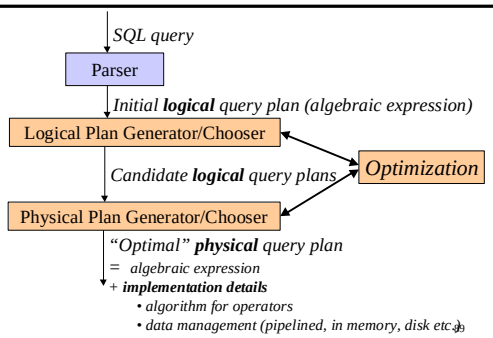
Query Processing



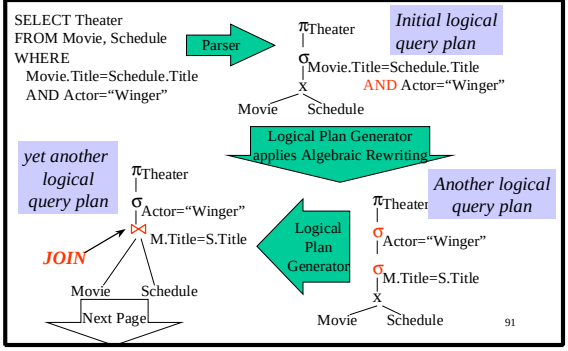
89

Query Processor

- Basic task: turn user queries (and updates) into a sequence of operations (a *plan*) on the database
  => *from high level queries to low level commands*
- Many choices:
  - *Which of the **algebraically equivalent** forms of a query will lead to the most efficient algorithm?*
  - *For each algebraic operator which **algorithm** should we use to run the operator?*
  - *How should the operations pass data from one to the other? (eg, main memory buffers, disk buffers)*

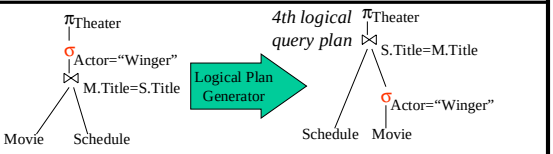
90

The Journey of a Query (Revisited)

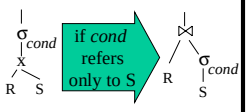


91

The Journey of a Query (Revisited)

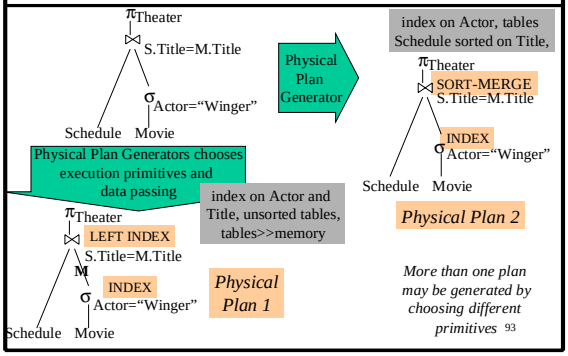


- 4 logical query plans created
- ***algebraic rewritings*** were used to produce candidate logical query plans
- the last one wins
- multiple logical plans may "win" eventually (and be considered subsequently)



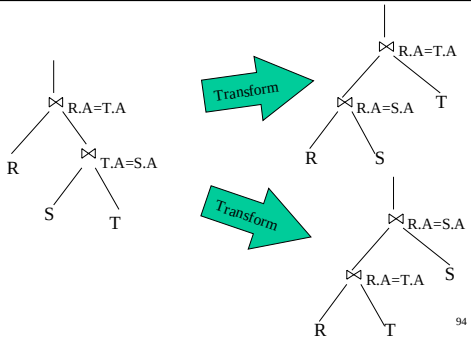
92

The Journey Continues at the Physical Plan Generator



93

More Than One Plan May be Generated and Considered for Evaluation



94

Issues in Query Processing and Optimization

- *Generate Plans*
  - systematically transform expressions
  - employ execution primitives for computing relational algebra operations
- *Estimate Cost of Generated Plans*
  - Statistics
- *"Smart" Search of the Space of Possible Plans*
  - always do the "good" transformations (relational algebra optimization)
  - prune the space (e.g., System R)
- Often the above steps are mixed

95

Bag Semantics of Operators

- SQL semantics for collections of tuples:
  - often ***bags*** (aka ***multisets***) instead of sets.
  - ***Union(R,S)***: a tuple t occurs in the result as many times as the **sum** of occurrences in each of R and S
  - ***Intersection of R and S***: ... the **minimum** number of occurrences in R and S
  - ***Difference of R and S***: ... occurrences in R **minus** occurrences in S
  - ***δ(R)*** converts the **bag** *R* into a **set**
    - SQL's **R UNION S** by default ***δ(R∪S)***
- **Example:** *R={A,B,B}* and *S={C,A,B,C}.*

96

Extended Algebra Operators

- **selection σ_C(R)**: condition C over arithmetic-, string-, and comparison-operators
  ~WHERE in SQL, but C depends only on individual tuples
- **projection π_A(R)**: attribute list A may include
  - x→y (*rename x to y in the output schema*)
  - a+b→x (*output schema contains x column with sum of a and b*)
  - c||d→y (*concatenate c and d as y*)
  - again: look at one tuple at a time
- alternatively: s**pecial purpose operators**:
  - **MULT $_{A,B\to C}$(R)**:
    - for each tuple of R, multiply attribute A with attribute B and put the result in a new attribute named C.
  - **PLUS $_{A,B\to C}$(R), CONCAT $_{A,B\to C}$(R), . . .**

97

Product and Joins

- **product (R×S)**:
  - *r* is *n* times in *R*, *s* is *m* times in *S*, then (*r*,*s*) is ***n*m*** times in the product
  - (qualify attributes if necessary: R.a vs S.a)
- **natural join R⋈S = $\pi_A \sigma_C$(R×S)** where
  - *C equates all common attributes*
  - *A* has all attributes from R and S (but only one copy per equated attribute)
  - (definition of join ~ simple rewriting rule)
- **theta join**
  - arbitrary condition involving attributes

98

Grouping and Aggregation

- **Grouping & Aggregation** in SQL:

```
SELECT selection-list FROM from-list WHERE...
GROUP BY groupby-list
HAVING group-qualification
```

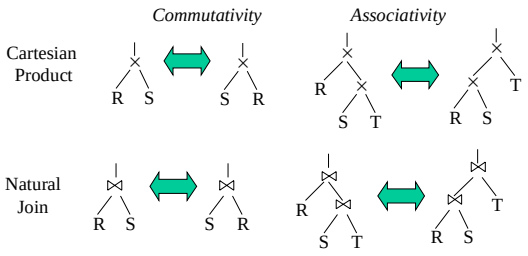
- **Example**: *given* **movie(title, year, actor)**, *find for each actor having more than two movies, the first year of appearance*

```
SELECT actor, MIN(year) as firstYear FROM movie
GROUP BY actor
HAVING COUNT(title) > 2
```

- Every attribute in the *selection-list* must appear in the *groupby-list* ***(Why?)***
- $grp_{actor,\ MIN(year)=>firstYear,\ COUNT(title)=>aux}$ *(movie)*

99

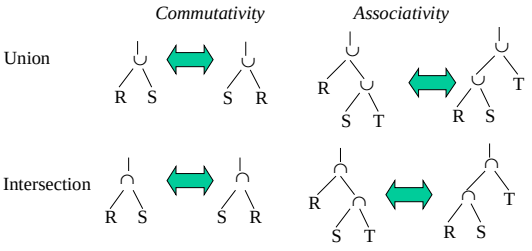
Algebraic Rewritings: Commutativity and Associativity



- *Do the above hold for both sets and bags?*
- *Do commutativity and associativity hold for arbitrary Theta Joins?*

100

Algebraic Rewritings: Commutativity and Associativity



101