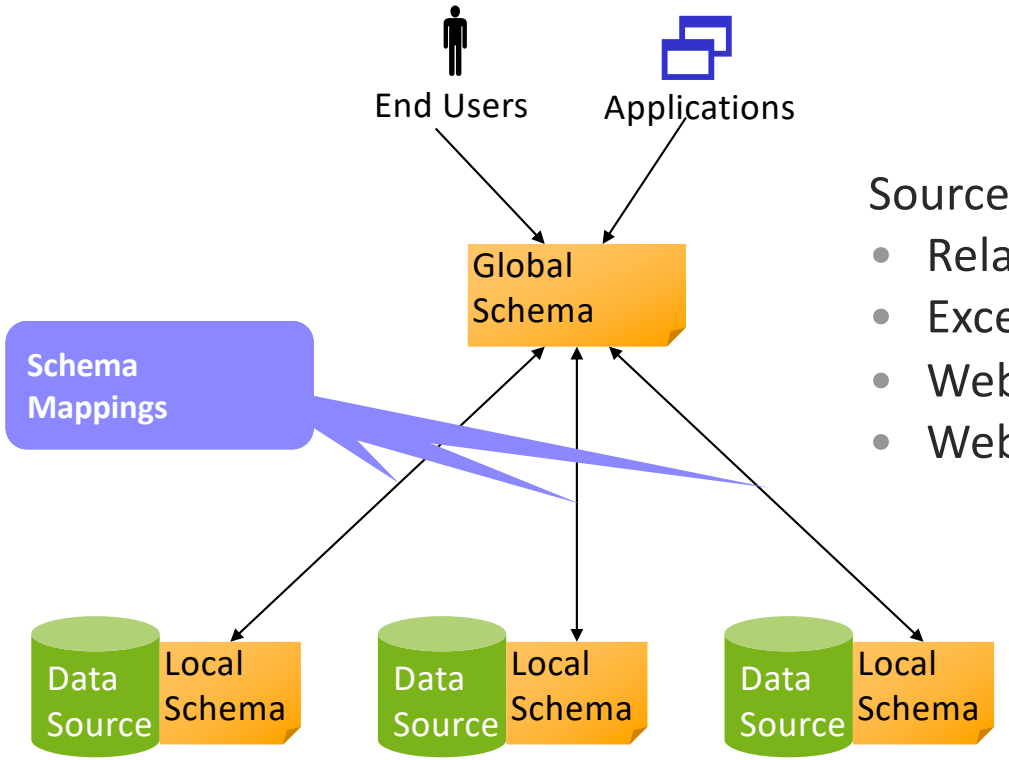
Virtual integration architecture

- Leave the data in the sources
- When a query comes in:
  - Determine the relevant sources to the query
  - Break down the query into sub-queries for the sources
  - Get the answers from the sources, filter them if needed and combine them appropriately
- Data is fresh
- Otherwise known as

On Demand Integration

Virtual integration architecture

Design-Time



Sources can be:

- Relational DBs
- Excel Files
- Web Sites
- Web Services

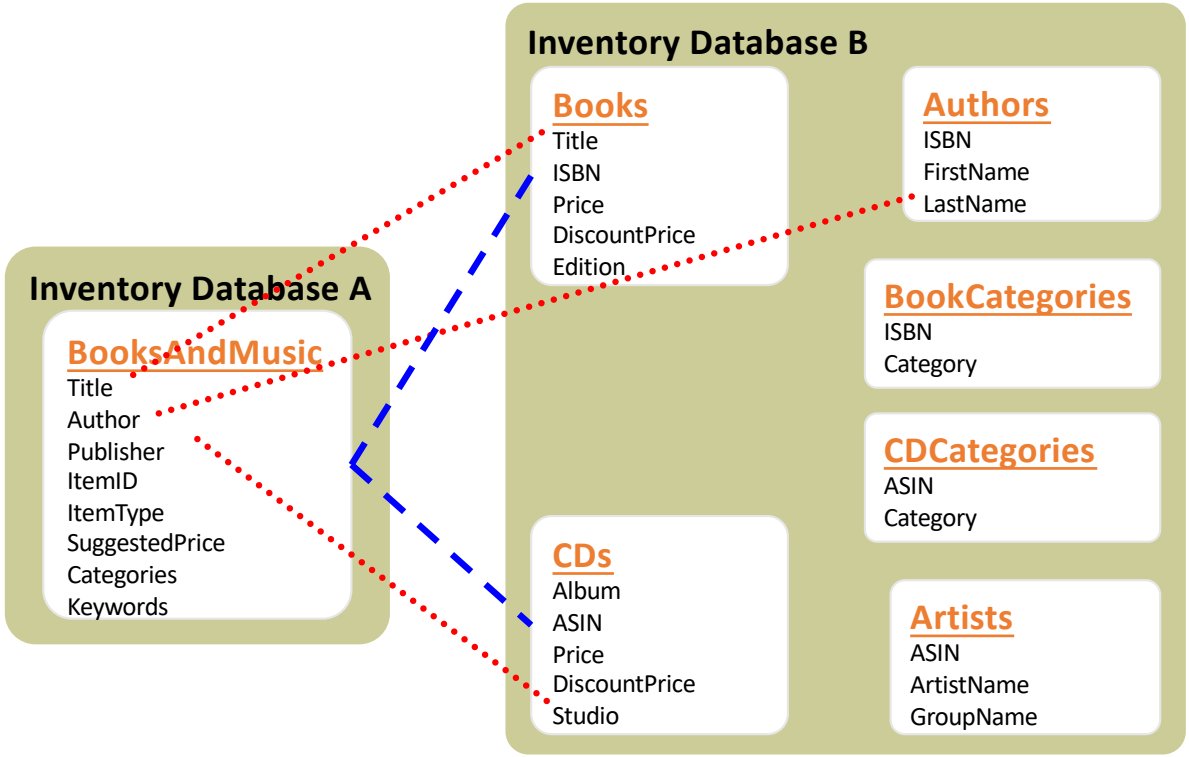
- Summarize what you learned today in 2-minutes

The information in this presentation has been compiled with the utmost care,
but no rights can be derived from its contents.

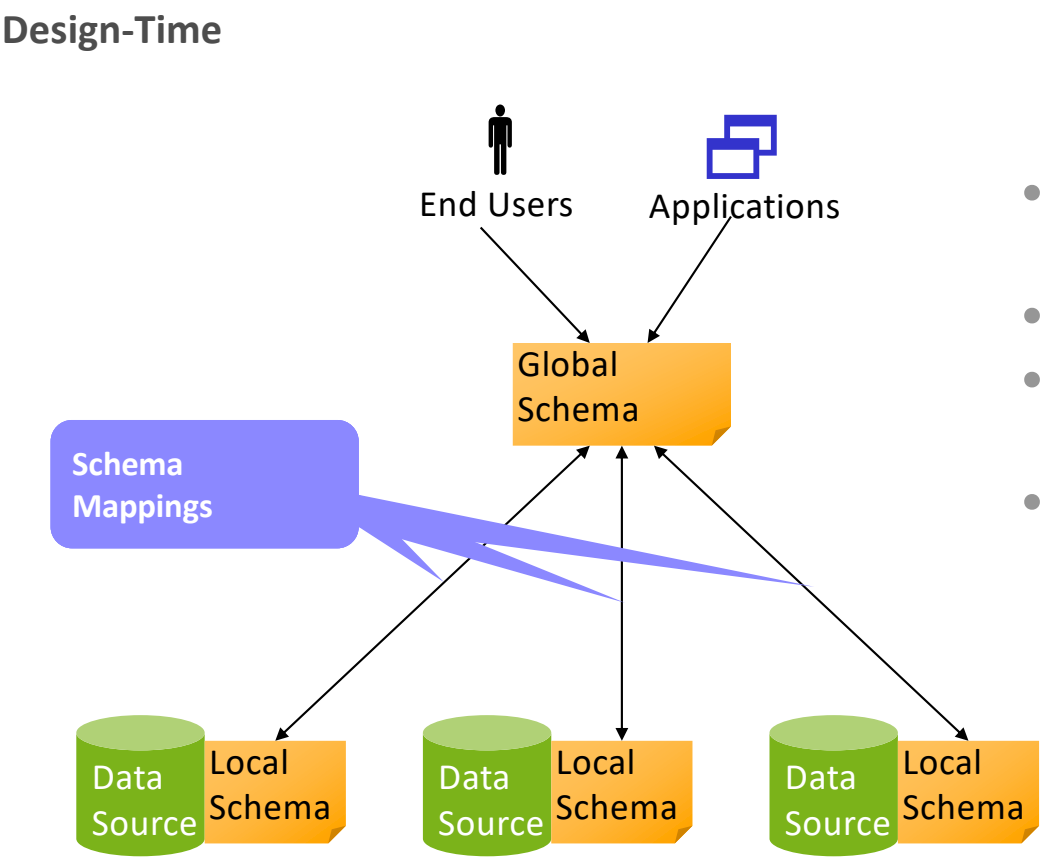
The rest of the slides is for your own interest –
They are not included in the exams

Schema mappings

- Differences in:
  - Names in schema
  - Attribute grouping
  - Coverage of databases
  - Granularity and format of attributes



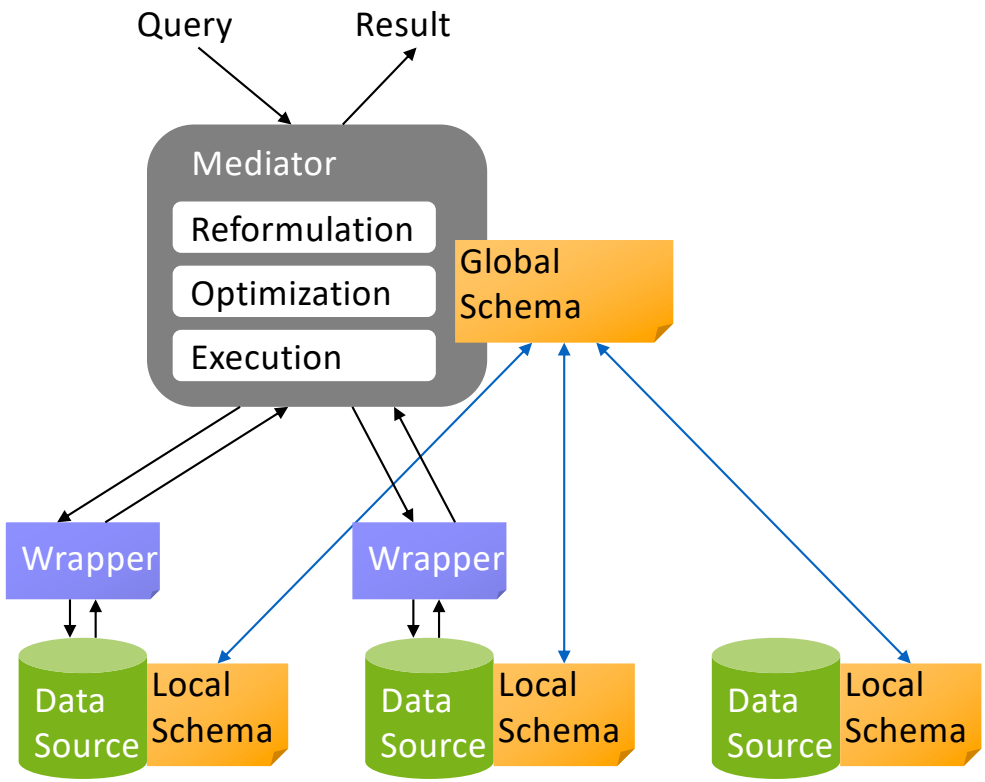
Issues for schema mappings



- What formalisms to express them?
- How to create them?
- Can we discover them somehow?
- How do we use them?

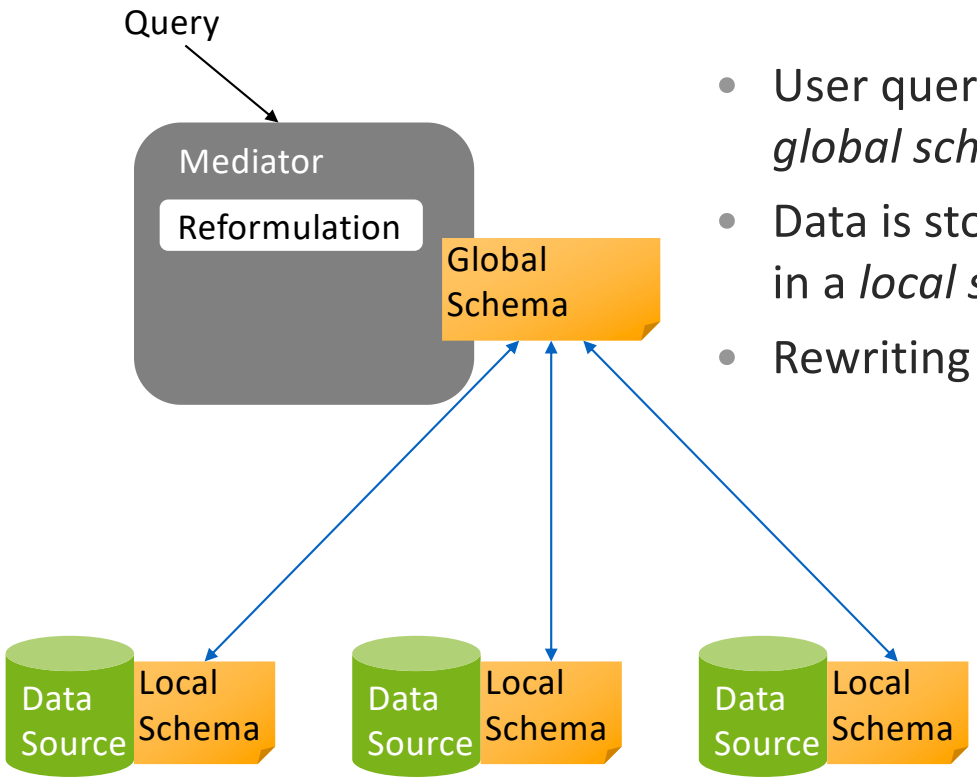
Virtual integration architecture

Run-Time



Issues for query processing

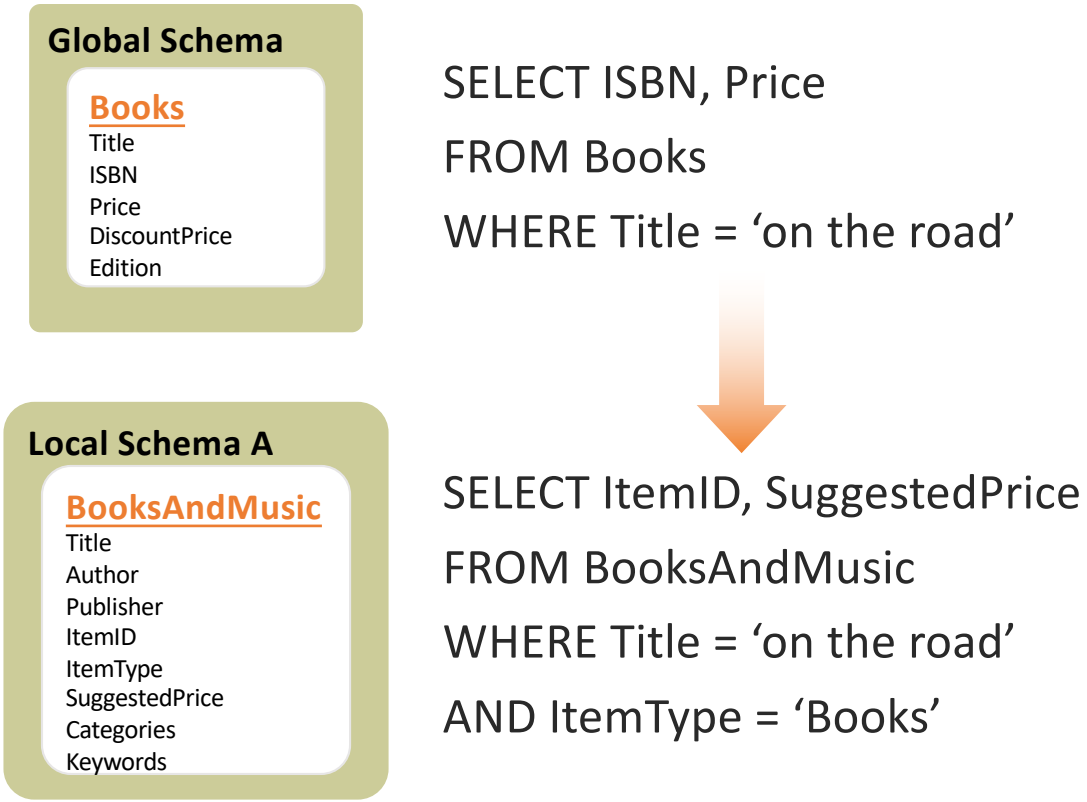
Reformulation



- User queries refer to the *global schema*
- Data is stored in the sources in a *local schema*
- Rewriting algorithms

Issues for query processing

Reformulation

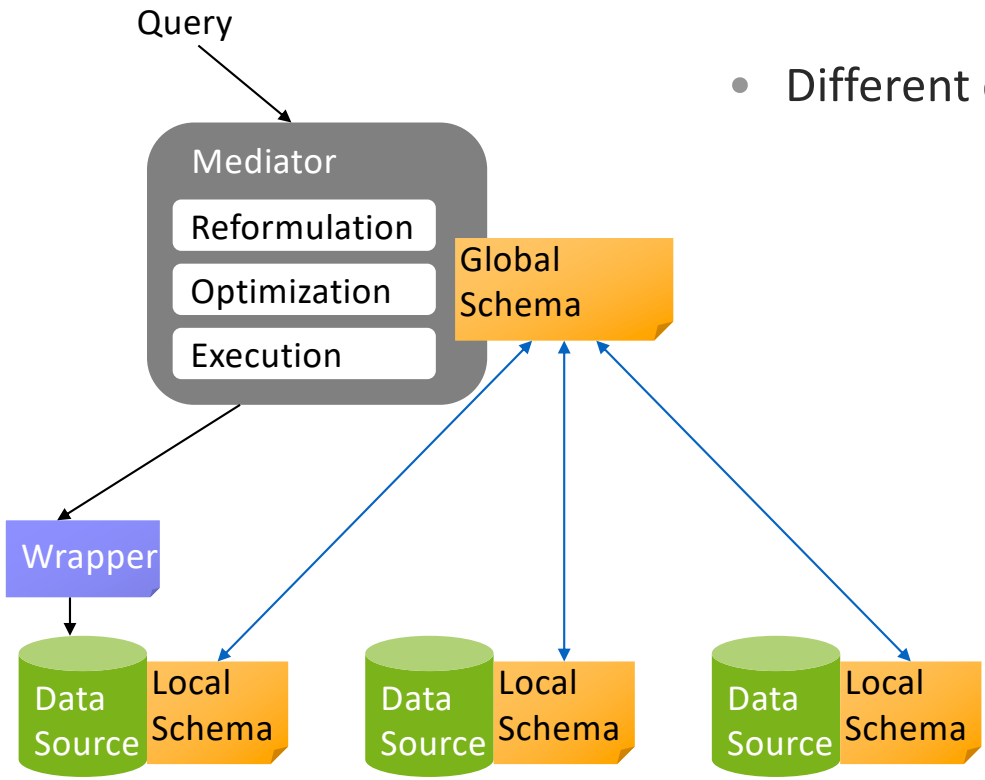



```
SELECT ISBN, Price
FROM Books
WHERE Title = 'on the road'
```

```
SELECT ItemID, SuggestedPrice
FROM BooksAndMusic
WHERE Title = 'on the road'
AND ItemType = 'Books'
```

Issues for query processing

Query Translation



- Different query languages

Issues for query processing

Query Translation

Global Schema

Books
Title
ISBN
Price
DiscountPrice
Edition

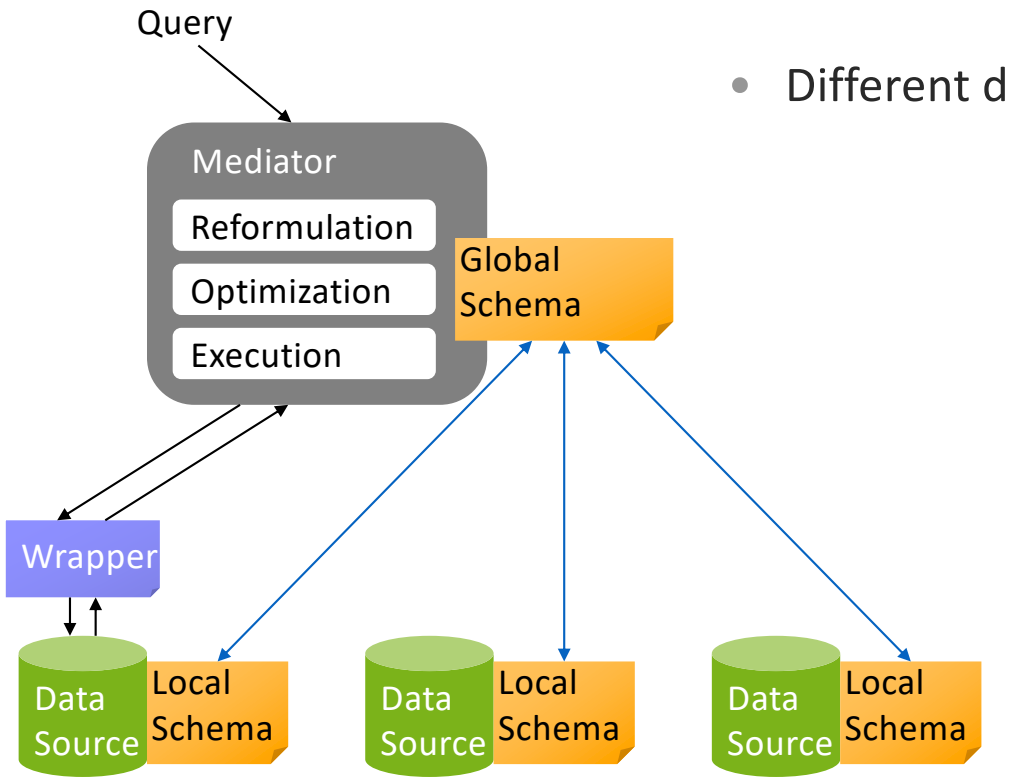
```
SELECT ISBN, Price
FROM Books
WHERE Title = 'on the road'
```

Local Source A

http://www.amazon.com/homepage.html?ItemType=Books&Title=on+the+road

Issues for query processing

Data Translation



- Different data models

Issues for query processing

Data Translation

Global Schema

Books
Title
ISBN
Price
DiscountPrice
Edition

Local Result A

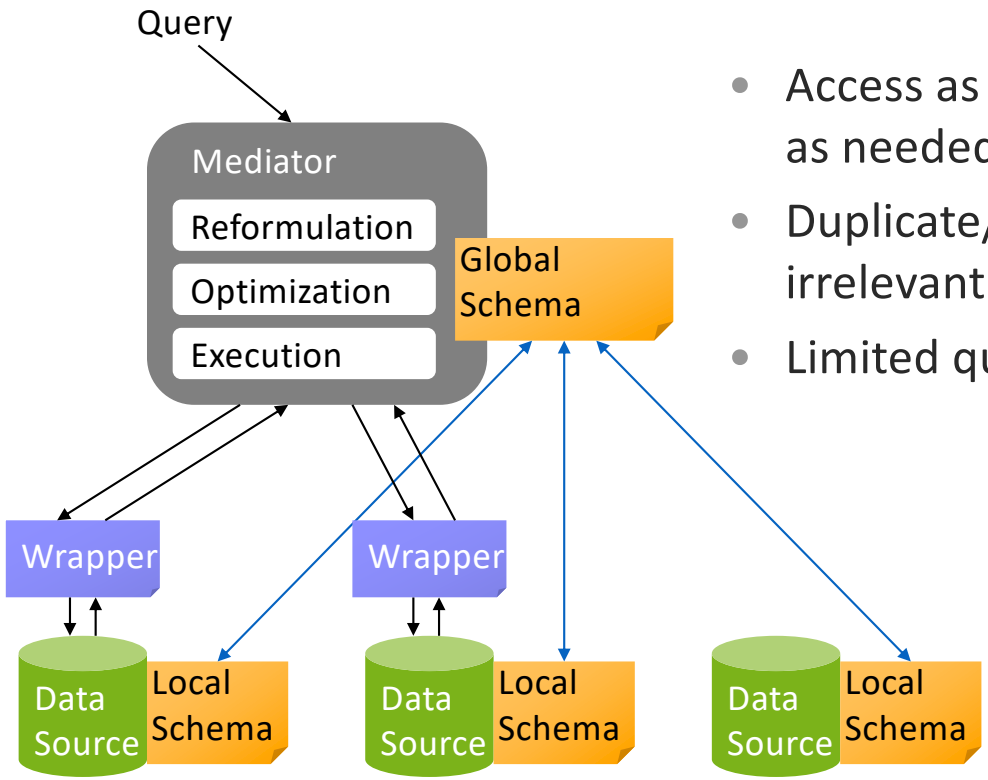
On the Road -- by Jack Kerouac; Paperback (Rate it)
Buy new: $10.86 -- Used & new from: $5.90

Title	ISBN	Price	...	...
On the Road	123	10.86	...	...

```
<table>
  <tr>
    <td>
      <a href=/details?isbn=123>
        <b>On the Road</b>
      </a>
      -- by Jack Kerouac; Paperback
      <br>
      <a href=/details?isbn=123>
        Buy new
      </a>
      :<b class=price>$10.86</b>
    </td>
  </tr>
</table>
```

Issues for query processing

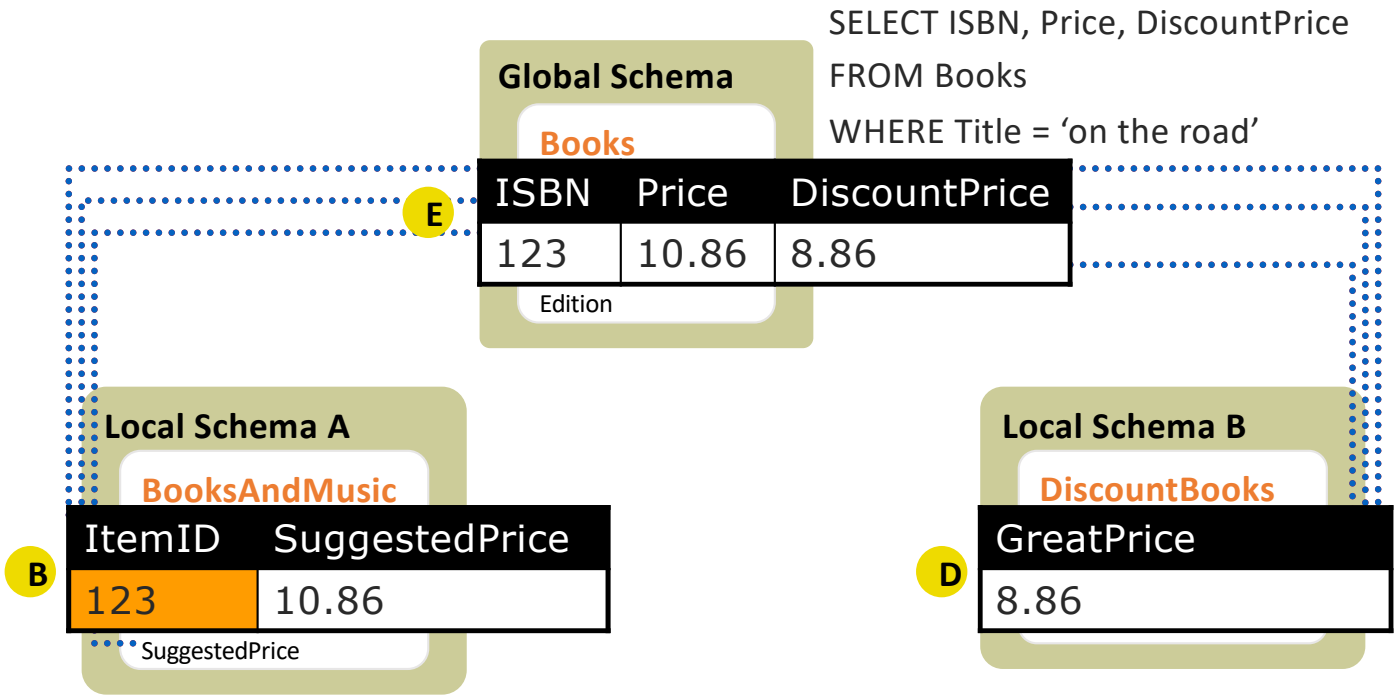
Query Execution



- Access as many data sources as needed
- Duplicate/redundant and irrelevant data
- Limited query capabilities

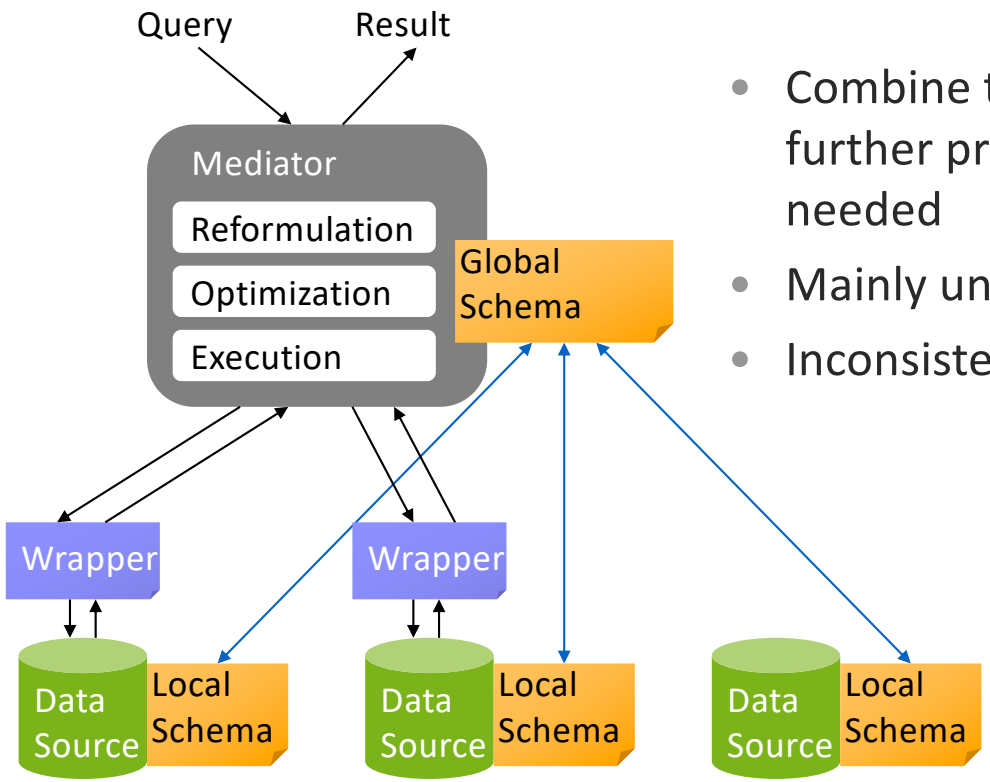
Issues for query processing

Limited Query Capabilities



Issues for query processing

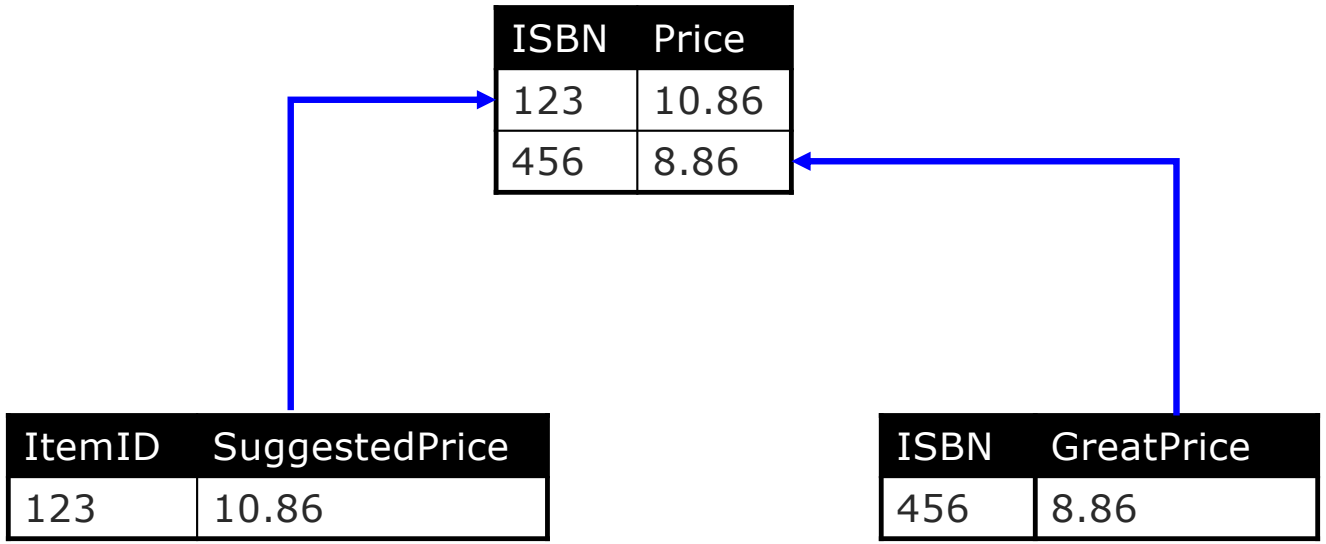
Query Answering



- Combine the results and further process them if needed
- Mainly union and merge
- Inconsistencies

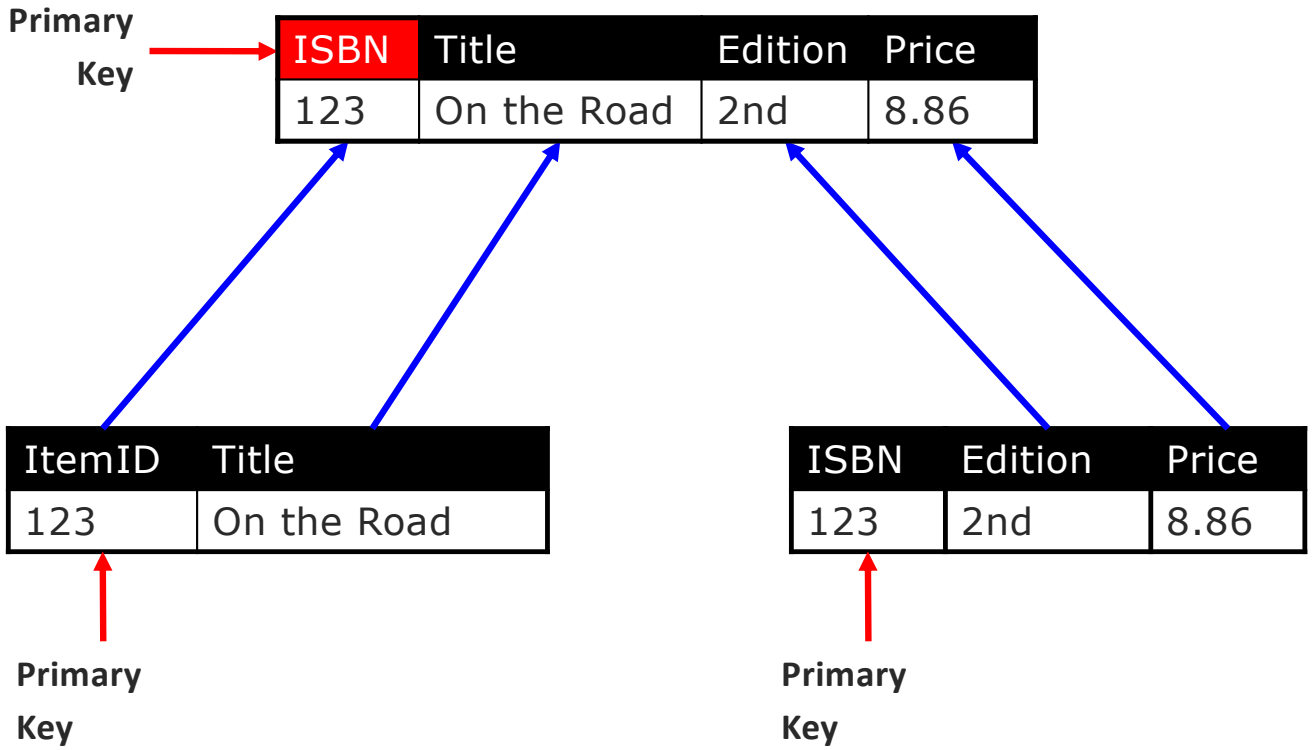
Issues for query processing

Query Answering (Union)



Issues for query processing

Query Answering (Merge)



Issues for query processing

Query Answering (Inconsistencies)