Piani di accesso

Risultato dell'analisi e ottimizzazione di una query SQL. Sono alberi di operazioni fisiche con le tabelle di input nelle foglie, che indicano *come* la query deve essere eseguita; tra tutti i piani possibili, vogliamo trovare idealmente quello più efficiente. Per farlo, partiamo dall'albero degli operatori logici, applichiamo trasformazioni (e.g. join al posto di subquery e spostamento delle selezioni prima dei join) e scegliamo per ciascun operatore un algoritmo che lo implementa.

Operatori fisici

Sono iteratori, oggetti con metodi `open` (inizializzazione, passaggio parametri, `open` dei figli), `next`, `isDone`, `reset`, `close`.

R	**TableScan**(R)
	IndexScan(R, I) segue l'ordine dell'indice
	SortScan(R, $\{A_1, \ldots, A_n\}$)
$\pi_{A_1,\ldots,A_n}$	**Project**(O, $\{A_1, \ldots, A_n\}$)
	Distinct(O) elimina i duplicati, O già ordinato
σ_ψ	**Filter**(O, ψ)
	IndexFilter(R, I, ψ) più efficiente e risultato ordinato
$\tau_{A_1,\ldots,A_n}$	**Sort**(O, $\{A_1, \ldots, A_n\}$)
$_{A_1,\ldots,A_n}\gamma_{f_1,\ldots,f_m}$	**GroupBy**(O, $\{A_1, \ldots, A_n\}$, $\{f_1, \ldots, f_m\}$)
$\bowtie_\psi$	**NestedLoop**(O_E, O_I, ψ)
	PageNestedLoop(O_E, O_I, ψ)
	IndexNestedLoop(O_E, O_I, ψ) con O_I = IndexFilter(R, I, ψ) (ψ passato da IndexNestedLoop con open), eventualmente preceduto da un Filter
	SortMerge(O_E, O_I, ψ) con O_E e O_I già ordinati

R è una tabella, O il risultato di un'operazione precedente.