CS585/DS503
Big Data Management
Spring 2019

Relational DBs: Overview

Elke Rundensteiner

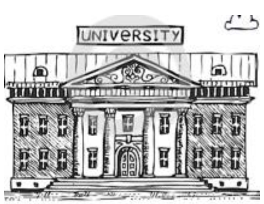
Relational DBs: History

Traditional Applications

Banking

Retail Sys

Airlines



Universities

- Started in mid 70s
- IBM developed the relational DB model
- System R

Relational DBs: Core Assumptions

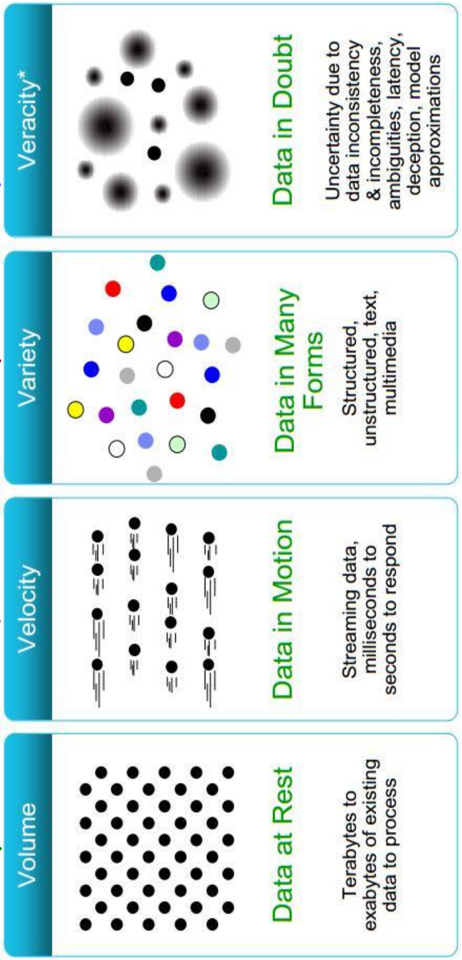
- **Data is nicely structured (known in advance)**
- **Data is correct & certain**
- **Data is relatively static & small-mid size**
- **Access pattern: Mix Read/Write**
- **Notion of transactions (Interactive)**

How these affected the design of DB Systems

Contradicting Assumptions: Big Data

- Data is nicely structured (known in advance)
- Data is correct & certain
- Data is relatively static & small-mid size

- Access pattern: Mix Read/Write
- Notion of transactions



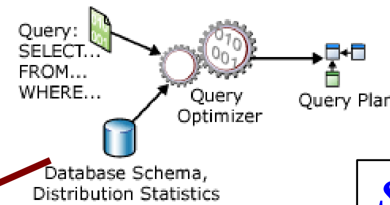
In Big Data: ***It is read only, No notion of transactions***

Assumptions →Design/Solution

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7733903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4859223	O'Reilly Press	77 Boston Ave, Cambridge
03-3920886	City Lights Books	99 Market, San Francisco

AuthorID	AuthorName	AuthorBDay
345-28-2938	Haile Selassie	14-Aug-92
392-48-9965	Joe Blow	14-Mar-15
454-22-4012	Sally Hemmings	12-Sept-70
663-59-1254	Hannah Arendt	12-Mar-06

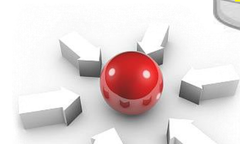
ISBN	AuthorID	PubID	Date	Title
1-34532-482-1	345-28-2938	03-4472822	1990	Cold Fusion for Dummies
1-38482-995-1	392-48-9965	04-7733903	1985	Macrame and Straw Tying
2-35921-499-4	454-22-4012	03-4859223	1952	Fluid Dynamics of Aquaducts
1-38278-293-4	663-59-1254	03-3920886	1967	Beads, Baskets & Revolution



Simple data types,
Straightforward processing

- **Data is nicely structured (known in advance)**
- **Data is correct & certain**
- **Data is relatively static & small-mid size**
- **Access pattern: Mix Read/Write**
- **Notion of transactions (Interactive)**

Disk-based Processing



Centralized

ACID Properties

Recovery Control

Concurrency Control

Relational Model

attributes (or columns)

customer_name	customer_street	customer_city
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

tuples (or rows)

customer

Order of rows & columns does not matter

DB is a set of interconnected relations

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7733903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4859223	O'Reilly Press	77 Boston Ave, Cambridge
03-3920886	City Lights Books	99 Market, San Francisco

AuthorID	AuthorName	AuthorBDay
345-28-2938	Haile Selassie	14-Aug-92
392-48-9965	Joe Blow	14-Mar-15
454-22-4012	Sally Hemmings	12-Sept-70
663-59-1254	Hannah Arendt	12-Mar-06

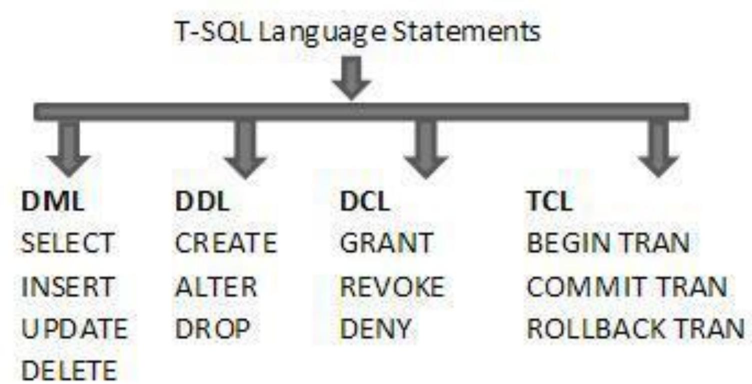
ISBN	AuthorID	PubID	Date	Title
1-34532-482-1	345-28-2938	03-4472822	1990	Cold Fusion for Dummies
1-38482-995-1	392-48-9965	04-7733903	1985	Macrame and Straw Tying
2-35921-499-4	454-22-4012	03-4859223	1952	Fluid Dynamics of Aquaducts
1-38278-293-4	663-59-1254	03-3920886	1967	Beads, Baskets & Revolution

SQL: Structured Query Language

The interface for interaction with DBMSs

```
CREATE TABLE Students
        (sid CHAR(20) Primary Key,
         name CHAR(20),
         login CHAR(10) Unique,
         age INTEGER,
         gpa REAL);
```

```
CREATE TABLE Courses
        (cid Varchar2(20) Primary Key,
         name varchar2(50),
         maxCredits integer,
         graduateFlag char(1));
```



High-Level Declarative Language

Foreign Keys in SQL

- Create "Students" relation

```
CREATE TABLE Students
        (sid CHAR(20) Primary Key ,
         name CHAR(20) NOT NULL,
         login CHAR(10),
         age INTEGER,
         gpa REAL Default 0);
```

- Create "Courses" relation

```
CREATE TABLE Courses
        (cid Varchar2(20) Primary Key,
         name varchar2(50),
         maxCredits integer,
         graduateFlag char(1));
```

- Create "Enrolled" relation

```
CREATE TABLE Enrolled
        (sid: CHAR(20) Foreign Key References Students (sid),
         X: Varchar2(20),
        enrollDate: date,
         grade: CHAR(2),
 Constraint fk_cid Foreign Key (X) References Courses (cid));
```

Two ways to define the FK constraint while creating a table

SQL: Insert, Update, & Delete Data

Performed using Data Manipulation Language of SQL (DML)

- Insertion
  - *Insert into Students values ('1111', …);*
- Deletion
  - *Delete from Students Where 1111';*
- Update
  - *Update Students Set GPA = GPA + 0.4 Where sid = '1111';*

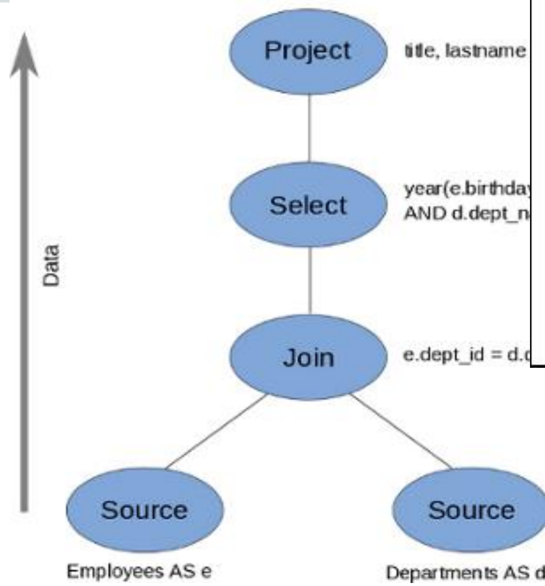
Record-Level Operations

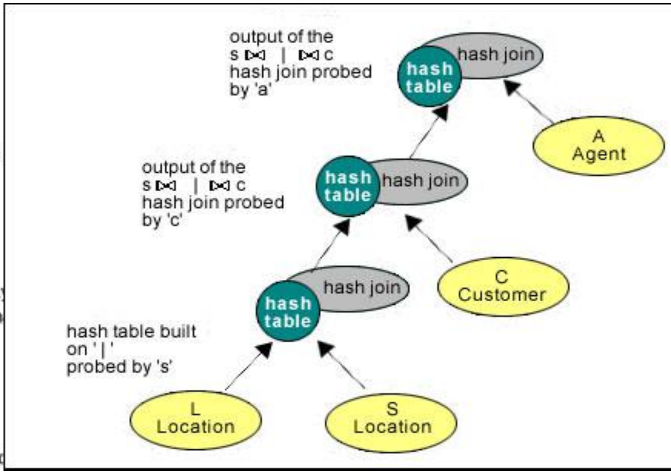
SQL: Query Data (Select Stmt)

SELECT *<projection list>*
FROM *<relation names>*
WHERE *<conditions>*
GROUP BY *<grouping columns>*
HAVING *<grouping conditions>*
ORDER BY <order columns>;

Will be mapped to the algebraic operators

Query Plans





Very Powerful Query Language

Query Processing (Ch. 15 & 16)

Book ➔ Database Systems: The Complete Book, Second Edition

Example

Data:

relation R (A, B, C)

relation S (C, D, E)

Query:

```
Select  B, D
From   R,  S
Where  R.A =  "c"  And  S.E = 2 And  R.C=S.C
```

Relational Algebra – Possible Query Plans

Select B, D **From** R, S **Where** R.A = "c" **And** S.E = 2 **And** R.C=S.C

Plan 1

$$\Pi_{B,D}$$

$$\sigma_{R.A=\text{“c”} \wedge S.E=2 \wedge R.C=S.C}$$

$$\times$$

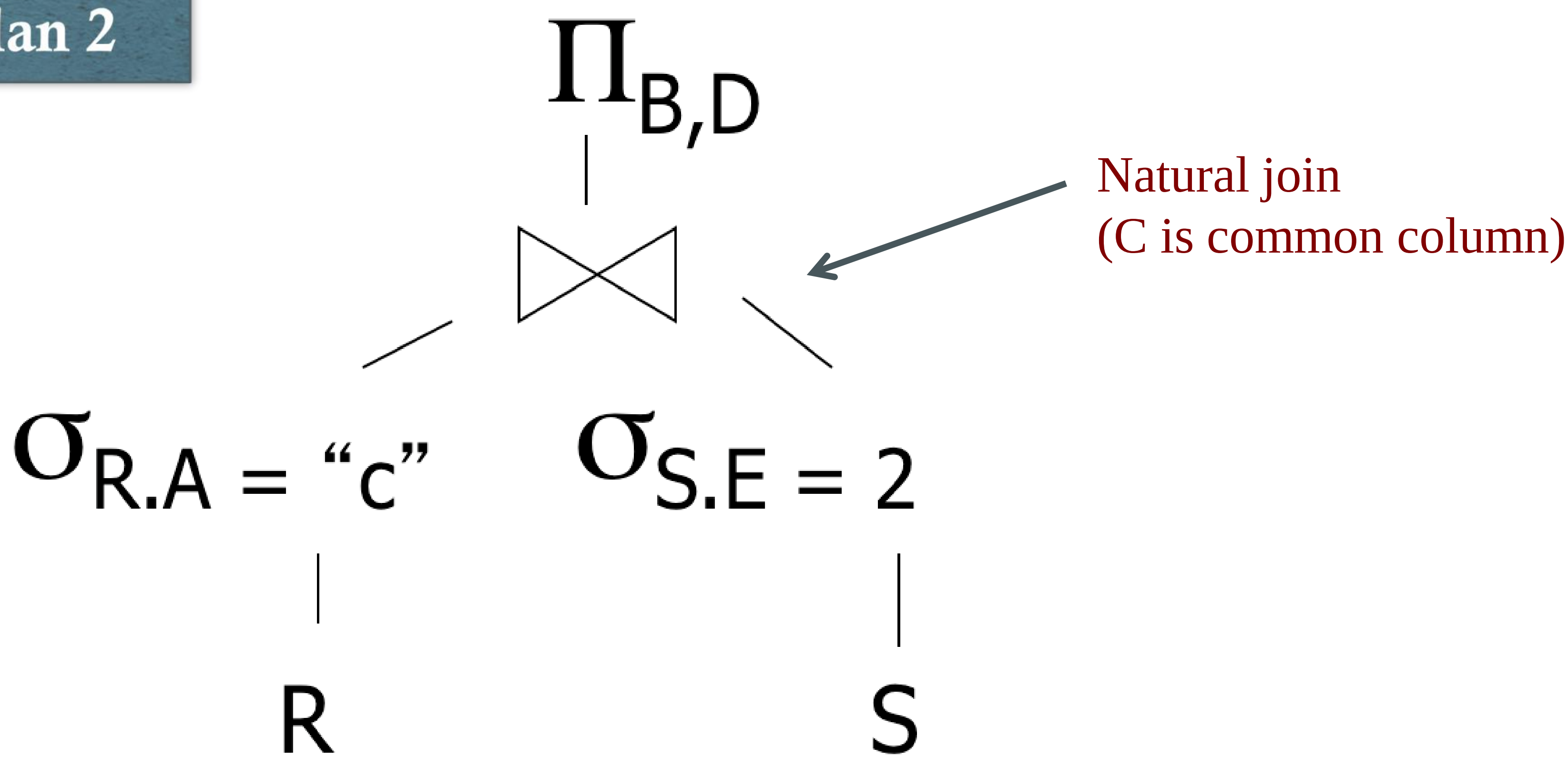
$$R \qquad\qquad S$$

<u>OR:</u> $\Pi_{B,D}\,[\,\sigma_{R.A=\text{“c”} \wedge S.E=2 \wedge R.C=S.C}\,(R \times S)]$

Relational Algebra – Possible Query Plans

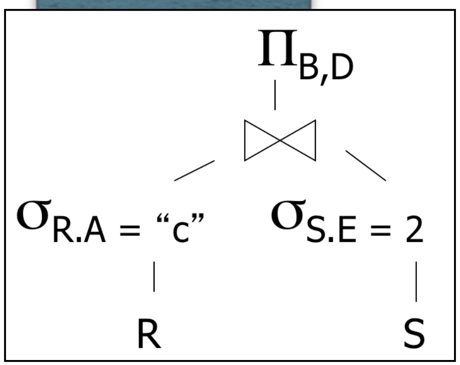
Select B, D **From** R, S **Where** R.A = "c" **And** S.E = 2 **And** R.C=S.C

Plan 2



Select B, D **From** R, S **Where** R.A = "c" **And** S.E = 2 **And** R.C=S.C

Plan 2



R

A	B	C
a	1	10
b	1	20
c	2	10
d	2	35
e	3	45

σ (R)

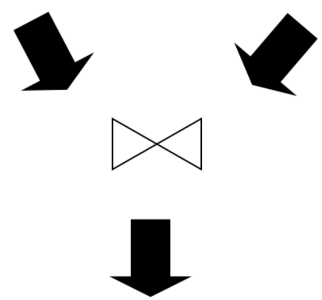
A	B	C
c	2	10

σ(S)

C	D	E
10	x	2
20	y	2
30	z	2

S

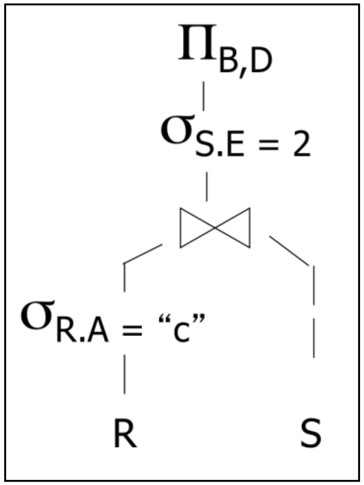
C	D	E
10	x	2
20	y	2
30	z	2
40	x	1
50	y	3



Select B, D **From** R, S **Where** R.A = “c” **And** S.E = 2 **And** R.C=S.C

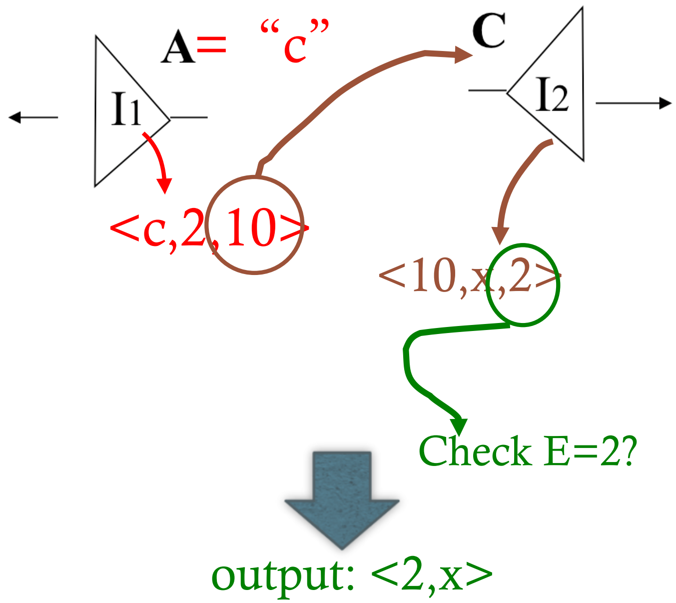
Assume indexes on R.A and S.C

Plan 3



R

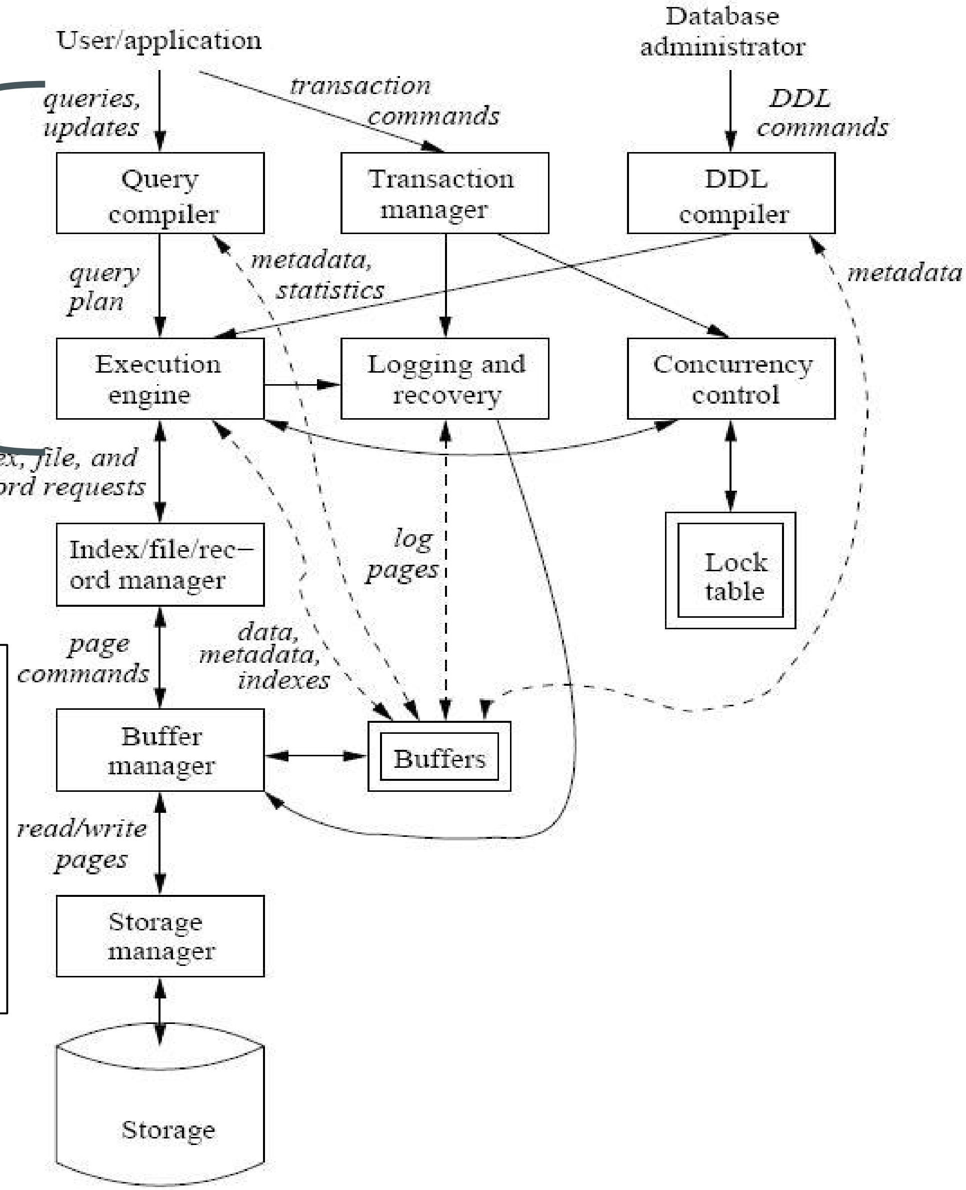
A	B	C
a	1	10
b	1	20
c	2	10
d	2	35
e	3	45



S

C	D	E
10	x	2
20	y	2
30	z	2
40	x	1
50	y	3

Query Compilation, Optimization and Execution



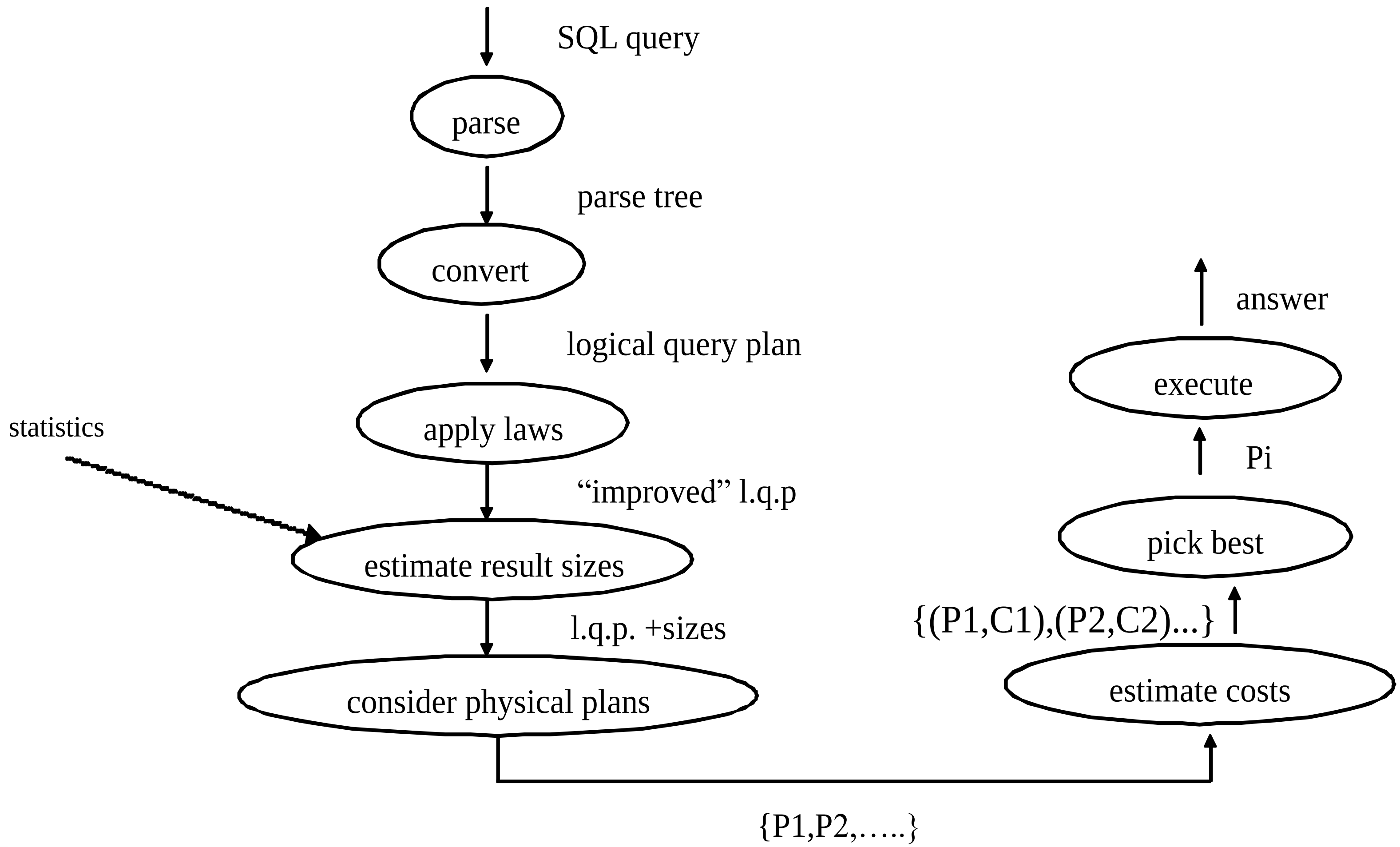
Database management system components

DB has the ability to optimize because:

- Schema & structure is known
- The query is known

Overview of Query Execution

SQL Query ➔ Compile ➔ Optimize ➔ Execute

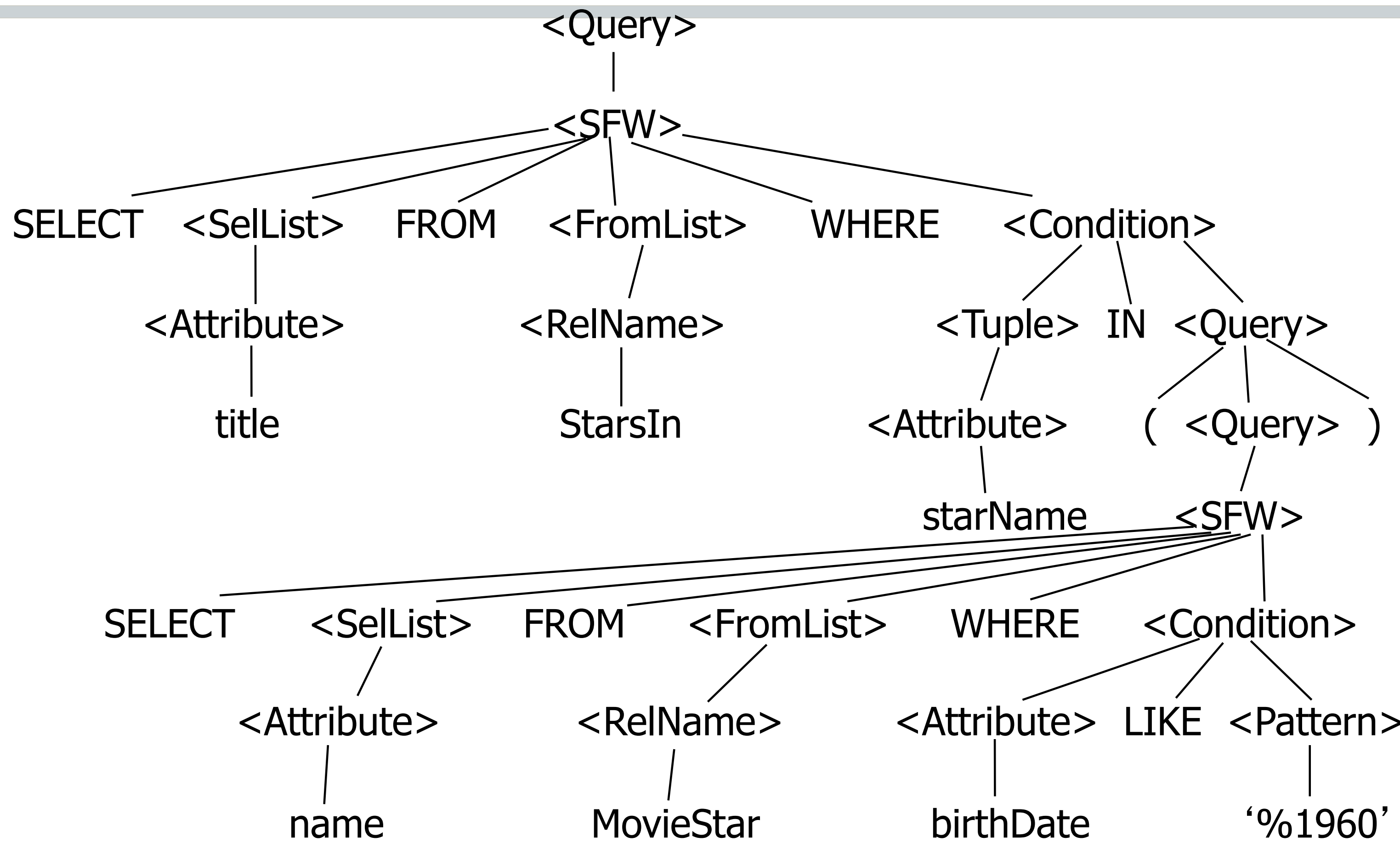


Example

Query : Find the movies with stars born in 1960

```
SELECT title
FROM StarsIn
WHERE starName IN (
        SELECT name
        FROM MovieStar
        WHERE birthdate LIKE ‘%1960’ );
```

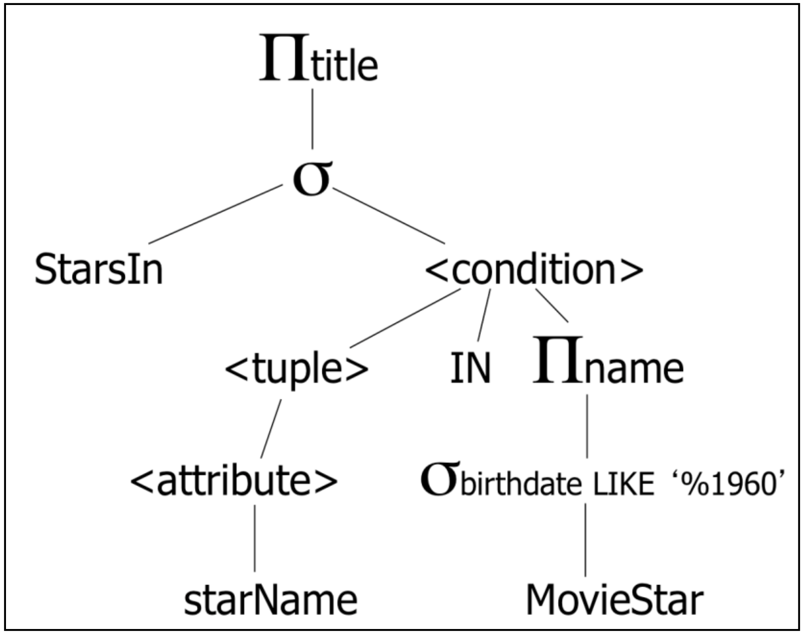
Step 1: Generate Parse Tree



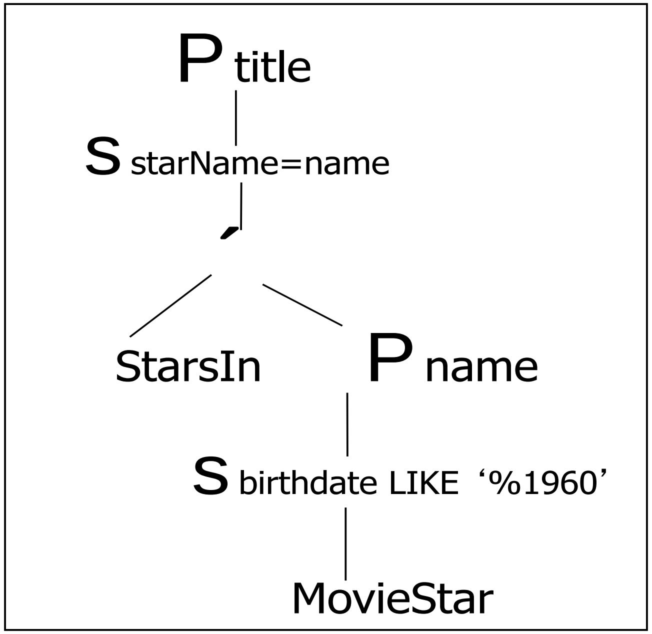
Step 2: Relational Algebra & Logical Plan

```
SELECT title
FROM StarsIn
WHERE starName IN (
   SELECT name FROM MovieStar WHERE birthdate LIKE '%1960' );
```

Expression Tree



Logical Query Plan

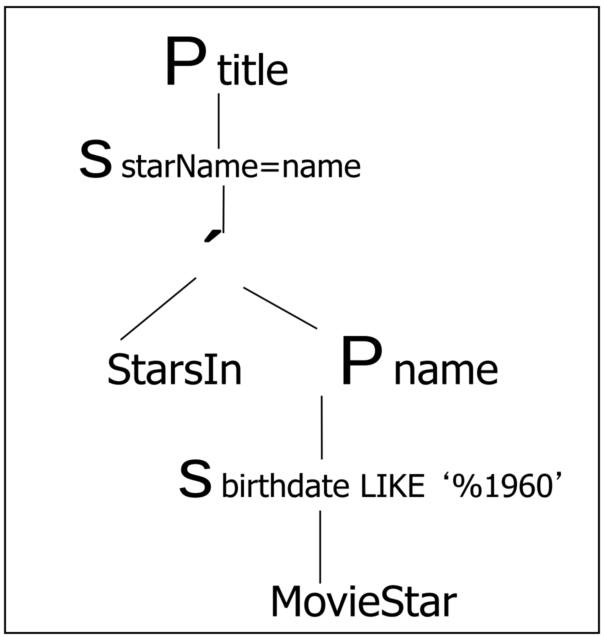


Expression Tree is a midway between a parse tree and relational algebra

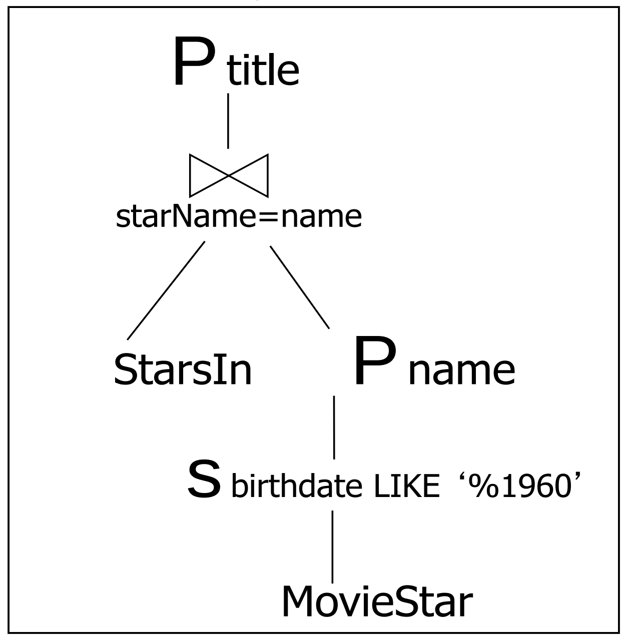
Step 3: Optimize & Create Several Logical Plans

```
SELECT title
FROM StarsIn
WHERE starName IN (
    SELECT name FROM MovieStar WHERE birthdate LIKE '%1960' );
```

Plan 1



Plan 2

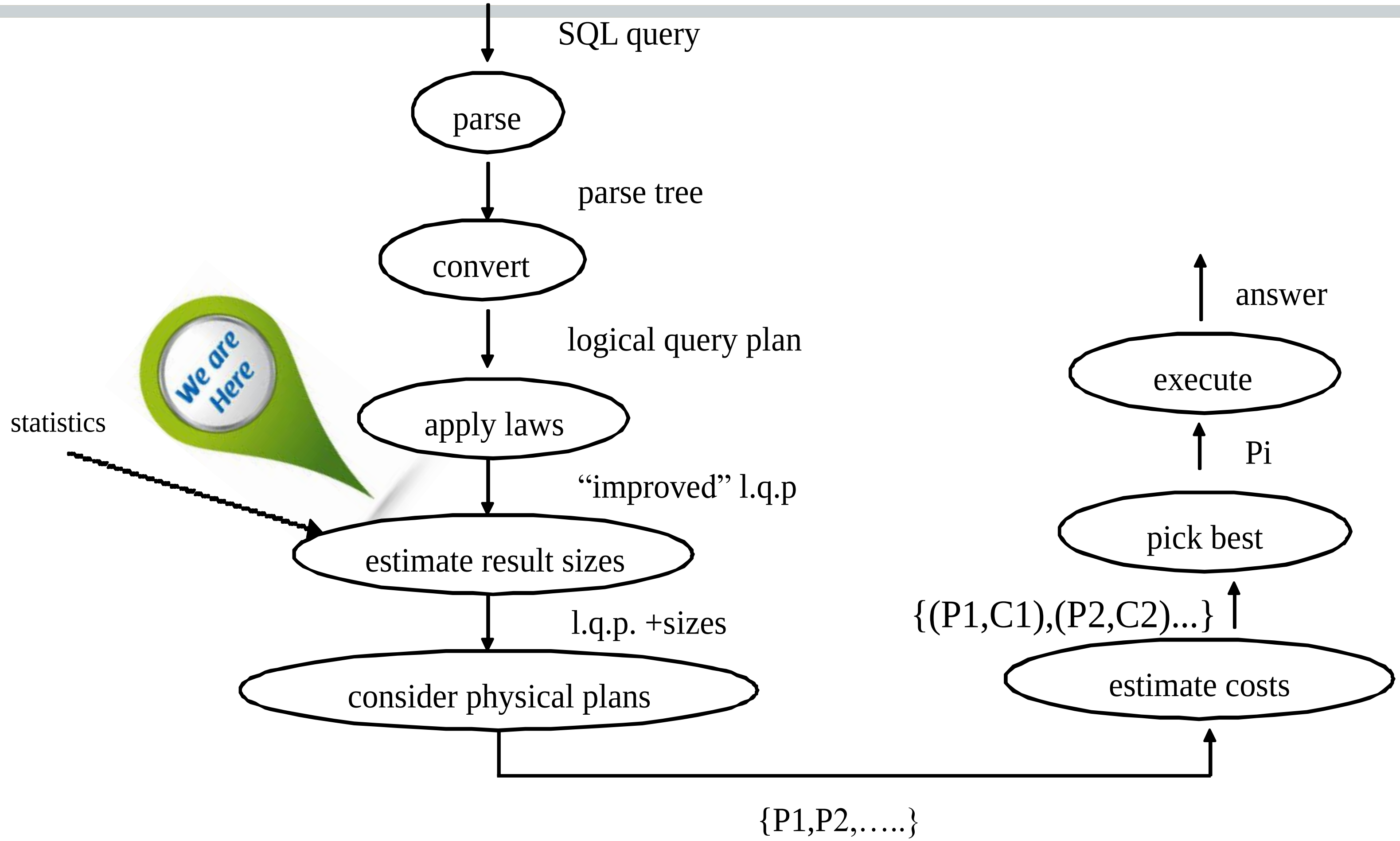


……

Question: Push project to StarsIn?

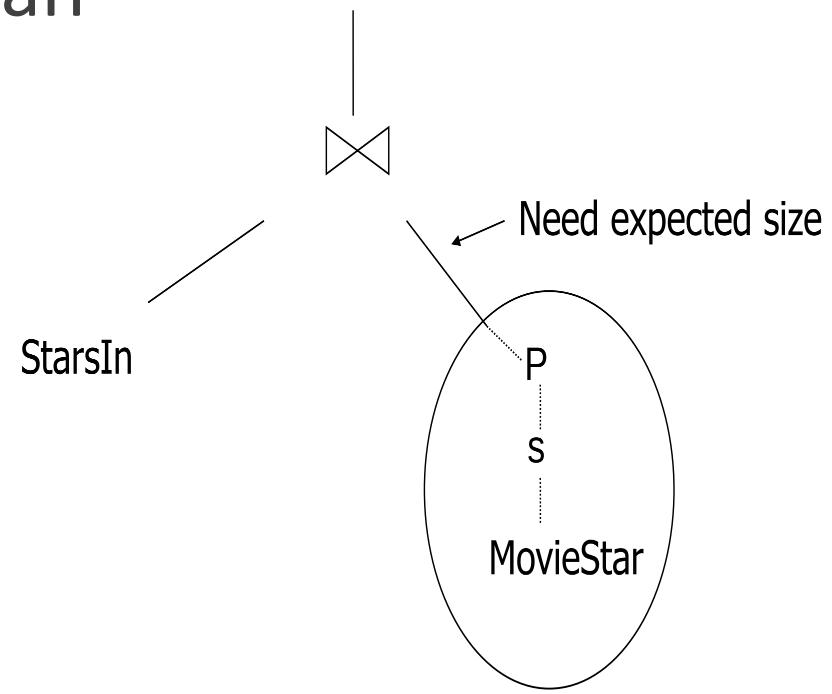
Overview of Query Execution

SQL Query ➔ Compile ➔ Optimize ➔ Execute



Step 4: Estimate the Sizes

- That is done for each plan
- **Statistics Collection**
  - For the entire relation
  - For each column
  - …

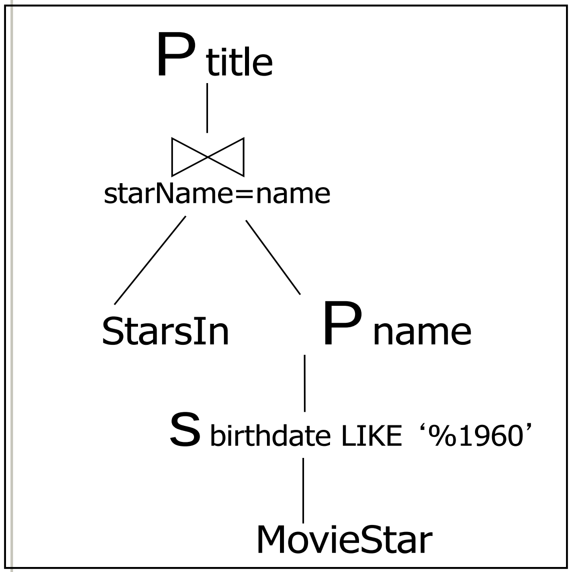


Step 5: Consider Physical Plans

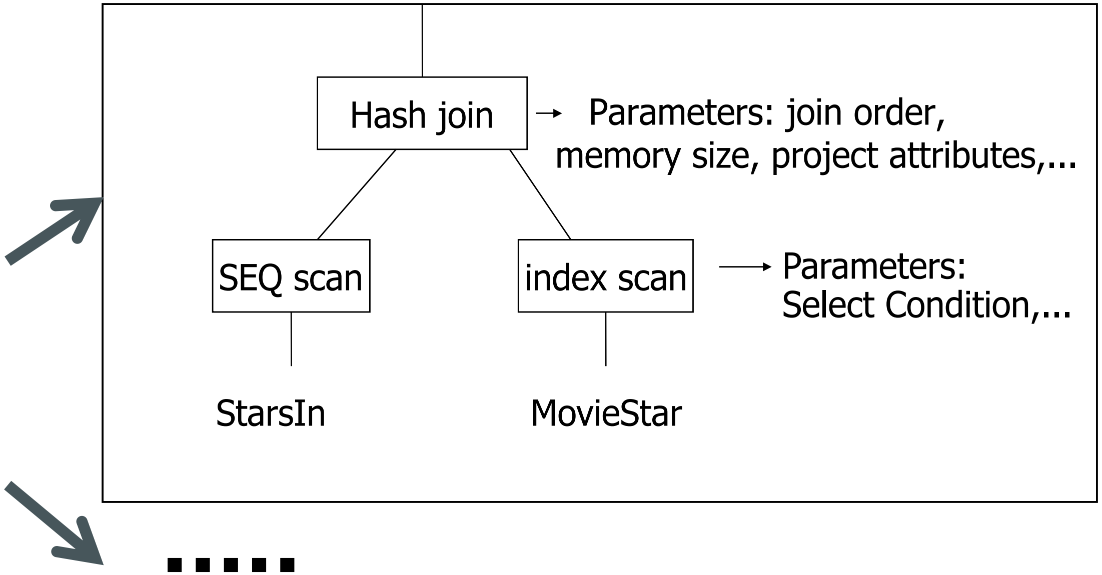
- **Physical plan means how each operator will execute (which algorithm)**
  - E.g., **Join** can be nested-loop, hash-based, merge-based, or sort-based

- **Each logical plan will map to multiple physical plans**

Logical Plan

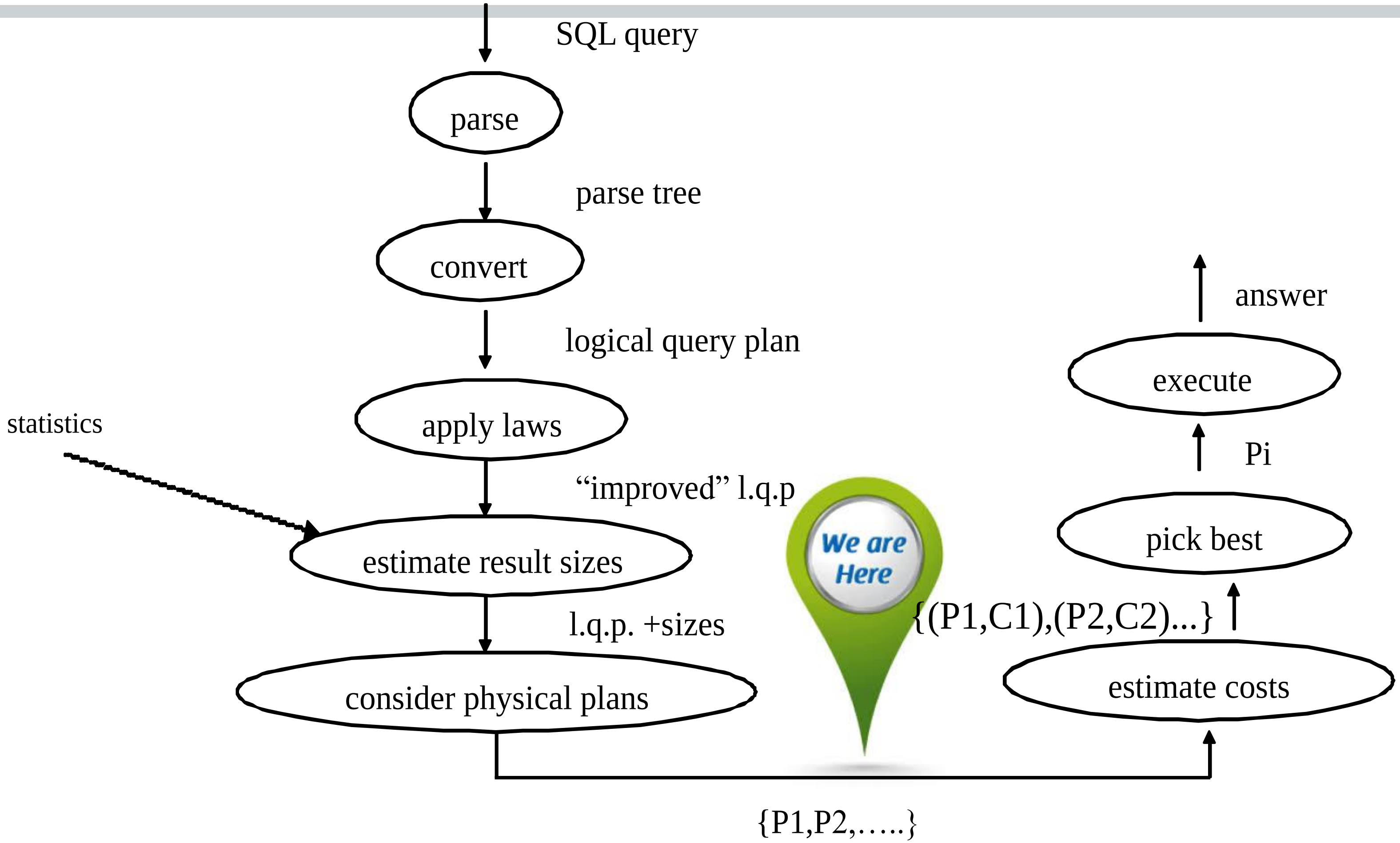


One Physical Plan



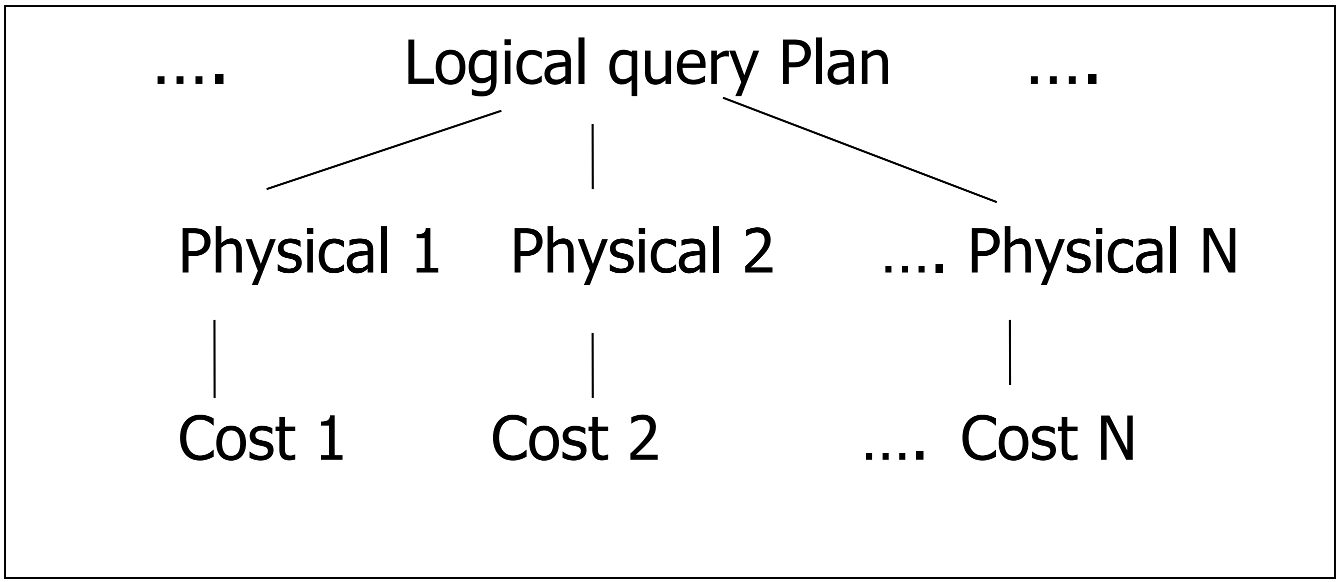
Overview of Query Execution

SQL Query ➔ Compile ➔ Optimize ➔ Execute



Step 6: Estimate the Cost

- This is done for each physical plan



Select the cheapest to execute…

Overview of Query Execution

SQL Query ➔ Compile ➔ Optimize ➔ Execute