Tech Info(vol.7)

소개

이 글은 Informix 의 내장 프로시저 사용에 관한 지침서입니다. 이 글의 내용은 Informix 제품 설명서, 교육 자료 그리고 Informix 기술 지원 엔지니어, 컨설턴트 및 시스템 엔지니어들의 실질적인 경험 등 여러 가지 다양한 자료로 구성되었습니다.

내장 프로시저는 클라이언트 응용 프로그램으로부터 특정 SQL 기능을 발췌하여 데이터베이스 서버로 읽어 들이기 위한 유용한 도구가 될 수 있습니다. 이 문서에서는 내장 프로시저 사용에 따른 장점, 단점 및 주의해야 할 사항들을 살펴 보고 이를 통해 Informix 의 구조를 최대한 효과적으로 이용하는 방법을 설명합니다.

일반적으로 내장 프로시저는 SQL 이외의 처리가 거의 없는, SQL 이 집중된 루틴에서 가장 잘 사용할 수 있습니다. 계산 처리를 집중적으로 하는 루틴에 대해서는 ESQL/C 와 같은 컴파일된 언어를 사용하는 것이 가장 적합합니다. 그러나 실제로 테스트해보지 않고 내장 프로시저가 성능에 가져올 전반적인 영향을 규정지을 수는 없습니다. 실질적인 성능은 하드웨어 구성, 운영 체제, 네트워크 대역폭과 대기 시간 및 전체 응용 프로그램의 설계에 의해 좌우됩니다.

이 문서에서 제공하는 정보는 Informix Dynamic Server 7.x 버전을 대상으로 하고 있지만, 대부분의 지침 사항은 Informix Online 5.0x 와 Informix Universal Data Option 9.0 에도 적용됩니다.

내장 프로시저의 장점

- 응용 프로그램 간에 코드를 공유하게 함으로써 프로그램의 복잡성을 줄여 주고 이를 통해 프로그램 유지 비용을 절감할 수 있습니다.

  내장 프로시저를 사용하여 공통으로 액세스하는 기능을 서버에 둠으로써 프로그램의 복잡성을 줄일 수 있습니다. 이런 방식으로, 다른 프로그램과 사용자 인터페이스에서도 이 공통 코드를 공유할 수 있습니다.

- SQL 집중적인 루틴에 유용합니다. (하나 이상의 SQL 구문)

한 내장 프로시저에서 여러 개의 SQL 구문을 실행하면 어느 정도 성능 향상을 얻을 수 있습니다. 그 이유는 프로시저가 데이터베이스 엔진에 직접 상주하므로, 특히 광역 네트워크(WAN)에서 클라이언트/서버간의 통신량이 감소하기 때문입니다. 또한 프로시저가 의사 코드(p-code)로 캐시에 저장되어 있어 구문 해석을 하거나 다시 최적화할 필요가 없으므로, 구문 해석과 최적화 역시 감소됩니다.

- 별도의 보안 레벨을 추가하여 업무 규칙을 강화할 수 있습니다.

  기반 테이블에 대한 모든 액세스를 취소함으로써 사용자들이 테이블에 직접 액세스하는 대신에 내장 프로시저를 사용하도록 규정할 수 있습니다. 내장 프로시저에 대한 액세스는 실행 허가에 대한 권한 부여 및 취소(Grand and Revoke)에 의해 제공됩니다. 내장 프로시저의 실행 허가는 사용자가 아니라 작성자에 의해 부여됩니다. 이렇게 함으로써 데이터베이스 관리자(DBA)는 일반적으로는 사용자의 액세스가 금지되어 있는 테이블의 행을 갱신/삽입/삭제하는데 대해 엄격한 규칙을 설정할 수 있습니다. 그렇게 되면 프로그래머는 테이블에 대한 작업을 수행할 때 내장 프로시저를 사용해야 할 것입니다.

- 서버 측에 복잡한 업무 로직을 구현하기 위해 여러 가지 트리거들과 결합해야 할 경우에 유용합니다.
- 반복을 요하는 문제 해결에 유용합니다.

  내장 프로시저는 청구서를 대량으로 발행하는 것과 같은 반복적인 작업에 유용합니다. 이것은 4GL 에서 커서로 잘 수행하지 못하는 유형의 기능입니다.

- 트랜잭션 모니터에서 서버로 연결하는 횟수를 줄이는 데 사용할 수 있습니다.

  Tuxedo 등의 트랜잭션 모니터와 함께 사용하는 경우에 내장 프로시저를 사용하여 서버에 대한 트랜잭션을 일괄 처리할 수 있으므로 서버에 대한 전체 연결 횟수를 줄일 수 있습니다.

내장 프로시저의 단점

SQL 에 없는 실행을 할 경우 내장 프로시저에는 추가 비용이 들어 갑니다.

- 내장 프로시저는 처음 호출 시 디스크로부터 검색됩니다.

  내장 프로시저는 데이터베이스의 시스템 카탈로그에 ASCII 형식으로 저장되어 있습니다. 내장 프로시저는 처음 호출 시 디스크로부터 읽어져서,

ASCII 형식에서 바이너리(의사 코드) 형식으로 변환되며, 내장 프로시저 캐시에 놓이게 됩니다. 내장 프로시저는 *가장 최근에 사용한 것을 기준으로* 하여 캐시에 보관됩니다.

- 프로시저에 필요한 열, 테이블 또는 색인이 변경되는 경우에는 프로시저를 다시 최적화해야 합니다. 기본적으로, 프로시저를 호출할 때마다 변경사항 여부를 점검합니다.

  내장 프로시저에서 사용하는 기반 테이블 또는 색인이 변경되었을 경우 언제나 내장 프로시저를 다시 최적화해야 합니다. 여기에는 통계 자료의 갱신, "권한 부여" 또는 "취소" 작업 등이 포함되는데 그 이유는 이러한 작업들이 테이블의 내부 버전을 변경시키기 때문입니다. SET OPTIMIZATION LOW 를 사용하여 이러한 최적화 작업을 최소화할 수 있습니다.

- 내장 프로시저를 포함하는 구문은 병렬적으로 실행할 수 없습니다.

  그러나 프로시저 내의 구문은 프로시저를 작성할 때 PDQPRIORITY 의 설정 여부에 따라 병렬적으로 실행할 수 있습니다.

- 반환되는 데이터의 양과 유형에 따라 성능이 떨어질 수 있습니다.

  반환되는 데이터의 행과 열의 수량이 많은 경우, 내장 프로시저는 ESQL/C 의 유사한 구문에 비해 비효과적일 수 있습니다. 왜냐하면 내장 프로시저에서 데이터를 반환하기에 앞서, 내부 메모리 스택에 있는 표현식으로 데이터를 이동시켜야 하기 때문입니다. 이와 반대로 ESQL/C 응용 프로그램은 직접 데이터 필드를 참조하므로 이러한 오버헤드를 피할 수 있습니다.

- 인수의 수량과 유형에 의해 성능이 좌우됩니다.

  배열 데이터 유형을 지원하지 않기 때문에, 내장 프로시저에 많은 수의 인수를 넘겨주고 구문 분석을 할 경우 ESQL/C 와 같이 배열 데이터를 지원하는 컴파일된 언어를 사용하는 것에 비해 덜 효과적일 수 있습니다.

- 내장 프로시저는 서버 측에서의 추가적인 처리를 발생시키기 때문에 서버 측에 추가적인 하드웨어 자원(CPU, 메모리 등)을 필요로 하는 경우도 있습니다.

또한, 내장 프로시저에는 다음과 같은 제한이 있습니다.

- 프로시저 코드와 그 전역 변수들을 합한 크기가 프로시저당 64K로 제한됩니다.

  64K 를 초과하는 프로시저는 분리시켜서 클라이언트 응용 프로그램에서

차례로 호출해야 합니다. 64K 가 넘는 프로시저의 경우 ESQL/C 와 같은 컴파일된 언어를 사용하는 것이 바람직합니다.

- 동적인 SQL을 수행하지 못합니다.

  내장 프로시저는 컴파일 시 구문 분석과 최적화가 이루어지므로 동적인 SQL 을 사용하지 못합니다.

- 내장 프로시저 내에서는 DDL(데이터베이스) 또는 UNLOAD 구문을 수행하지 못합니다.
- 반환 세트의 크기가 제한되어 있습니다(플랫폼에 따라 다름).
- 디버깅이 복잡하고 파일 시스템을 근거로 하며 디버깅 툴들이 대화식이 아닐 수 있습니다.
- SPL 언어의 경우에는 시간이나 문자열 처리의 지원에 제한이 있습니다.

  Informix Dynamic Server 7.3 버전에 소개된 몇몇 새로운 SQL 기능은 문자열 처리를 크게 개선시켜 줍니다.

- 내장 프로시저 캐시의 크기는 제한되어 있으며, 사용 중인 프로시저의 수량을 수용하기 위해 언제든지 조정될 수 있어야 합니다.

  기본 내장 프로시저 캐시는 최대 50 개의 내장 프로시저를 유지하도록 설계되었습니다. 프로시저가 50 개가 넘는 응용 프로그램은 내장 프로시저 캐시의 크기를 조정하여(아래의 PC_POOLSIZE 및 PC_HASHSIZE 를 참조), 비용이 많이 드는 프로시저 스왑이나 디스크 다시 읽기가 발생하지 않도록 해야 합니다.

지침 사항

- 변경하지 않는 프로시저를 가진 정적인 생산 환경에서 내장 프로시저를 호출하는 경우에는 SET OPTIMIZATION LOW를 사용하십시오.

  내장 프로시저를 호출하는 경우에는 그 프로시저를 실행할 때마다 다시 최적화해야 할 필요가 있는지를 확인하지 않도록 응용 프로그램을 반드시 SET OPTIMIZATION LOW 로 설정해야 합니다. 다른 모든 작업에 대해서는 반드시 기본값인 SET OPTIMIZATION HIGH 로 다시 설정해야 합니다. 최적화 정도가 낮은 프로시저를 실행하는데 따르는 위험은, 다른 사용자가 시스템 카탈로그 테이블에 대해 변경한 사항들을 프로시저에서 인식하지 못할 가능성이 있다는 것입니다.

- 절대적으로 필요한 경우가 아니라면 서버 측의 기능들을 구현하기 위해 내장 프로시저를

사용해서는 안됩니다.

내장 프로시저는 간단한 업무 로직을 가진 다중 SQL 문을 실행하는데 가장 적합하도록 설계되었습니다. 내장 프로시저는 의사 코드를 이용하여 실행하기 때문에, 계산 집중적인 기능은 ESQL/C 와 같은 컴파일된 언어로 구현하는 것이 가장 바람직합니다.

- 다른 내장 프로시저에서 한 내장 프로시저를 여러 번 호출하는 것을 피해야 합니다.

  내장 프로시저를 호출하는데 따른 오버헤드는 낮은 편이지만 다른 프로시저의 WHILE 또는 FOREACH 루프 내에서 내장 프로시저를 여러 번 호출하는 경우에는 성능 저하가 심해집니다. 가능하다면, 호출되는 프로시저의 코드를 호출하는 프로시저 내에 직접 두는 것이 좋습니다. 반복 호출은 지원하지만 다른 프로시저에 대한 중첩 호출(nested calls)은 시스템 자원에 대한 충돌을 유발할 수 있습니다.

- 루프 내에서 내장 프로시저를 호출해서는 안됩니다.

  루프 내에서 내장 프로시저를 호출해서는 안됩니다. 프로시저는 동적으로 업데이트되므로, 읽기 도중에 변경되는 것을 막기 위해 프로시저를 읽는 동안 전역 프로시저 캐시는 뮤텍스 잠금으로 잠깁니다. 심하게 루프되는 동안에 내장 프로시저를 호출하면 뮤텍스 잠금에 대한 충돌을 발생시켜 성능을 단일 스레드 수준으로 저하시킬 수가 있습니다.

- 데이터베이스 서버의 병렬 처리 기능을 이용하고자 하는 경우에는 데이터 처리 구문(DML) 내에서 프로시저 호출의 사용을 제한하십시오.

  내장 프로시저 호출을 포함한 모든 구문은 병렬로 실행되지 않기 때문에 데이터 처리 구문에서 내장 프로시저 호출을 사용하는 것을 삼가해야 합니다.

- 다른 응용 프로그램 코드를 관리하듯이 내장 프로시저를 관리하십시오.

  내장 프로시저는 데이터베이스에 저장되기 때문에 데이터베이스가 삭제될 때 함께 삭제됩니다. 따라서 내장 프로시저 개발자는 표준 버전 제어와 메이크파일(makefiles)을 이용하여 응용 프로그램 코드를 관리하듯이 내장 프로시저를 관리해야 합니다. dbschema 유틸리티를 사용하여 데이터베이스에 저장되어 있는 내장 프로시저의 최신 버전을 알아볼 수 있습니다.

작업 시 고려해야 할 사항

- 문서화되지 않은 onconfig 매개변수인 PC_HASHSIE와 PC_POOLSIZE를 이용하여 내장 프로시저 캐시의 크기를 조정해야 합니다.

  PC_HASHSIE - 해시 버킷(hash bucket)의 개수(기본값은 31 이며, 솟수이어야 합니다.)

  PC_POOLSIZE - 엔트리의 최대 개수(기본값 50)

  전역 내장 프로시저 캐시는 문서화되지 않은 명령어인 Onstat - g prc 로 모니터할 수 있습니다.

- 빈번하게 사용되는 다른 데이터 테이블을 조정하듯이 내장 프로시저와 시스템 카탈로그 테이블을 조정하십시오.

  내장 프로시저는 시스템 카탈로그 내에 저장되어 있기 때문에 빈번하게 사용하면 시스템 카탈로그 테이블이 급속히 증가할 수 있습니다. 내장 프로시저를 이용하는 시스템의 성능은 빈번하게 사용되는 데이터 테이블을 조정하듯이 핵심적인 시스템 카탈로그를 조정함으로써 좋아질 수 있습니다. 즉, (a) 내장 프로시저 시스템 카탈로그 테이블에 대한 확장을 8 개 이하로 유지합니다. (b)가능하다면 시스템 카탈로그를 별도 장치로 분리하여 I/O 충돌을 줄입니다. (c) 하나의 DB 영역 내에서 개별적인 여러 물리적 장치에 걸쳐 내장 프로시저 시스템 카탈로그를 분산시킵니다.

- 테이블 특성이 바뀌면 통계를 업데이트하십시오.

  통계 업데이트는 내장 프로시저에서 SQL 을 다시 최적화하는 데에 영향을 미칩니다.

맺음말

내장 프로시저는 여러 프로그램에서 액세스하는 업무 규칙을 구현하기 위해 고안되었으며, 적절히 사용하는 경우 코드 유지 필요성을 감소시키고 성능을 개선시킬 수 있습니다.

내장 프로시저는 자주 액세스되는 기능을 서버로 이동시킴으로써 프로그램의 복잡성을 감소시킵니다. 그러나 복합 서버에 기반을 둔 응용 프로그램을 구현하기 위해 사용해서는 안됩니다. SPL 은 n 티어 응용 프로그램 개발을

가지고 있을 경우 Informix Uneversal Data Option 은 사용자 정의 기능을 서버 내에 직접 삽입할 수 있는 기능을 제공합니다. 이러한 기능은 C, C++ 또는 Java 등의 컴파일된 언어로도 작성할 수 있을 것입니다.

결론적으로 내장 프로시저를 이용하는 경우에는 서버 하드웨어 용량이 추가로 필요할 수도 있다는 것을 명심하십시오. 내장 프로시저는 CPU 와 공유 메모리 자원을 사용하며 통신 트래픽을 발생시킵니다. 하드웨어 사양에서 이러한 것들을 제대로 지원하지 않으면 엔진 성능에 영향을 미칠 수 있습니다.