CS848 Fall 2025: Algorithmic Aspects of Query Processing

Traditional Query Processing

Xiao Hu

Sep 8, 2025

Agenda

- First class: Introduction
- This class: Traditional query processing
  - Relational Algebra
  - Pairwise Framework
  - Yannakakis algorithm
  - Extensions

Pointers to Related Work

- Abiteboul, Hull, Vianu. Foundations of Databases. Addison Wesley, 1995. http://webdam.inria.fr/Alice/ , Ch 6.4: Acyclic joins, semi-join reduction, Yannakakis, GYO algorithm.
- Yannakakis. Algorithms for acyclic database schemes. VLDB 1981 https://dl.acm.org/doi/10.5555/1286831.1286840
- Bernstein, Chiu. Using semi-joins to solve relational queries. JACM 1981. https://doi.org/10.1145/322234.322238
- Bernstein, Goodman. Power of natural semi-joins. SIAM J. 1981. https://doi.org/10.1137/0210059
- Beeri, Fagin, Maier, Yannakakis. On the desirability of acyclic database schemes. 1983 https://doi.org/10.1145/2402.322389

Recap on Natural Join

- Input: Two relations with common attributes
- Output: All pairs of tuples with the same value on the common attribute

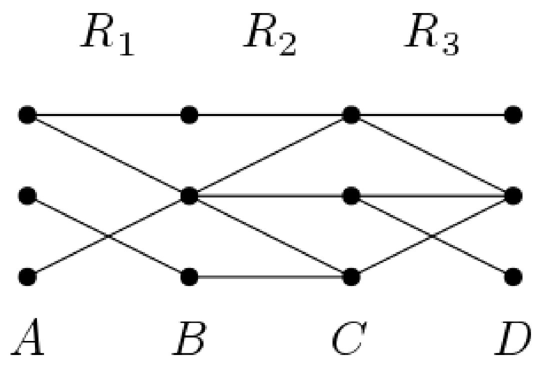
- Nested-loop join
  - $O(|R| \cdot |S|)$
- Sort-merge join
  - $O(|R| + |S| + |R \bowtie S|)$ ignoring the log factor
- Hash join

R	A	B
	1	2
	4	2
	6	6
	7	7
	1	7
	1	6

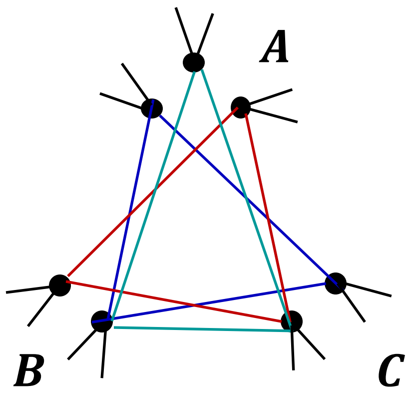
S	B	C
	2	3
	2	5
	6	4
	8	9

$[R \bowtie S]$	A	B	C
	1	2	3
	1	2	5
	4	2	3
	4	2	5
	6	6	4
	1	6	4

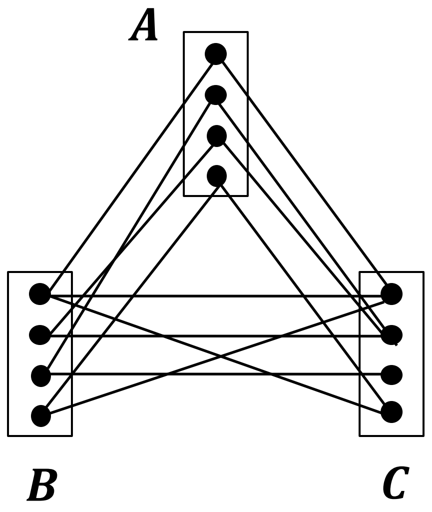
Multi-way Joins as a Hypergraph



- Path join: $R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(C,D)$
  - Results = all paths from A to D
- Triangle join: C_3: $R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(A,C)$
  - Results = all triangles



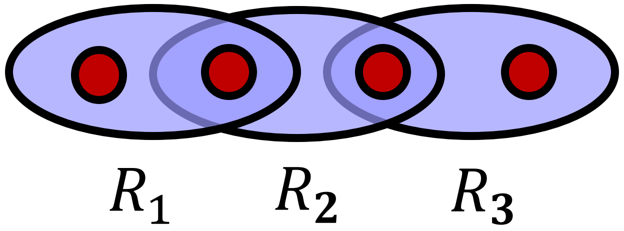
- Star join : $R_0(A,B,C) \bowtie R_1(A,D_1) \bowtie R_2(B,D_2) \bowtie R_3(C,D_3)$



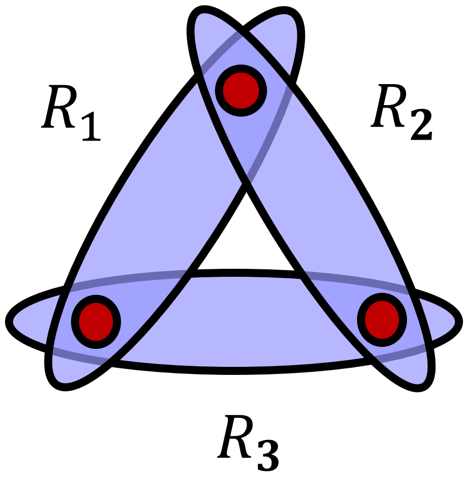
vertices: values
hyperedges: tuples

Query Pattern

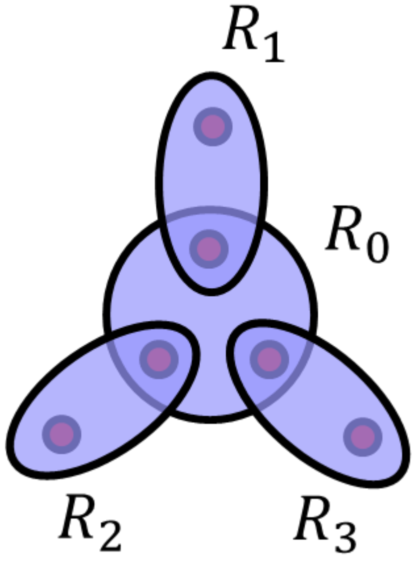
- Path join: $R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(C,D)$



- Triangle join: $R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(A,C)$



- Star join: $R_0(A,B,C) \bowtie R_1(A,D_1) \bowtie R_2(B,D_2) \bowtie R_3(C,D_3)$



vertices: attributes
hyperedges: relations

The Logic Perspective

- $R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(A,C) \equiv R_1(A,B) \land R_2(B,C) \land R_3(A,C)$
  - $R_1(A,B)$: "relation" or "atom"; A, B: "attributes" or "variables"
- 3SAT as a join:
  - $\psi = (x_1 \lor \overline{x_2} \lor x_3) \land (x_2 \lor \overline{x_3} \lor x_4) \land (\overline{x_1} \lor x_3 \lor \overline{x_4})$
  - Satisfiable truth assignments of ψ are the join results of $R_1(x_1,x_2,x_3) \bowtie R_2(x_2,x_3,x_4) \bowtie R_3(x_1,x_3,x_4)$ on the instance:
    $R_1 = \{(x_1,x_2,x_3) \mid x_1 \lor \overline{x_2} \lor x_3 = 1\}$
    $R_2 = \{(x_2,x_3,x_4) \mid x_2 \lor \overline{x_3} \lor x_4 = 1\}$
    $R_3 = \{(x_1,x_3,x_4) \mid \overline{x_1} \lor x_3 \lor \overline{x_4} = 1\}$

	Database Theory	Logic
Query size	Constant (usually)	Large
Domain	Infinite	$\{0,1\}$
Instance size	Large	Constant

Graph Pattern Matching as Join

- Store all edges in a relation $E(x, y)$
  - Note: every edge $\{u, v\}$ should be stored twice: $(u, v), (v, u)$
- Find all triangles (each 3 times):
  - $E(A, B) \bowtie E(B, C) \bowtie E(C, A)$
  - Also called a "self-join"
- Find all length-3 paths:
  - $E(A, B) \bowtie E(B, C) \bowtie E(C, D)$
  - $\sigma_{A \neq C \wedge A \neq D \wedge B \neq D}\big(E(A, B) \bowtie E(B, C) \bowtie E(C, D)\big)$
  - Allowing inequalities pushes us to full relational algebra (first-order logic)!

Queries studied

Select + Projection + Join + Union + Difference + Aggregation (SPJUDA)

- Relations:
  - *Movies(Title, Director, Actor)*
  - *Location(Theater, Address, Phone Number)*
  - *Pariscope(Theater, Title, Schedule)*
- Selection: Find all showings after 22:00
  - $\sigma_{Schedule>22:00} Pariscope$
- Projection: Find all directors directing a movie shown after 22:00
  - $\pi_{Director}(Movies \bowtie \sigma_{Schedule>22:00} Pariscope)$
- Aggregation: How many late-night movies has each director directed?
  - $\pi_{Director,\, COUNT(*)}(Movies \bowtie \pi_{Title} \sigma_{Schedule>22:00} Pariscope)$

Problems studied

- Given a **SPJUDA** query Q
- Listing: Return Q
- Enumeration: After some pre-processing, enumerate tuples in Q one by one.
- Boolean: Is $Q = \emptyset$?
- Counting: Find $|Q|$
- Approximate counting: $(1 + \epsilon)$-approximation of $|Q|$
- Sampling: return a uniform sample from Q

- We first focus on join queries in the next two lectures!

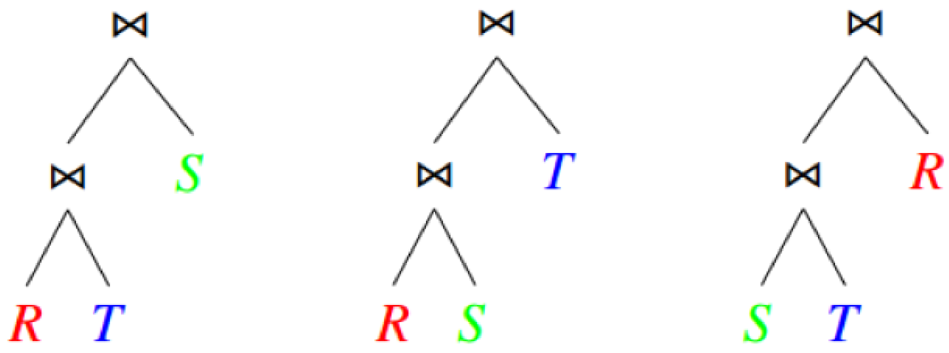
Data Complexity

- Computing join query is NP-hard in terms of both query size and data size [Chandra-Merlin, STOC'77]

- We focus on the data complexity [Vardi, STOC'82]
  - Input size $N =$ # tuples in the database
  - Output size OUT = # query results

Query Processing in Traditional Database Systems

- **Join Query:** A highly optimized version of Pairwise Framework
  - A query plan is a binary tree
  - Estimate the cost of each query plan using data statistics
  - Pick the one with the minimum cost



$$Q := R(A,B) \bowtie S(B,C) \bowtie T(C,D)$$

- Join is commutative and associative
  - $R \bowtie S = S \bowtie R$
  - $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

Theoretical Insights

R

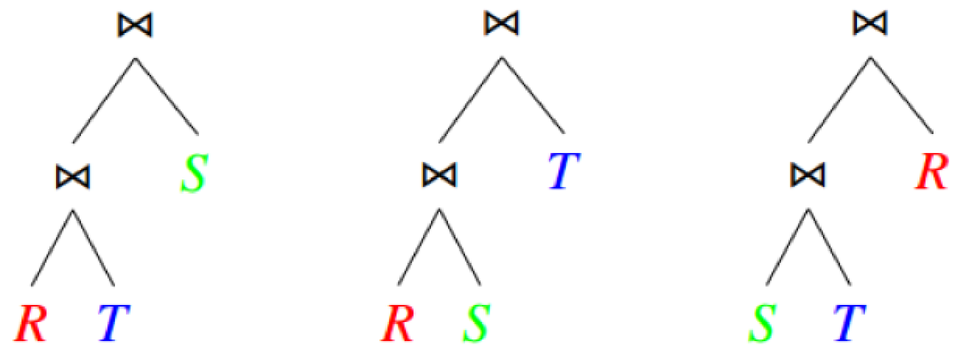
A	B
a_1	b_1
a_2	b_1
a_1	b_2
a_2	b_3
a_1	b_4

S

B	C
b_1	c_3
b_1	c_5
b_3	c_3
b_5	c_5
b_6	c_6

T

C	D
c_1	d_1
c_2	d_2
c_5	d_1
c_6	d_2
c_6	d_3



$$Q := R(A,B) \bowtie S(B,C) \bowtie T(C,D)$$

Semi-Join $R(A,B) \ltimes S(B,C)$

- $R \ltimes S = \pi_{A,B}(R \bowtie S)$: all tuples in R that can be joined with at least one tuple in S
- Law of semi-joins: $R \bowtie S = (R \ltimes S) \bowtie S$
- $O(|R| + |S|)$ ignoring log-factors
- How to use semi-join to remove dangling tuples those won't participate in any join result?

R

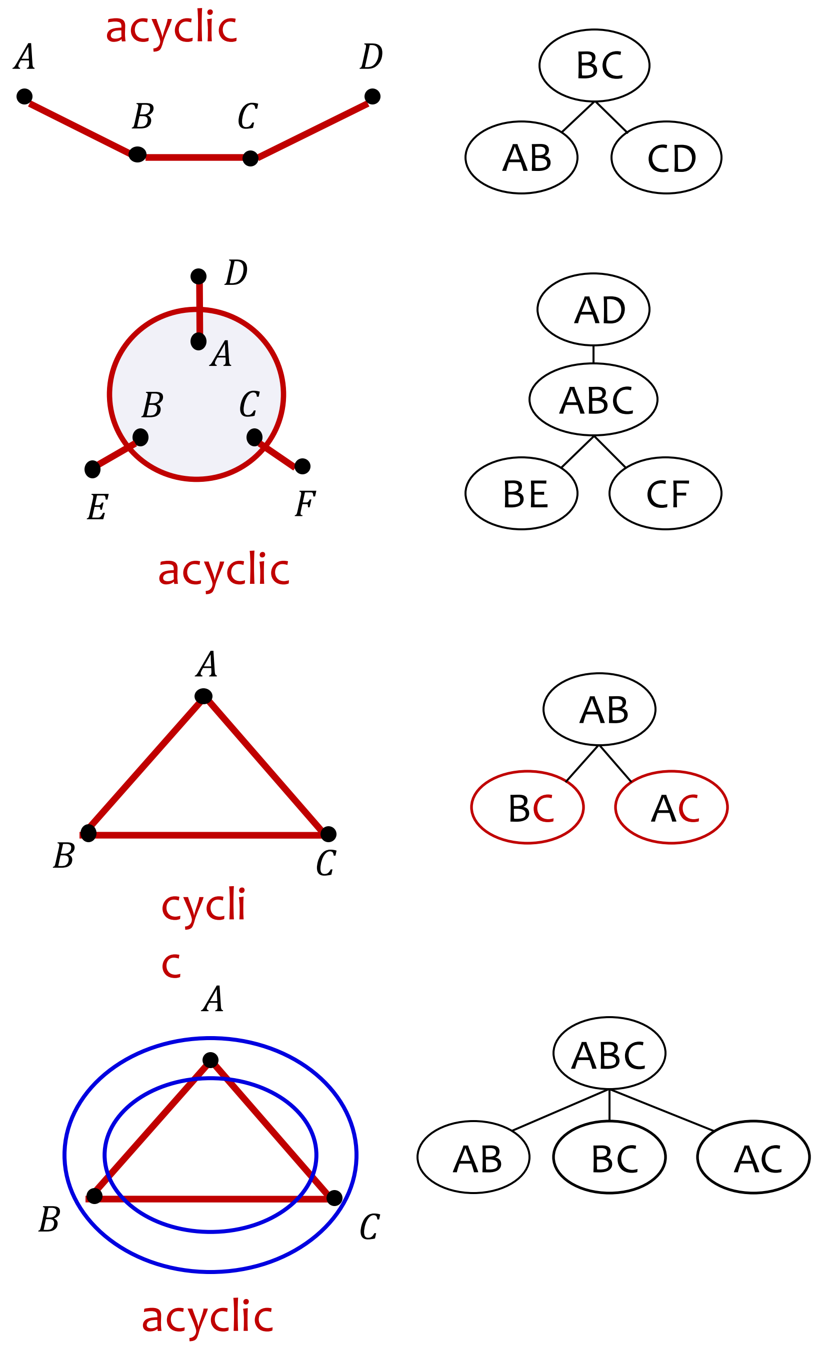
A	B
a_1	b_1
a_2	b_1
~~a_1~~	~~b_2~~
a_2	b_3
~~a_1~~	~~b_4~~

S

B	C
b_1	c_3
b_1	c_5
b_3	c_3
b_5	c_5
b_6	c_6

Acyclic Join

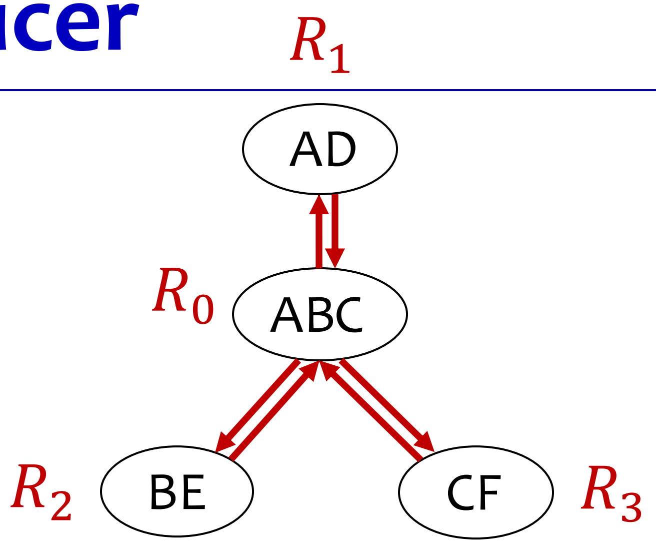
- A join query $Q = (V, E)$ is acyclic if it has a join tree T such that
  - one-to-one correspondence between nodes in T with the relations in E;
  - for any attribute $A \in V$, all nodes containing A form a connected subtree.



Yannakakis Algorithm: Semi-Join Reducer

Take an arbitrary join tree

- In a bottom-up phase:
  - pick a non-visited node e (with its parent e')
  - update $R_{e'}$ with $R_{e'} \ltimes R_e$
- In a top-down phase:
  - pick a node e
  - For each child e' of e, update $R_{e'}$ with $R_{e'} \ltimes R_e$
- Data Complexity: $O(N)$ ignoring log-factors



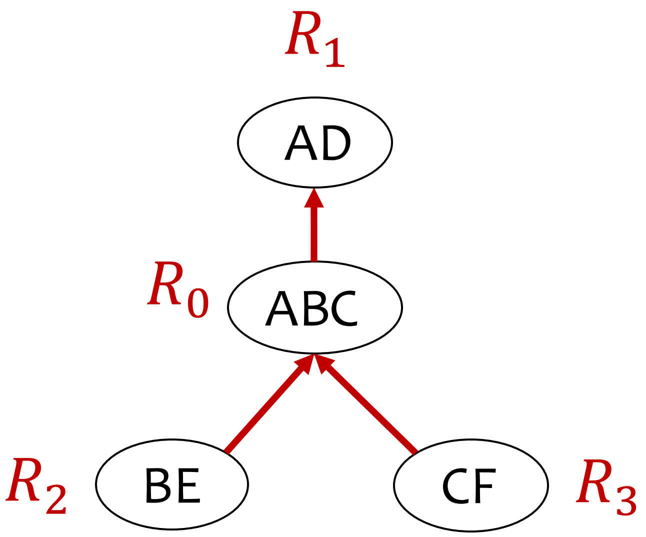
$R_0 := R_0 \ltimes R_2$
$R_0 := R_0 \ltimes R_3$
$R_1 := R_1 \ltimes R_0$

$R_0 := R_0 \ltimes R_1$
$R_2 := R_2 \ltimes R_0$
$R_3 := R_3 \ltimes R_0$

Yannakakis Algorithm: Pairwise Framework

Take an arbitrary join tree

- In a bottom-up phase:
  - pick a non-visited node e (with its parent e')
  - update $R_{e'}$ with $R_{e'} \bowtie R_e$
- Output R_r for the root node r
- The intermediate join size is bounded by $O(OUT)$
- Data complexity: $O(OUT)$



$$R_0 := R_0 \bowtie R_2$$
$$R_0 := R_0 \bowtie R_3$$
$$R_1 := R_1 \bowtie R_0$$

Recap Presentation Guide

- Every class will have **1 paper presentation.**
- Any change in presentation schedule needs to be announced **2 weeks before the presentation.**
- Each presentation should be prepared for **50 mins + 20 mins**
- The presenter should expect the audience to have general knowledge about the field but not expects – **sufficient background** is needed in the beginning!
- Please start preparation as early as possible!