**1. SIP**

최적화의 의해 생성된 플랜이 좋더라도 비용이 큰 조인연산들은 여전히 남아있다. 이전조인 혹은 인덱싱스캔이 정렬로 주어져 조인입력들은 조인-속성 값에 의해 정렬되기 때문에 일반적으로 머지조인이 가장 빠른 알고리즘이지만 해쉬조인이 더 선호된다. 비 로컬 정보를 고려함으로써 하나의 복잡한 트리내부의 싱글조인의 비용을 줄일 수 있는 다양한 방법들이 있다. 이러한 방법들은 대략적으로 SIP로 추론되고, 실행 계획 트리 구조를 절단하는 방법으로 정보를 운영자에게 전달한다. SIP 전략은 컴파일타임과 런타임 두가지 방법들로 구성되는데 컴파일타임은 세미조인 감속기와 변환기로 구성되어진다. 세미조인은 조인 속성 값들을 트리내부의 다른 피연산자로 보내고 조인 속성 값들은 런타임의 필터로 제공된다. 이 원칙을 계단식하여 세미 조인 전체 프로그램을 구축할 수 있습니다. rewriting은 중첩된 SQL블록이나 뷰에 대한 query를 포함한 query블록 전체에 해당 패러다임을 적용 및 확장시킨다. 이는 비용과 관련하여 컴파일타임에 미리 계획되어질 필요가 있다. 일단 하나의 세미조인이 실행계획안에 생성되어지면 비 선택적이더라도 런타임에 실행되어진다.

반면에, 적응형 런타임 메더스는 영향을 받는 연산자가 실제로 실행될 때까지 실행되는 것은 미루어진다. 적응형 query 처리는 다른 실행계획구조의 조정과 튜플 당 재 정렬을 조인하는 방법이다. 최근의 연구에서는 컴파일시 비용 모델 실행 시 이익 추정치에 의해 보완되고 SIP 기반의 필터가 쿼리를 실행하는 동안 적응 적으로 활성화 또는 비 활성화되는 같은 SIP 전략에 대처했습니다. 그러나 SIP 필터는 완료된 연산자의 결과에 대해서만 구축되어 결합 (및 그룹화)에 의해서만 사용된다.

**2. U-SIP**

U-SIP는 Ubiquitous Sideways Information Passing의 약어로, 도처에 있는 연산결과를 통해 정보들을 구성한다. 기존의 SIP와 다르게 생성된 연산자가 실행이 완료될 때 필터정보가 전달된다. U-SIP는 개념적으로 모든 연산자들의 사이에 활성화될 수 있고, 비용에 민감한 컴파일타임 최적화와 복잡한 비용측정의 런타임 조정이 필요로 하지 않는다. 읽고 병합할 데이터의 양을 줄이기 위한 방법으로 아래와 같다.

- 다른 연산자에 대한 지식은 가치격차의 정보로부터 잠재적인 이익이 될 수 있고, 이는 주어진 query에 대한 다양한 연결을 기초로 하여 query 컴파일타임에서 드러난다.

- 스캔과 병합하는 과정 속에서 정렬된 정보리스트사이에 중요한 차이에 대한 경량화 된 부기와 해쉬 조인들에 의해 생성된 경량화 도메인 필터들을 통해 오버헤드를 낮게 유지한다.

모든 런타임 메커니즘은 연산자 단위로 완전히 비동기로 동작하는데, 이는 파이프라인 아키텍처 및 실행 엔진의 내재된 멀티 스레딩을 방해하지 않으면서 동적 SIP를 추가하기 위함이다.

2.1 Compile-Time Preparation

U-SIP를 가진 query 연산자 트리는 연산자들 사이에서 정보의 sideways flow를 나타낸다. 이러한 에지들을 추론하는 것은 동등 변수들의 개념이다. 만약 변수들이 가지고 있는 값이 완전히 같다면 두 개의 변수는 동등하다고 할 수 있다(DRW00002ab44478 ).

결과적으로 동등 변수들의 집합을 처리함으로써 변수 DRW00002ab4447a의 동등 클래스 DRW00002ab4447c를 추론할 수 있다.(Figure 4)

|  |
| --- |
| DRW00002ab4447e |

[ Figure 4 : Equivalence variable example ]

2.2 Run-Time Handling of Merge Joins

머지 조인은 조인 변수들의 값들을 오름차순으로 입력 값들을 읽는다. 따라서 모든 머지조인과 연관된 모든 동등변수들은 이러한 변수들의 연결한 가장 큰 현재 변수에 대입될 수 있다. 동등 클래스를 위한 다음 연결로써의 값을 추론할 수 있다.

2.3 Run-Time Handling of Hash Joins

각각의 변수들의 동등 클래스를 위해 내재된 변수 연결을 묘사하는 도메인 필터를 구축하며, 이는 Bloom filte와 range bounds를 포함한다.

|  |
| --- |
| DRW00002ab44480 |

해당 클래스의 모든 변수에 대해 같은 potential domain 및 개별 해시 조인 연산자의 구축 단계에 의해 유지되는 observed domain과 구별한다.

potential domain은 모든 변수를 DRW00002ab44482로 초기화하는 반면에, observed domain은 DRW00002ab44484로 초기화 한다. 특정 변수의 범위의 부재를 테스트하기 위해 Bloom filters를 사용하고, 임의의 해시 함수를 사용하는 것이 아닌 0 ~ m-1 사이에 x를 매칭하기 위해 function h(x)를 사용한다.

|  |
| --- |
| ((h(x) < h(y) ∧ D[h(x)] = D[h(y)] = 1  ∧ ∀i : h(x) < i < h(y) : D[i] = 0) ∨  (h(x) > h(y) ∧ D[h(x)] = D[h(y)] = 1  ∧ ∀i : i > h(x) ∨ i < h(y) : D[i] = 0))  ⇒ (y < x ∨ |y − x| > c(h(y) h(x))) |

양의 상수 C와 modulo-m는 difference DRW00002ab44486다음과 같이 정의한다.

|  |
| --- |
| DRW00002ab44488 |