



Informatica®  
10.4.0

# 성능 조정 가이드

Informatica 성능 조정 가이드  
10.4.0  
2019년12월

© 저작권 Informatica LLC 2009, 2020

이 소프트웨어와 설명서는 사용 및 공개에 대한 제한 사항이 포함되어 있는 별도의 사용권 계약에 따라서만 제공됩니다. 본 문서의 어떤 부분도 Informatica LLC의 사전 통지 없이 어떠한 형태나 수단(전자적, 사진 복사, 녹음 등)으로 복제되거나 전송될 수 없습니다.

Informatica 및 Informatica 로고는 미국과 전 세계 여러 관할 국가에서 Informatica LLC의 상표 또는 등록 상표입니다. Informatica 상표의 현재 목록은 <https://www.informatica.com/trademarks.html> 웹에서 확인할 수 있습니다. 다른 회사 및 제품명은 해당 소유자의 상표 또는 등록 상표일 수 있습니다.

미국 정부 권한. 미국 정부 고객에게 제공되는 프로그램, 소프트웨어, 데이터베이스, 관련 문서 및 기술 데이터는 해당하는 연방 입수 규정 및 기관별 보안 규정에 따라 "상용 컴퓨터 소프트웨어" 또는 "상용 기술 데이터"입니다. 따라서 사용, 복제, 공개, 수정 및 조정은 해당하는 정부 계약에 규정된 제한 사항 및 라이선스 조건을 따르며, 정부 계약 조건에 의해 적용 가능한 한도 내에서, FAR 52.227-19, 상용 소프트웨어 라이선스에 규정된 추가 권한이 적용됩니다.

이 소프트웨어 및/또는 설명서의 일부에는 타사의 저작권이 적용될 수 있습니다. 필요한 타사 고지 사항은 제품에 포함되어 있습니다.

이 설명서의 정보는 예고 없이 변경될 수 있습니다. 이 문서에서 문제가 발견되는 경우 [infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com)으로 보고해 주십시오.

Informatica 제품은 제품이 제공될 당시의 계약 조건에 따라 보증됩니다. Informatica는 상품성과 특정 목적에의 적합성에 대한 보증 그리고 비침해에 대한 보증 또는 조건을 포함하여 어떠한 종류의 명시적이거나 묵시적인 보증 없이 이 문서의 정보를 "있는 그대로" 제공합니다.

발행 날짜: 2020-02-04

# 목차

<b>서문</b>	<b>7</b>
Informatica 리소스	7
Informatica 네트워크	7
Informatica 기술 자료	7
Informatica 설명서	7
Informatica Product Availability Matrix	8
Informatica Velocity	8
Informatica Marketplace	8
Informatica 글로벌 고객 지원 센터	8
<b>장 1: 성능 조정 개요</b>	<b>9</b>
성능 조정 개요	9
성능 조정 프로세스	9
대상 병목 현상	10
소스 병목 현상	10
매핑 병목 현상	11
컴퓨터 시스템 병목 현상	11
Windows의 시스템 병목 현상 식별	11
UNIX의 시스템 병목 현상 식별	11
런타임 병목 현상	12
SQL 데이터 서비스 최적화 병목 현상	12
웹 서비스 최적화 병목 현상	12
연결 병목 현상	13
<b>장 2: 대상 최적화</b>	<b>14</b>
대상 최적화 개요	14
플랫 파일 대상 최적화	14
데이터베이스 검사점 간격	15
대량 로드	15
데이터베이스 대상 최적화	15
<b>장 3: 소스 최적화</b>	<b>17</b>
소스 최적화 개요	17
플랫 파일 소스 최적화	18
쿼리 최적화	18
조건부 필터	19
고유 항목 선택	19
힌트	19
힌트 규칙 및 지침	20
힌트 작성	20

제약 조건. ....	21
제약 조건 구성. ....	21
사용자 지정 데이터 개체 최적화. ....	22
데이터베이스 소스 최적화. ....	22

## 장 4: 변환 최적화..... 23

변환 최적화. ....	23
집계 변환 최적화. ....	23
식 최적화. ....	24
Java 변환 최적화. ....	26
Java 변환의 초기 선택 최적화. ....	26
Java 변환의 푸시인 최적화. ....	27
조이너 변환 최적화. ....	28
조희 변환 최적화. ....	29
분류기 변환 최적화. ....	31
SQL 변환 최적화. ....	32
SQL 변환의 초기 선택 최적화. ....	32
SQL 변환의 푸시인 최적화. ....	32
변환 캐시. ....	33
변환 오류 제거. ....	34
변환 부작용. ....	34
웹 서비스 소비자 변환 최적화. ....	35
웹 서비스 소비자 변환의 초기 선택 최적화. ....	35
웹 서비스 소비자 변환의 푸시인 최적화. ....	35

## 장 5: 매핑 최적화..... 38

매핑 최적화 개요. ....	38
최적화 방법. ....	39
최적화 수준. ....	39
필터 최적화. ....	40
초기 예측 최적화 방법. ....	40
조건자 최적화 방법. ....	40
비용 기반 최적화 방법. ....	41
데이터십 조인 최적화 방법. ....	42
반 조인 최적화 방법. ....	43
초기 선택 최적화 방법. ....	43
글로벌 조건자 최적화 방법. ....	44
분기 잘라내기 최적화 방법. ....	44
푸시인 최적화 방법. ....	44
푸시다운 최적화. ....	45
전체 푸시다운 최적화. ....	45
소스 푸시다운. ....	46

푸시다운 최적화 규칙 및 지침.....	46
단일 패스 읽기.....	47
필터 최적화.....	47
데이터 유형 변환 최적화.....	48
오류 추적.....	48

## 장 6: 분할된 매핑 최적화..... 49

분할된 매핑 최적화 개요.....	49
여러 개의 CPU 사용.....	49
최대 병렬도 값 증가.....	50
분할을 위한 플랫폼 파일 최적화.....	50
분할을 위한 플랫폼 파일 소스 최적화.....	50
분할을 위한 플랫폼 파일 대상 최적화.....	51
분할을 위해 관계형 데이터베이스 최적화.....	51
분할을 위해 소스 데이터베이스 최적화.....	51
분할을 위해 대상 데이터베이스 최적화.....	52
분할을 위한 변환 최적화.....	53

## 장 7: 런타임 최적화..... 54

런타임 최적화 개요.....	54
응용 프로그램 서비스 최적화.....	54
분석 서비스 최적화.....	54
데이터 통합 서비스 최적화.....	55
모델 리포지토리 서비스 최적화.....	56
모니터링 통계.....	56
메모리 할당.....	57
데이터 개체 캐싱.....	58
캐시 테이블에 대한 데이터 유형.....	59
데이터 개체 캐시 최적화.....	60
시스템 최적화.....	61

## 장 8: SQL 데이터 서비스 최적화..... 62

SQL 데이터 서비스 최적화 개요.....	62
타사 클라이언트 도구 최적화.....	63
SQL 데이터 서비스 최적화 수준.....	63
데이터 미리보기에 대한 SQL 데이터 서비스 최적화 수준 구성.....	64
배포된 SQL 데이터 서비스에 대한 최적화 수준 구성.....	64
SQL 데이터 서비스 쿼리 계획.....	65
SQL 쿼리 계획 보기.....	66
SQL 데이터 서비스의 메모리 및 동시 요청에 대한 속성.....	66
SQL 데이터 서비스에 대한 결과 집합 캐시.....	67
SQL 데이터 서비스 결과 집합 캐시 속성.....	68

SQL 데이터 서비스에 대한 결과 집합 캐싱 활성화.....	68
가상 데이터를 임시 테이블에 보관.....	68
임시 테이블 구현.....	69
<b>장 9: 웹 서비스 최적화.....</b>	<b>70</b>
웹 서비스 최적화 개요.....	70
HTTP 요청 최적화.....	71
웹 서비스 메시지 압축.....	71
웹 서비스 최적화 수준.....	71
데이터 미리보기에 대한 웹 서비스 최적화 수준 구성.....	72
배포된 웹 서비스에 대한 최적화 수준 구성.....	72
메모리 및 동시 요청에 대한 웹 서비스 속성.....	73
동시 웹 서비스 요청에 대한 데이터 통합 서비스 구성 예제.....	74
활성 DTM 인스턴스를 구성하는 웹 서비스 속성.....	75
웹 서비스 결과 집합 캐싱.....	75
웹 서비스에 대한 결과 집합 캐싱 활성화.....	75
웹 서비스 로그 관리.....	76
<b>장 10: 연결 최적화.....</b>	<b>77</b>
연결 최적화 개요.....	77
연결 풀링.....	77
연결 개체의 풀링 속성.....	77
데이터베이스 네트워크 패킷 크기.....	78
<b>인덱스.....</b>	<b>80</b>

# 서문

*Informatica® 성능 조정 가이드*에서 매핑 성능을 최적화하는 방법을 알아보십시오. 각 매핑 구성 요소 내의 성능 병목 현상을 식별하고 제거하는 방법이 설명되어 있습니다.

## Informatica 리소스

Informatica는 Informatica Network 및 기타 온라인 포털을 통해 다양한 범위의 제품 리소스를 제공합니다. 리소스를 통해 Informatica 제품 및 솔루션을 최대한 활용하고 다른 Informatica 사용자 및 주제별 전문가로부터 배울 수 있습니다.

### Informatica 네트워크

Informatica Network는 Informatica 기술 자료, Informatica 글로벌 고객 지원 센터 등 여러 리소스로 연결되는 관문입니다. Informatica Network를 시작하려면 <https://network.informatica.com>을 방문하십시오.

Informatica Network 멤버인 경우 다음 옵션이 가능합니다.

- 기술 자료에서 제품 리소스를 검색할 수 있습니다.
- 제품 사용 가능 여부에 대한 정보를 봅니다.
- 지원 사례를 생성하고 검토할 수 있습니다.
- 거주 지역의 Informatica 사용자 그룹 네트워크를 검색하고 동료와 협업 관계 유지

### Informatica 기술 자료

Informatica 기술 자료를 사용하여 사용 방법 문서, 모범 사례, 비디오 자습서, 자주 묻는 질문에 대한 답변 등 제품 리소스를 확인할 수 있습니다.

기술 자료를 검색하려면 <https://search.informatica.com>을 방문하십시오. 기술 자료에 대한 질문, 의견 또는 아이디어가 있는 경우 [KB\\_Feedback@informatica.com](mailto:KB_Feedback@informatica.com)을 통해 Informatica 기술 자료 팀에 문의해 주시기 바랍니다.

### Informatica 설명서

Informatica 설명서 포털에서 확장된 설명서 라이브러리를 탐색하여 현재 및 최근 제품 릴리스를 확인할 수 있습니다. 설명서 포털을 탐색하려면 <https://docs.informatica.com>을 방문하십시오.

제품 설명서에 대한 질문, 의견 또는 아이디어가 있는 경우 [infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com)에서 Informatica 설명서 팀에 문의해 주시기 바랍니다.

## Informatica Product Availability Matrix

PAM(Product Availability Matrix)은 제품 릴리스에서 지원하는 운영 체제 버전, 데이터베이스 및 데이터 소스 유형과 대상을 나타냅니다.

<https://network.informatica.com/community/informatica-network/product-availability-matrices>에서 Informatica PAM을 찾을 수 있습니다.

## Informatica Velocity

Informatica Velocity는 수백 가지 데이터 관리 프로젝트의 실제 경험을 토대로 Informatica 전문 서비스업에서 개발한 팁과 모범 사례 모음입니다. Informatica Velocity는 전 세계의 조직과 협력하여 성공적인 데이터 관리 솔루션을 계획, 개발, 배포 및 유지 관리하는 Informatica 컨설턴트의 포괄적인 지식을 보여줍니다.

Informatica Velocity 리소스는 <http://velocity.informatica.com>에서 확인할 수 있습니다. Informatica Velocity에 대한 질문, 주석 또는 아이디어가 있으시면 Informatica 전문 서비스업([ips@informatica.com](mailto:ips@informatica.com))에 문의하십시오.

## Informatica Marketplace

Informatica Marketplace는 Informatica 구현을 확대 및 개선하기 위한 솔루션을 찾을 수 있는 포럼입니다. Marketplace에서 Informatica 개발자와 파트너가 제공하는 수백 개의 솔루션을 활용하여 생산성을 향상시키고 프로젝트의 구현에 걸리는 시간을 줄일 수 있습니다. <https://marketplace.informatica.com>에서 Informatica Marketplace를 찾을 수 있습니다.

## Informatica 글로벌 고객 지원 센터

전화 또는 Informatica Network를 통해 글로벌 지원 센터에 문의할 수 있습니다.

해당 지역의 Informatica 글로벌 고객 지원 전화 번호는 Informatica 웹 사이트 (<https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>)를 방문하여 찾을 수 있습니다.

Informatica Network에서 온라인 지원 리소스를 찾으려면 <https://network.informatica.com>을 방문하고 eSupport 옵션을 선택하십시오.



# 제 1 장

## 성능 조정 개요

이 장에 포함된 항목:

- [성능 조정 개요, 9](#)
- [성능 조정 프로세스, 9](#)
- [대상 병목 현상, 10](#)
- [소스 병목 현상, 10](#)
- [매핑 병목 현상, 11](#)
- [컴퓨터 시스템 병목 현상, 11](#)
- [런타임 병목 현상, 12](#)
- [SQL 데이터 서비스 최적화 병목 현상, 12](#)
- [웹 서비스 최적화 병목 현상, 12](#)
- [연결 병목 현상, 13](#)

## 성능 조정 개요

성능 조정의 목표는 성능 병목 현상을 제거하는 것입니다. 병목 현상은 데이터 서비스에서 자주 실행되고 처리량이 낮은 매핑에서 발생합니다. 병목 현상이 발생하면 매핑의 전체 성능이 저하됩니다.

매핑을 최적화하려면 성능 병목 현상을 찾아 제거한 다음 다른 병목 현상을 찾으십시오. 한 번에 한 매핑 구성 요소를 최적화합니다. 변경 전과 후의 매핑 시간을 측정하여 최적화가 성능을 향상시켰는지 확인할 수 있습니다.

소스, 대상 및 연결 같은 매핑 구성 요소에서 병목 현상을 찾고 수정할 수 있습니다. 데이터 통합 서비스와 데이터 통합 서비스를 실행하는 시스템에서 병목 현상을 확인할 수 있습니다. 웹 서비스 및 SQL 데이터 서비스의 속성을 조정할 수도 있습니다.

## 성능 조정 프로세스

일련의 단계를 수행하여 매핑 구성 요소를 조정하고 성능을 개선할 수 있습니다.

다음과 같은 순서로 매핑 구성 요소를 최적화할 수 있습니다.

1. 대상
2. 소스

3. 매핑
4. 변환
5. Administrator 도구의 Informatica 환경
6. 컴퓨터 시스템
7. 데이터 서비스 또는 웹 서비스

다음 방법을 사용하여 성능 병목 현상을 찾습니다.

- 테스트 매핑을 실행합니다. 플랫 파일 소스에서 읽거나 플랫 파일 대상에 쓰는 테스트 매핑을 구성하여 소스 및 대상 병목 현상을 식별할 수 있습니다.
- 성능 세부 정보를 분석합니다. 최적화 방법 같은 성능 세부 정보를 분석하여 매핑 성능이 저하되는 지점을 확인합니다.
- 시스템 성능을 모니터링합니다. 시스템 모니터링 도구를 사용하여 CPU 사용률, I/O 대기 시간, 페이징 및 시스템 리소스 사용 현황을 볼 수 있습니다.

## 대상 병목 현상

대상 병목 현상은 데이터 통합 서비스가 대상에 쓸 때 성능이 저하되는 현상입니다. 데이터베이스가 작은 검사점 간격이나 작은 데이터베이스 네트워크 패킷 크기를 사용할 경우 대상 병목 현상이 발생할 수 있습니다.

가장 일반적인 성능 병목 현상은 데이터 통합 서비스가 대상 데이터베이스에 쓸 때 발생합니다. 데이터베이스가 작은 검사점 간격을 사용할 경우 자주 검사점을 쓰는 만큼 데이터베이스 처리가 느려집니다. 작은 데이터베이스 네트워크 패킷 크기로 인해 병목 현상이 발생할 수 있습니다. 더 큰 데이터 패킷이 한 번에 네트워크를 통해 전달 되도록 허용할 수 있습니다.

대상 병목 현상을 확인하기 위해 데이터베이스 대상 대신 플랫 파일 대상을 사용하는 매핑 복사본을 생성할 수 있습니다. 이렇게 하여 성능이 상당히 향상된다면 대상 병목 현상이 있는 것입니다. 매핑이 이미 플랫 파일 대상에 쓰는 경우 대개 대상 병목 현상이 없습니다.

## 소스 병목 현상

소스 병목 현상이 있으면 데이터 통합 서비스가 소스 데이터베이스에서 읽을 때 성능이 저하됩니다. 소스 쿼리가 효율적이지 않거나 데이터베이스 네트워크 패킷 크기가 작은 경우 소스 병목 현상이 발생할 수 있습니다.

매핑이 관계형 소스에서 읽는 경우 다음 방법을 사용하여 소스 병목 현상을 확인할 수 있습니다.

- 매핑에 필터 변환을 추가합니다. 소스 이후에 필터 변환을 추가합니다. 필터 변환이 아무 데이터도 반환하지 않도록 필터 조건을 **false**로 설정합니다. 매핑에 걸리는 시간이 거의 같다면 매핑에 소스 병목 현상이 있는 것입니다.
- 읽기 테스트 매핑을 생성합니다. 매핑의 복사본을 만들고 모든 변환, 조인 또는 쿼리를 제거합니다. 소스를 대상에 연결합니다. 매핑 성능이 원래 매핑과 비슷하다면 소스 병목 현상이 있는 것입니다.
- 소스 데이터베이스에서 직접 읽기 쿼리를 실행합니다. 매핑 로그에서 읽기 쿼리를 복사합니다. **isql** 같은 쿼리 도구를 사용하여 소스 데이터베이스에 대해 쿼리를 실행합니다. 런타임과 쿼리가 행을 반환하는 데 걸리는 시간을 측정합니다.

## 매핑 병목 현상

소스 병목 현상이나 대상 병목 현상이 없음을 확인했다면 매핑 병목 현상을 확인해야 합니다. 작은 캐시 크기, 적은 버퍼 메모리, 짧은 커밋 간격 등으로 인해 매핑 병목 현상이 발생할 수 있습니다.

매핑 병목 현상을 확인하려면 매핑 로그에서 성능 세부 정보를 분석합니다. 성능 세부 정보에는 입력 행, 출력 행 및 오류 행 수와 같은 각 변환에 대한 정보가 포함됩니다.

각 대상 정의 이전에 필터 변환을 추가할 수도 있습니다. 필터 변환이 대상 테이블로 아무 데이터도 로드하지 않도록 필터 조건을 **false**로 설정합니다. 새 매핑을 실행하는 데 걸리는 시간이 원래 매핑과 같다면 매핑 병목 현상이 있는 것입니다.

## 컴퓨터 시스템 병목 현상

Windows 또는 UNIX에서 Informatica 서비스를 실행할 때 리소스 사용량을 확인할 수 있습니다. Windows에서 작업 관리자를 사용합니다. UNIX에는 성능 검토에 사용할 수 있는 다양한 도구가 있습니다.

### Windows의 시스템 병목 현상 식별

작업 관리자의 성능 및 프로세스 탭에서 시스템 정보를 볼 수 있습니다. 작업 관리자의 성능 탭에는 CPU 사용량과 사용된 총 메모리의 개요가 나타납니다. 더 자세한 정보를 보려면 성능 모니터를 사용합니다.

다음 테이블에는 Windows 성능 모니터에서 도표를 생성하는 데 사용할 수 있는 시스템 정보가 설명되어 있습니다.

속성	설명
Percent processor time(프로세서 시간 백분율)	둘 이상의 CPU가 있는 경우 각 CPU의 프로세서 시간 백분율을 모니터링합니다.
Pages/second(페이지/초)	Pages/second(페이지/초)가 5보다 큰 경우 스래싱(thrashing)으로 알려진 과도한 메모리 사용이 발생할 수 있습니다.
Physical disks percent time(실제 디스크 백분율 시간)	읽기 또는 쓰기 요청을 수행하여 실제 디스크가 사용된 시간의 백분율입니다.
Physical disks queue length(실제 디스크 대기열 길이)	동일한 디스크 장치에 액세스하기 위해 대기 중인 사용자의 수입니다.
Server total bytes per second(서버의 초당 총 바이트 수)	서버가 네트워크에서 주고 받은 바이트 수입니다.

### UNIX의 시스템 병목 현상 식별

UNIX에서 시스템 병목 현상을 확인하려면 다음 도구를 사용합니다.

- **top.** 전체적인 시스템 성능을 표시합니다. 이 도구는 시스템과 시스템에서 실행되는 개별 프로세스의 대한 CPU 사용량, 메모리 사용량 및 스왑 사용량을 보여 줍니다.

- **iostat.** 데이터베이스 서버에 장착된 모든 디스크에 대한 로드 작업을 모니터링합니다. **iostat**은 디스크가 실제로 활성화된 시간의 백분율을 표시합니다. 디스크 배열을 사용하는 경우 **iostat** 대신 디스크 배열과 함께 제공된 유틸리티를 사용하십시오.
- **vmstat.** 디스크 스왑 작업을 모니터링합니다.
- **sar.** CPU, 메모리 및 디스크 사용량에 대한 자세한 시스템 활동 보고서를 표시합니다. 이 도구를 사용하여 CPU 부하를 모니터링할 수 있습니다. 이 도구는 사용자, 시스템, 유휴 시간 및 대기 시간의 사용량을 제공합니다. 이 도구를 사용하여 디스크 스왑 작업을 모니터링할 수도 있습니다.

## 런타임 병목 현상

성능 기능을 활성화하고 데이터 통합 서비스 속성을 조정하여 매핑 성능을 최적화할 수 있습니다.

**Administrator** 도구에서 데이터 통합 서비스 및 모델 리포지토리 서비스에 대한 최적화 설정을 구성합니다.

최적의 시스템 성능에 필요한 메모리를 할당하고 데이터 통합 서비스가 매핑을 실행하면서 생성하는 로그 이벤트의 수를 줄일 수 있도록 오류 추적 수준을 구성합니다.

데이터 통합 서비스가 모든 동시 요청을 실행하기 위해 할당하는 최대 메모리 양을 구성할 수 있습니다. 또한 데이터 통합 서비스가 지정된 요청에 할당하는 최대 메모리 양을 제한할 수도 있습니다.

데이터 통합 서비스가 각 **SQL** 데이터 서비스 쿼리 및 웹 서비스 요청과 연결된 **DTM** 프로세스의 결과를 캐싱할 수 있도록 결과 집합 캐시를 구성할 수 있습니다.

## SQL 데이터 서비스 최적화 병목 현상

**SQL** 데이터 서비스를 최적화하여 최종 사용자가 타사 클라이언트 도구를 사용하여 **SQL** 데이터 서비스에 대한 **SQL** 쿼리를 실행할 때 성능을 향상시킬 수 있습니다. **SQL** 데이터 서비스가 가상 테이블 매핑을 사용하는 경우 변환 및 매핑을 최적화할 수 있습니다.

**JDBC** 드라이버를 최적화하여 **SQL** 데이터 서비스를 쿼리할 때 성능을 향상시킬 수 있습니다. 또한 데이터 통합 서비스에 대한 데이터 개체 캐시를 구성하여 매핑 및 **SQL** 쿼리의 성능을 향상시킬 수 있습니다.

## 웹 서비스 최적화 병목 현상

데이터 통합 서비스가 웹 서비스 요청을 실행하는 경우 웹 서비스를 최적화하여 성능을 향상시킬 수 있습니다. 메모리를 관리하고, 동시 웹 서비스 요청을 처리하고, **DTM** 프로세스를 활성 상태로 유지하여 둘 이상의 웹 서비스 요청을 처리할 수 있도록 데이터 통합 서비스를 조정합니다.

웹 서비스 성능을 향상시키려면 웹 서비스 메시지 압축을 사용하고, **HTTP** 요청을 최적화하고, 데이터 개체 캐시를 구성합니다.

## 연결 병목 현상

연결을 최적화하여 성능을 개선할 수 있습니다. 데이터베이스 연결의 유희 연결 인스턴스 풀을 관리할 수 있습니다. 네트워크 패킷 크기를 늘리면 네트워크를 통해 한 번에 더 많은 데이터 패킷을 전송할 수 있습니다.

## 제 2 장

# 대상 최적화

이 장에 포함된 항목:

- [대상 최적화 개요, 14](#)
- [플랫 파일 대상 최적화, 14](#)
- [데이터베이스 검사점 간격, 15](#)
- [대량 로드, 15](#)
- [데이터베이스 대상 최적화, 15](#)

## 대상 최적화 개요

데이터 통합 서비스가 대상에 효율적으로 쓸 수 있게 하려면 대상을 최적화하십시오. 매핑을 실행하기 전에 인덱스와 키 제약 조건을 삭제하고, 데이터베이스의 검사점 간격 수를 늘리고, 데이터 개체에 대한 쓰기 속성에서 대량 로드를 구성하고, **Oracle** 대상 데이터베이스를 최적화할 수 있습니다.

다음과 같은 최적화 기법을 사용하여 대상을 최적화할 수 있습니다.

- 플랫 파일 대상을 최적화합니다.
- 데이터베이스 검사점 간격을 늘립니다.
- 대량 로드를 사용합니다.
- **Oracle** 대상 데이터베이스를 최적화합니다.

## 플랫 파일 대상 최적화

플랫 파일 대상을 최적화하여 매핑 성능을 향상시킬 수 있습니다. 변환 태스크를 명령에 푸시하여 성능을 향상시킬 수도 있습니다.

플랫 파일 대상의 병목 현상을 줄이려면 다음과 같은 해결 방법을 고려하십시오.

**변환 태스크를 데이터 통합 서비스 대신 명령에 푸시합니다.**

변환 태스크를 데이터 통합 서비스 대신 명령에 푸시하여 매핑 성능을 향상시킬 수 있습니다. 명령을 사용하여 대상 데이터를 정렬하거나 압축할 수도 있습니다. **Developer tool**에서 플랫 파일 대상에 대한 런타임 속성의 명령 속성을 구성합니다.

**UNIX**에서는 유효한 **UNIX** 명령이나 셸 스크립트를 사용합니다. **Windows**에서는 유효한 **DOS** 명령이나 배치 파일을 사용합니다. 플랫 파일 기록기가 플랫 파일 대상 대신 명령에 데이터를 보냅니다.

예를 들어 대상 데이터로부터 압축 파일을 생성하려면 다음 명령을 사용합니다.

```
compress -c - > MyTargetFiles/MyCompressedFile.Z
```

**서비스 프로세스 노드에 대해 로컬인 플랫폼 파일 대상에 씁니다.**

데이터 통합 서비스가 단일 노드에서 실행되고 있고 플랫폼 파일 대상에 쓰는 경우 서비스 프로세스 노드에 대해 로컬인 플랫폼 파일 대상에 씌으로써 매핑 성능을 최적화할 수 있습니다.

## 데이터베이스 검사점 간격

데이터베이스에서 검사점이 실행될 때마다 데이터 통합 서비스가 대기해야 하므로 성능이 저하됩니다.

데이터베이스 검사점 병목 현상을 줄이려면 다음과 같은 해결 방법을 고려하십시오.

**데이터베이스에서 검사점 간격을 늘립니다.**

검사점 수를 줄여 성능을 향상시키려면 데이터베이스에서 검사점 간격을 늘리십시오.

검사점 수를 줄이면 성능이 향상되지만 데이터베이스가 예기치 않게 종료되었을 때 복구 시간이 늘어납니다.

## 대량 로드

데이터베이스 로그는 성능을 향상시키는데, 대량 로드를 사용하면 데이터 통합 서비스가 이 데이터베이스 로그를 바이패스합니다.

대량 로드 병목 현상을 줄이려면 다음과 같은 해결 방법을 고려하십시오.

**데이터 개체의 쓰기 속성에 대량 로드를 구성합니다.**

대량 로드를 사용하면 DB2, Sybase ASE, Oracle 또는 Microsoft SQL Server 데이터베이스에 대량의 데이터를 삽입하는 매핑의 성능을 향상시킬 수 있습니다.

하지만 데이터베이스 로그에 쓰지 않기 때문에 대상 데이터베이스에서 롤백을 수행할 수 없습니다. 결과적으로 복구를 수행할 수 없게 됩니다. 대량 로드를 사용할 경우 향상된 매핑 성능의 중요성을 불완전한 매핑을 복구하는 기능과 비교하여 결정해야 합니다.

## 데이터베이스 대상 최적화

저장소 절, 공간 할당과 롤백 또는 실행 취소 세그먼트를 확인하여 대상 데이터베이스를 최적화할 수 있습니다.

데이터베이스 대상 병목 현상을 줄이려면 다음과 같은 해결 방법을 고려하십시오.

**데이터베이스에서 롤백 또는 실행 취소 세그먼트를 적절한 테이블스페이스, 가능하면 다른 디스크에 있는 테이블스페이스에 저장하는지 확인합니다.**

데이터베이스에 쓸 경우 데이터베이스가 로드되는 동안 롤백 또는 실행 취소 세그먼트가 사용됩니다. 데이터베이스 관리자에게 문의하여 데이터베이스에서 롤백 또는 실행 취소 세그먼트를 적절한 테이블스페이스, 가능하면 다른 디스크에 있는 테이블스페이스에 저장하는지 확인하십시오. 롤백 또는 실행 취소 세그먼트에는 적절한 저장소 절도 있어야 합니다.

#### **데이터베이스 다시 실행 로그를 조정합니다.**

데이터베이스를 최적화하려면 데이터베이스 다시 실행 로그를 조정하십시오. 데이터베이스는 다시 실행 로그를 사용하여 로드 작업을 기록합니다. 다시 실행 로그 크기와 버퍼 크기가 최적화되어 있는지 확인하십시오. Oracle 데이터베이스인 경우 `init.ora` 파일에서 다시 실행 로그 속성을 볼 수 있습니다.

#### **IPC 프로토콜을 통해 Oracle 데이터베이스에 연결합니다.**

데이터 통합 서비스가 단일 노드에서 실행되고 있고 Oracle 인스턴스가 서비스 프로세스 노드에 대해 로컬인 경우 IPC 프로토콜을 사용하여 Oracle 데이터베이스에 연결하면 성능을 최적화할 수 있습니다. `listener.ora` 및 `tnsnames.ora`에서 Oracle 데이터베이스 연결을 설정할 수 있습니다.



## 제 3 장

# 소스 최적화

이 장에 포함된 항목:

- [소스 최적화 개요, 17](#)
- [플랫 파일 소스 최적화, 18](#)
- [쿼리 최적화, 18](#)
- [조건부 필터, 19](#)
- [고유 항목 선택, 19](#)
- [힌트, 19](#)
- [제약 조건, 21](#)
- [사용자 지정 데이터 개체 최적화, 22](#)
- [데이터베이스 소스 최적화, 22](#)

## 소스 최적화 개요

데이터 통합 서비스가 소스 데이터를 효율적으로 읽을 수 있게 하려면 플랫 파일, 관계형 및 사용자 지정 데이터 소스를 최적화하십시오.

다음과 같은 최적화 기법을 사용하여 소스를 최적화할 수 있습니다.

- 소스 데이터를 효율적으로 읽습니다.
- 쿼리 최적화 기법을 사용합니다.
- SQL 쿼리에 조건부 필터를 사용합니다.
- 소스에서 고유 값을 선택합니다.
- SQL 쿼리에 힌트를 적용합니다.
- 논리적 데이터 개체, 실제 데이터 개체 및 가상 테이블에 대한 제약 조건을 구성합니다.
- 최적화를 위해 사용자 지정 데이터 개체를 구성합니다.
- 최적화를 위해 Oracle, Sybase 및 Microsoft SQL Server 데이터베이스를 구성합니다.

## 플랫 파일 소스 최적화

데이터 통합 서비스가 소스 데이터를 효율적으로 읽을 수 있게 하려면 플랫 파일 소스의 형식 속성을 구성하십시오.

플랫 파일 소스 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**구분자로 분리된 플랫 파일의 형식 속성에 따옴표 또는 이스케이프 문자를 사용하지 마십시오.**

이스케이프 문자를 지정하면 데이터 통합 서비스가 구분자 문자를 문자열에 포함된 일반 문자로 읽습니다. 소스 파일에 따옴표 또는 이스케이프 문자가 포함되어 있지 않으면 매핑 성능이 다소 향상될 수 있습니다.

**데이터 통합 서비스가 각 줄에서 읽는 바이트 수를 설정합니다.**

매핑이 플랫 파일 소스에서 읽는 경우 데이터 통합 서비스가 각 줄에서 읽는 바이트 수를 설정하면 매핑 성능을 향상시킬 수 있습니다. 플랫 파일 소스의 런타임 속성에 순차 정렬 버퍼 길이 속성을 구성합니다.

기본적으로 데이터 통합 서비스는 각 줄에서 1024바이트를 읽습니다. 소스 파일의 각 줄이 기본 설정보다 짧은 경우 매핑 속성에서 순차 정렬 버퍼 길이를 줄일 수 있습니다.

## 쿼리 최적화

매핑이 사용자 지정 데이터 개체 하나에서 여러 소스 테이블을 조인하는 경우 최적화 힌트로 쿼리를 최적화하여 성능을 향상시킬 수 있습니다. 또한 ORDER BY 또는 GROUP BY 절이 있는 단일 테이블 Select 문은 인덱스 추가 등에서 최적화를 활용할 수 있습니다.

쿼리 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**데이터베이스가 특정 소스 테이블 집합에 대해 쿼리를 실행하는 방법을 지정하는 최적화 프로그램 힌트를 생성합니다.**

일반적으로 데이터베이스 최적화 프로그램이 소스 데이터를 처리하는 가장 효율적인 방법을 결정합니다. 하지만 사용자가 데이터베이스 최적화 프로그램이 알지 못하는 소스 테이블 속성을 알고 있는 경우도 있습니다. 데이터베이스 관리자는 데이터베이스가 특정 소스 테이블 집합에 대해 쿼리를 실행하는 방법을 지정하는 최적화 프로그램 힌트를 생성할 수 있습니다.

**모든 행을 한 번에 반환하는 대신 최대한 빨리 행을 반환하도록 최적화 프로그램 힌트를 구성합니다.**

쿼리를 실행한 시점과 데이터 통합 서비스가 데이터의 첫 행을 받는 시점 사이에 긴 지연 시간이 있는 경우 최적화 힌트를 사용합니다. 모든 행을 한 번에 반환하는 대신 최대한 빨리 행을 반환하도록 최적화 프로그램 힌트를 구성합니다. 이렇게 하면 데이터 통합 서비스가 쿼리 실행과 병렬로 행을 처리할 수 있습니다.

**ORDER BY 또는 GROUP BY 열에서 인덱스를 생성합니다.**

ORDER BY 또는 GROUP BY 열에서 인덱스를 생성하면 ORDER BY 또는 GROUP BY 절을 포함하는 쿼리의 성능이 향상될 수 있습니다. 쿼리를 최적화했으면 SQL 재정의 옵션을 사용하여 최적화를 최대한 활용합니다.

**병렬 쿼리를 실행하도록 데이터베이스를 구성합니다.**

병렬 쿼리를 실행하도록 소스 데이터베이스를 구성하여 성능을 향상시킬 수도 있습니다. 병렬 쿼리 구성에 대한 자세한 내용은 데이터베이스 설명서를 참조하십시오.

데이터 통합 서비스가 데이터 읽기에 사용하는 쿼리가 SQL 데이터 서비스의 가상 데이터베이스에 나타납니다. 또한 사용자 지정 데이터 개체에서도 이 쿼리를 찾을 수 있습니다. 데이터베이스 관리자에게 쿼리를 분석하여 소스 테이블에 대한 최적화 프로그램 힌트와 인덱스를 생성하게 하십시오.

## 조건부 필터

인덱스가 누락된 경우 소스 데이터베이스에 대한 단순한 소스 필터가 성능에 악영향을 미칠 수 있습니다. 성능을 향상시키려면 사용자 지정 데이터 개체에서 조건부 필터를 사용하십시오.

조건부 필터 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**동일한 소스에서 동시에 읽는 여러 매핑에 조건부 필터를 사용합니다.**

여러 매핑이 동일한 소스에서 동시에 읽는 경우 조건부 필터가 성능을 향상시킬 수 있습니다.

하지만 일부 매핑은 소스 데이터베이스에서 소스 데이터를 필터링하면 성능이 더 향상될 수 있습니다. 데이터베이스 필터와 조건부 필터 양쪽으로 매핑을 테스트하여 어떤 필터가 성능을 향상시키는지 확인하십시오.

## 고유 항목 선택

고유 항목 선택 옵션을 사용하여 사용자 지정 데이터 개체의 소스에서 고유 값을 선택할 수 있습니다. 고유 항목 선택을 사용하면 데이터 통합 서비스가 기본 SQL 쿼리에 **SELECT DISTINCT** 문을 추가합니다.

고유 항목 선택 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**고유 항목 선택 옵션을 사용하여 데이터 흐름의 초기에 불필요한 데이터를 필터링합니다.**

데이터 통합 서비스가 소스에서 고유 값을 선택하게 하려면 사용자 지정 데이터 개체에 대해 고유 항목 선택 옵션을 사용합니다. 고유 항목 선택 옵션을 사용하여 데이터 흐름의 초기에 불필요한 데이터를 필터링합니다. 이렇게 하려면 성능이 향상될 수 있습니다.

예를 들어 고유 항목 선택 옵션을 사용하여 총 매출을 나열하는 테이블에서 고유한 고객 ID를 추출할 수 있습니다. 매핑에서 사용자 지정 데이터 개체를 사용할 경우 데이터 통합 서비스가 데이터 흐름의 초기에 불필요한 데이터를 필터링하므로 성능이 향상될 수 있습니다.

## 힌트

소스 SQL 쿼리에 힌트를 추가하여 데이터베이스 최적화 프로그램에 명령을 전달할 수 있습니다. 최적화 프로그램이 힌트를 사용하여 소스에 액세스할 쿼리 실행 계획을 선택합니다.

힌트 필드는 관계형 데이터 개체 인스턴스 또는 사용자 지정된 데이터 개체의 **쿼리** 보기에 나타납니다. 소스 데이터베이스는 **Oracle**, **Sybase**, **IBM DB2** 또는 **Microsoft SQL Server**여야 합니다. 다른 데이터베이스 유형의 경우 힌트 필드가 나타나지 않습니다.

데이터 통합 서비스는 소스 쿼리를 생성할 때 사용자가 **Developer** 도구에서 입력한 것과 동일한 SQL 힌트를 쿼리에 추가합니다. 데이터 통합 서비스는 힌트를 구문 분석하지 않습니다. 소스가 포함된 매핑을 실행하는 경우 매핑 로그에서 쿼리 및 쿼리의 힌트를 보여 줍니다.

데이터 통합 서비스는 데이터베이스 유형에 따라 쿼리의 특정 위치에 SQL 힌트를 삽입합니다. 힌트 구문에 대한 자세한 내용은 데이터베이스 설명서를 참조하십시오.

### Oracle

데이터 통합 서비스가 **SELECT/UPDATE/INSERT/DELETE** 키워드 바로 뒤에 힌트를 추가합니다.

```
SELECT /*+ <힌트> */ FROM ...
```

'+'는 힌트의 시작을 나타냅니다.

힌트는 주석(/ \* ... \*/ 또는 -- ... 줄의 끝까지) 내에 포함됩니다.

## Sybase

데이터 통합 서비스가 쿼리 뒤에 힌트를 추가합니다. 힌트에 계획 이름을 구성합니다.

```
SELECT ... PLAN <plan>
```

제목 계획 "(scalar\_agg (i\_scan type\_price\_ix titles ))"에서 평균(가격) 선택

## IBM DB2

optimize-for 절을 힌트로 입력할 수 있습니다. 데이터 통합 서비스가 이 절을 쿼리의 끝에 추가합니다.

```
SELECT ... OPTIMIZE FOR <n> ROWS
```

optimize-for 절은 쿼리에서 처리할 수 있는 행의 수를 데이터베이스 최적화 프로그램에 알려 줍니다. 이 절은 행의 수를 제한하지 않습니다. 데이터베이스가 <n>개가 넘는 행을 처리하는 경우 성능이 저하될 수 있습니다.

## Microsoft SQL Server

데이터 통합 서비스가 쿼리의 끝에 OPTION 절의 일부로 힌트를 추가합니다.

```
SELECT ... OPTION ( <query_hints> )
```

# 힌트 규칙 및 지침

SQL 쿼리의 힌트를 구성할 때 다음 규칙 및 지침을 따르십시오.

- 푸시다운 최적화를 활성화하거나 관계형 데이터 개체에서 반 조인을 사용하는 경우 원래 소스 쿼리가 변경됩니다. 데이터 통합 서비스는 수정된 쿼리에 힌트를 적용하지 않습니다.
- 힌트를 조인 및 필터 재정의와 결합할 수 있지만 SQL 재정의의 구성하는 경우 SQL 재정의가 우선하고 데이터 통합 서비스가 다른 재정의의 적용하지 않습니다.
- **쿼리** 보기에는 샘플 보기 또는 고급 보기가 표시됩니다. 샘플 보기에서 힌트를 필터, 정렬 또는 조인 재정의와 함께 입력하는 경우 **Developer** 도구가 고급 보기에서 전체 쿼리 재정의의 보여 줍니다.

# 힌트 작성

쿼리 계획을 결정하도록 데이터베이스 최적화 프로그램에 지침을 보내는 쿼리를 작성할 수 있습니다.

1. 사용자 지정된 데이터 개체 또는 관계형 데이터 개체 인스턴스를 엽니다.
2. **읽기** 보기를 선택합니다.
3. **출력 변환**을 선택합니다.
4. **쿼리 속성**을 선택합니다.
5. **단순 쿼리**를 선택합니다.
6. **힌트** 필드 옆에서 **편집**을 클릭합니다.  
**힌트 대화 상자**가 나타납니다.
7. **SQL 쿼리** 필드에 힌트를 입력합니다.  
**Developer** 도구는 힌트의 유효성을 검사하지 않습니다.
8. **확인**을 클릭합니다.
9. 데이터 개체를 저장합니다.

## 제약 조건

데이터 통합 서비스는 관계형 소스, 플랫 파일 소스, 논리적 데이터 개체 또는 가상 테이블에서 제약 조건을 읽을 수 있습니다. 제약 조건은 조건부 식으로, 데이터 행의 값이 이 식을 충족시켜야 합니다.

데이터 통합 서비스가 제약 조건을 읽으면서, 적용된 최적화 방법에 기반하는 데이터 행에 대해 **TRUE**로 평가되지 않는 행을 삭제할 수 있습니다.

제약 조건을 설정하기 전에 소스 데이터가 제약 조건에 의해 설정된 조건을 충족하는지 확인해야 합니다.

예를 들어 소스 데이터베이스에 **AGE** 열이 있으며, **AGE < 70**인 행만 있는 것으로 나타납니다. 이 경우 소스 데이터베이스에 대해 **AGE < 70**인 제약 조건을 설정할 수 있습니다. 데이터 통합 서비스는 제약 조건 **AGE < 70**을 사용하여 소스 데이터베이스에서 레코드를 읽습니다. 데이터 통합 서비스가 **AGE >= 70**인 레코드를 읽은 경우 **AGE >= 70**인 행을 삭제합니다.

데이터베이스에 연결할 때 데이터베이스 환경에 대한 제약 조건을 설정하는 **SQL** 명령을 데이터베이스에서 사용할 수 있습니다. 데이터 통합 서비스에서 데이터베이스에 연결할 때마다 연결 환경 **SQL**을 실행합니다.

**Developer tool**을 사용하여 논리적 데이터 개체, 실제 데이터 개체 및 가상 테이블에 대한 제약 조건을 설정합니다. 제약 조건을 설정할 경우 각 데이터 행에 대해 **TRUE**로 평가되는 식을 입력해야 합니다.

## 제약 조건 구성

관계형 데이터 개체, 플랫 파일 데이터 개체, 사용자 지정된 데이터 개체, 논리적 데이터 개체 및 가상 테이블에 제약 조건을 추가할 수 있습니다. 제약 조건을 추가한 후 제약 조건을 편집하거나 삭제할 수 있습니다.

1. **Object Explorer** 보기에서 읽기 변환으로 추가된 관계형 데이터 개체가 포함된 매핑을 엽니다. 또는 플랫 파일 데이터 개체, 사용자 지정된 데이터 개체, 논리적 데이터 개체 또는 가상 테이블을 엽니다.
  - 매핑에 읽기 변환으로 추가된 관계형 데이터 개체에 대한 제약 조건을 설정하려면 매핑에서 읽기 변환을 선택합니다. **속성** 보기에서 **고급** 탭을 선택합니다.
  - 플랫 파일 데이터 개체에 대한 제약 조건을 설정하려면 **고급** 보기를 선택하고 **런타임: 읽기** 섹션을 확장합니다.
  - 사용자 지정된 데이터 개체에 대한 제약 조건을 설정하려면 **읽기** 보기를 선택하고 소스 변환의 **출력** 포트를 선택합니다. **속성** 보기에서 **고급** 탭을 선택합니다.
  - 논리적 데이터 개체에 대한 제약 조건을 설정하려면 논리적 데이터 모델을 선택하고 논리적 데이터 개체를 선택합니다. **속성** 보기에서 **고급** 탭을 선택합니다.
  - 가상 테이블에 대한 제약 조건을 설정하려면 **SQL** 끝점에서 가상 테이블을 엽니다. **속성** 보기에서 **고급** 탭을 선택합니다.
2. 제약 조건의 값 필드를 클릭합니다.  
**제약 조건** 대화 상자가 표시됩니다.
3. **새로 만들기**를 클릭하여 식 편집기를 엽니다.
4. 제약 조건 논리를 구성하고 식 함수 및 열을 매개 변수로 사용합니다.
5. **유효성 검사**를 클릭합니다.
6. **확인**을 클릭합니다.

## 사용자 지정 데이터 개체 최적화

사용자 지정 데이터 개체를 구성하여 성능을 향상시킬 수 있습니다. 사용자 지정 데이터 개체의 소스에서 SQL 쿼리를 최적화하고, 조건부 필터를 사용하고, 고유 값을 선택할 수 있습니다.

사용자 지정 데이터 개체 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**데이터 통합 서비스가 소스 데이터를 읽는 특수한 SELECT 문을 실행하도록 사용자 지정 쿼리를 생성합니다.**

사용자 지정 쿼리는 데이터 통합 서비스가 소스의 데이터를 읽는 데 사용하는 기본 쿼리를 대체합니다.

**데이터 통합 서비스가 소스 데이터를 읽을 때 행을 필터링합니다.**

필터 조건을 포함시키면 데이터 통합 서비스가 기본 쿼리에 WHERE 절을 추가합니다.

**소스에서 고유 값을 선택합니다.**

고유 항목 선택을 선택하면 데이터 통합 서비스가 기본 SQL 쿼리에 SELECT DISTINCT 문을 추가합니다.

**데이터베이스 힌트를 적용합니다.**

소스 SQL 쿼리에 힌트를 추가하여 데이터베이스 최적화 프로그램에 명령을 전달할 수 있습니다.

**소스 데이터에 대한 제약 조건을 구성합니다.**

사용자 지정 데이터 개체에서 플랫폼 파일과 관계형 테이블에 대한 제약 조건을 구성하면 데이터 통합 서비스가 데이터 행에서 TRUE로 평가되지 않는 행을 삭제합니다.

## 데이터베이스 소스 최적화

소스 데이터베이스가 Oracle인 경우 IPC 프로토콜을 사용하여 Oracle 데이터베이스에 연결하면 데이터 통합 서비스 성능을 최적화할 수 있습니다. 임시 데이터베이스를 디스크 배열로 이동하여 성능을 향상시킬 수도 있습니다.

데이터베이스 소스 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**IPC 프로토콜을 사용하여 Oracle 데이터베이스에 연결합니다.**

데이터 통합 서비스가 단일 노드에서 실행되고 있고 Oracle 인스턴스가 서비스 프로세스 노드에 대해 로컬 인 경우 IPC 프로토콜을 사용하여 Oracle 데이터베이스에 연결하면 성능을 최적화할 수 있습니다.

listener.ora 및 tnsnames.ora에서 Oracle 데이터베이스 연결을 설정할 수 있습니다.

**임시 데이터베이스와 다시 실행 로그를 디스크 배열이나 고속 드라이브로 이동합니다.**

데이터베이스에서 대규모 테이블을 조인할 때 캐시 위치로 RAID(Redundant Array of Independent Disks)를 사용할 수 있습니다. 또한, 다른 디스크에 있는 기본 파일 그룹에 더 많은 파일을 추가하여 디스크 간에 로드를 분산시킬 수 있습니다.

## 제 4 장

# 변환 최적화

이 장에 포함된 항목:

- [변환 최적화, 23](#)
- [집계 변환 최적화, 23](#)
- [식 최적화, 24](#)
- [Java 변환 최적화, 26](#)
- [조이너 변환 최적화, 28](#)
- [조회 변환 최적화, 29](#)
- [분류기 변환 최적화, 31](#)
- [SQL 변환 최적화, 32](#)
- [변환 캐시, 33](#)
- [변환 오류 제거, 34](#)
- [변환 부작용, 34](#)
- [웹 서비스 소비자 변환 최적화, 35](#)

## 변환 최적화

데이터 통합 서비스가 매핑의 변환을 효율적으로 처리할 수 있도록 변환을 최적화합니다.

다음과 같은 최적화 기법을 사용하여 변환을 최적화할 수 있습니다.

- 최적화를 위해 변환을 구성합니다.
- 변환 오류를 제거합니다.
- 변환 캐시를 구성합니다.

## 집계 변환 최적화

집계 변환은 데이터를 그룹화한 후 처리할 수 있기 때문에 성능을 저하시키는 경우가 많습니다. 집계 변환이 중간 그룹 결과를 유지하려면 추가 메모리가 필요합니다.

집계 변환 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

#### 단순한 열을 기준으로 그룹화합니다.

단순한 열을 기준으로 그룹화하면 집계 변환을 최적화할 수 있습니다. 가능한 경우 GROUP BY에 사용되는 열에서 문자열이나 날짜 대신 숫자를 사용합니다. 집계 식에서 복잡한 식을 사용하지 않습니다.

#### 정렬된 입력을 사용합니다.

매핑 성능을 향상시키려면 집계 변환에 사용할 데이터를 정렬하십시오. 정렬된 입력 옵션을 사용하여 데이터를 정렬합니다.

정렬된 입력 옵션은 집계 캐시 사용을 줄입니다. 정렬된 입력 옵션을 사용하면 데이터 통합 서비스에서 모든 데이터가 그룹을 기준으로 정렬된다고 가정합니다. 데이터 통합 서비스는 그룹에 대한 행을 읽을 때 집계 계산 수행하고, 필요에 따라 메모리에 그룹 정보를 저장합니다.

정렬된 입력 옵션은 매핑 중에 캐싱되는 데이터의 양을 줄여 성능을 향상시킵니다. 정렬된 입력 옵션이나 분류기 변환을 사용하여 정렬된 데이터를 집계 변환으로 전달합니다.

여러 개의 파티션이 있는 매핑에서 정렬된 입력 옵션을 사용하면 성능을 향상시킬 수 있습니다.

#### 데이터를 집계하기 전에 필터링합니다.

매핑에서 필터 변환을 사용하는 경우 집계 변환 앞에 변환을 배치하여 불필요한 집계를 줄이십시오.

#### 포트 연결을 제한합니다.

연결된 입력/출력 또는 출력 포트 수를 제한하여 집계 변환이 데이터 캐시에 저장하는 데이터의 양을 줄이십시오.

## 식 최적화

변환에 사용된 일부 식이 성능을 저하시킬 수 있습니다.

식 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

#### 느린 식을 격리합니다.

느린 식은 매핑 성능을 저하시킵니다. 느린 식을 격리하려면 매핑에서 한 번에 하나씩 식을 제거하고 매핑을 실행하여 식이 없이 매핑을 실행하는 데 걸리는 시간을 확인합니다. 매핑 런타임에 상당한 차이가 있는 경우 느린 식을 최적화할 방법을 찾습니다.

식 성능을 평가하려면 다음 단계를 수행합니다.

1. 원래 식이 있는 상태로 매핑의 시간을 측정합니다.
2. 매핑의 복사본을 만들고 복잡한 식의 절반을 상수로 바꿉니다.
3. 편집한 매핑을 실행하고 시간을 측정합니다.
4. 매핑의 또 다른 복사본을 만들고 복잡한 식의 다른 절반을 상수로 바꿉니다.
5. 편집한 매핑을 실행하고 시간을 측정합니다.

#### 공통 논리를 추출합니다.

매핑이 여러 곳에서 동일한 태스크를 수행하는 경우 매핑의 앞부분으로 태스크를 이동하여 매핑이 태스크를 수행하는 횟수를 줄입니다. 예를 들어 매핑에 대상 테이블 5개가 있다고 가정합니다. 각 대상에는 주민등록번호 조회가 필요합니다. 조회를 5번 수행하는 대신 매핑에서 데이터 흐름이 분할되기 전에 조회 변환을 배치합니다. 그런 다음, 조회 결과를 5개 대상 모두에 전달합니다.



### 집계 함수 호출을 최소화합니다.

식을 작성할 때 가능한 한 집계 함수 호출 횟수를 줄입니다. 집계 함수 호출을 사용할 때마다 데이터 통합 서비스가 데이터를 검색하고 그룹화해야 합니다. 예를 들어 다음 식에서 데이터 통합 서비스는 COLUMN\_A를 읽고 합계를 구하고, COLUMN\_B를 읽고 합계를 구한 다음, 최종적으로 두 합계의 합계를 구합니다.

```
SUM(COLUMN_A) + SUM(COLUMN_B)
```

이 경우 공통적인 집계 함수 호출을 줄이면 아래와 같이 데이터 통합 서비스가 COLUMN\_A와 COLUMN\_B를 더한 다음 둘 모두의 합계를 구합니다.

```
SUM(COLUMN_A + COLUMN_B)
```

### 공통 식을 로컬 변수로 바꿉니다.

한 변환에서 동일한 식을 여러 번 사용하는 경우 식을 로컬 변수로 만들 수 있습니다. 로컬 변수는 변환 내에서만 사용할 수 있습니다. 하지만 변수를 한 번만 계산하여 성능을 향상시킬 수 있습니다.

### 문자열 연산자 대신 숫자 연산자를 선택합니다.

데이터 통합 서비스는 숫자 연산을 문자열 연산보다 빠르게 처리합니다. 예를 들어 EMPLOYEE\_NAME과 EMPLOYEE\_ID라는 두 열에서 대량의 데이터를 조회하는 경우 EMPLOYEE\_ID를 조회하도록 구성하면 성능이 향상됩니다.

### CHAR-CHAR 및 CHAR-VARCHAR 비교를 최적화합니다.

데이터 통합 서비스가 CHAR 열과 VARCHAR 열 간의 비교를 수행할 때 행에서 후행 공백을 찾을 때마다 속도가 느려집니다. Informatica Administrator에서 데이터 통합 서비스를 구성할 때 TreatCHARasCHARonRead 옵션을 사용하면 데이터 통합 서비스가 Char 소스 필드의 끝에서 후행 공백을 자르지 않습니다.

### LOOKUP 대신 DECODE를 선택합니다.

LOOKUP 함수를 사용하면 데이터 통합 서비스가 데이터베이스에서 테이블을 조회해야 합니다. DECODE 함수를 사용할 때는 조회 값을 식에 포함시킬 수 있으므로 데이터 통합 서비스가 별도의 테이블을 조회할 필요가 없습니다. 따라서 변하지 않는 값의 작은 집합을 조회할 때 DECODE를 사용하여 성능을 향상시킬 수 있습니다.

### 함수 대신 연산자를 사용합니다.

데이터 통합 서비스는 함수로 작성된 식보다 연산자로 작성된 식을 더 빠르게 읽습니다. 가능한 경우 연산자를 사용하여 식을 작성하십시오. 예를 들어 중첩된 CONCAT 함수를 포함하는 다음과 같은 식을 가정합니다.

```
CONCAT( CONCAT( CUSTOMERS.FIRST_NAME, ' ') CUSTOMERS.LAST_NAME)
```

이 식을 || 연산자를 사용하여 다음과 같이 다시 작성할 수 있습니다.

```
CUSTOMERS.FIRST_NAME || ' ' || CUSTOMERS.LAST_NAME
```

### IIF 함수를 최적화합니다.

IIF 함수는 값과 동작을 반환할 수 있으므로 보다 간결한 식을 작성할 수 있습니다. 예를 들어 소스에 3개의 Y/N 플래그인 FLG\_A, FLG\_B, FLG\_C가 있다고 가정합니다. 각 플래그의 값을 기반으로 값을 반환하려고 합니다.

다음과 같은 식을 사용합니다.

```
IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'Y',  
    VAL_A + VAL_B + VAL_C,  
    IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'N',  
        VAL_A + VAL_B ,  
        IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'Y',  
            VAL_A + VAL_C,  
            IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'N',  
                VAL_A ,  
                IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'Y',  
                    VAL_B + VAL_C,
```

```

IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'N',
VAL_B,
IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'Y',
VAL_C,
IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'N',
0.0,
)))))))))

```

## Java 변환 최적화

매핑에서 일부 Java 변환이 성능을 저하시킬 수 있습니다.

Java 변환 성능을 높이려면 다음과 같은 해결 방법을 고려하십시오.

**Java 변환에서 초기 선택 또는 푸시인 필터 최적화 방법을 활성화합니다.**

Java 변환에서 초기 선택 또는 푸시인 최적화를 활성화할 수 있습니다. Java 변환의 **최적화 프로그램 인터페이스** 탭에서 코드 조각을 업데이트합니다.

### Java 변환의 초기 선택 최적화

Java 변환에 부작용이 없는 경우 초기 선택 최적화에 대해 능동 또는 수동 Java 변환을 활성화할 수 있습니다. 최적화 프로그램은 Java 변환을 통해 필터 논리를 전달하고 필요에 따라 필터 조건을 수정합니다.

초기 선택 최적화를 위한 코드 조각을 보려면 **최적화 프로그램 인터페이스** 탭의 탐색기에서 PredicatePushOptimization을 선택합니다.

#### allowPredicatePush

부울. 초기 선택을 활성화합니다. 초기 선택을 활성화하려면 함수를 true 결과와 메시지를 반환하도록 변경합니다. 기본값은 false이며 함수에서 최적화가 지원되지 않는다는 메시지가 반환됩니다.

```

public ResultAndMessage allowPredicatePush(boolean ignoreOrderOfOp) {
    // To Enable PredicatePushOptimization, this function should return true
    //return new ResultAndMessage(true, "");
    return new ResultAndMessage(false, "Predicate Push Optimization Is Not Supported");
}

```

#### canGenerateOutputFieldEvalError

부울. Java 변환이 0으로 나누기 오류 같은 출력 필드 오류를 반환할 수 있는지 여부를 나타냅니다. Java 변환이 출력 필드 오류를 생성하지 않는 경우 false를 반환하도록 함수를 변경하십시오. Java 변환이 실패 오류를 생성할 수 있는 경우에는 데이터 통합 서비스가 초기 선택 최적화를 사용할 수 없습니다.

```

public boolean canGenerateOutputFieldEvalError() {
    // If this Java transformation can never generate an output field evaluation error,
    // return false.
    return true;
}

```

#### getInputExpr

입력 필드의 입력 값 중에서 출력 필드를 구성하는 입력 값을 설명하는 Informatica 식을 반환합니다. 최적화 프로그램이 변환을 통해 필터 논리를 푸시하려면 어떤 입력 필드가 출력 필드를 구성하는지 알아야 합니다.

```

public InfaExpression getInputExpr(TransformationField field,
TransformationDataInterface group) {
    // This should return an Informatica expression for output fields in terms of input fields
    // We will only push predicate that use fields for which input expressions are defined.
    // For example, if you have two input fields in0 and in1 and three output fields out0, out1, out2
    // out0 is the pass-through of in1, out2 is sum of in1 and in2, and out3 is unknown, the code should

```

```

be:
    //if (field.getName().equals("out0"))
    //    return new InfaExpression("in0", instance);
    //else if (field.getName().equals("out1"))
    //    return new InfaExpression("in0 + in1", instance);
    //else if (field.getName().equals("out2"))
    //    return null;
    return null;
}

```

예를 들어 매핑에 필터 식 "out0 > 8"이 포함되어 있다고 가정합니다. out0은 Java 변환에서 out0 출력 포트의 값입니다. out0의 값을 in0 입력 포트 + 5의 값으로 정의할 수 있습니다. 최적화 프로그램이 다음 식 "(in0 + 5) > 8"을 초기 선택 최적화를 사용하는 Java 변환으로 푸시할 수 있습니다. 출력 필드에 입력 필드 식이 없는 경우에는 NULL을 반환할 수 있습니다. 이 경우 최적화 프로그램이 필터 식을 입력 식이 없는 출력 필드에 푸시하지 않습니다.

다음과 같은 코드를 포함할 수 있습니다.

```

if (field.getName().equals("out0"))
    return new InfaExpression("in0 + 5", instance);
else if (field.getName().equals("out2"))
    return null;

```

## inputGroupsPushPredicateTo

필터 논리를 받을 수 있는 그룹 목록을 반환합니다. Java 변환에는 입력 그룹 하나가 있습니다. Java 변환의 경우 이 함수를 수정하지 마십시오.

```

public List<TransformationDataInterface> inputGroupsPushPredicateTo(
    List<TransformationField> fields) {
    // This functions returns a list of input data interfaces to push predicates to.
    // Since JavaTx only has one input data interface, you should not have to modify this function
    AbstractTransformation tx = instance.getTransformation();
    List<DataInterface> dis = tx.getDataInterfaces();
    List<TransformationDataInterface> inputDIs = new ArrayList<TransformationDataInterface>();
    for (DataInterface di : dis){
        TransformationDataInterface tdi = (TransformationDataInterface) di;
        if (tdi.isInput())
            inputDIs.add(tdi);
    }
    if(inputDIs.size() == 1)
        return inputDIs;
    else
        return null;
}

```

## Java 변환의 푸시인 최적화

Java 변환에 부작용이 없으며 최적화가 매핑 결과에 영향을 미치지 않는 경우 푸시인 최적화에 대한 능동 Java 변환을 활성화할 수 있습니다.

Java 변환에 대한 푸시인 최적화를 구성할 때 Java 변환이 최적화 프로그램에서 받는 필터 조건을 저장하는 방법을 정의합니다. 필터 조건을 검사하는 코드를 추가합니다. Java 변환은 필터 논리를 포함할 수 있으면 true 조건을 최적화 프로그램에 전달합니다. 최적화 프로그램은 최적화된 매핑에서 필터 변환을 제거합니다.

Java 변환을 구성할 때, 최적화 중에 필터 조건을 변환 메타데이터로 저장하는 코드를 작성합니다. 런타임에 필터 조건을 검색하고 필터 논리에 따라 행을 삭제하는 코드도 작성합니다.

Java 변환을 정의할 때 Java 변환 **최적화 프로그램 인터페이스** 탭에서 푸시인 최적화를 위한 코드를 추가합니다. 푸시인 최적화를 위한 코드 조각에 액세스하려면 변환 **최적화 프로그램 인터페이스** 탭의 탐색기에서 **FilterPushdownOptimization**을 선택합니다.

개발자 도구에 푸시인 최적화를 활성화하고 최적화 프로그램에서 필터 조건을 검색하는 코드 조각이 표시됩니다. 최적화를 활성화하고 필터 논리를 변환 메타데이터로 저장하도록 코드 조각을 업데이트합니다.

## isFilterSupported

푸시인 최적화를 활성화하려면 `true`를 반환합니다. 푸시인 최적화를 비활성화하려면 `false`를 반환합니다. 푸시인 최적화를 활성화하려면 `true`를 반환하도록 함수를 변경합니다.

```
public ResultAndMessage isFilterSupported() {
    // To enable filter push-into optimization this function should return true
    // return new ResultAndMessage(true, "");
    return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

## pushFilter

최적화 프로그램에서 필터 조건을 받습니다.

필터를 검사하고 변환에서 필터 논리를 사용할 수 있는지 여부를 확인하는 코드를 추가합니다. 변환은 필터를 포함할 수 있으면 다음 메시지를 사용하여 필터 조건을 변환 메타데이터로 저장합니다.

`storeMetadata(문자열 키, 문자열 데이터)`

키는 메타데이터에 대한 식별자입니다. 원하는 문자열을 키로 정의할 수 있습니다. 데이터는 런타임에 삭제할 행을 결정하기 위해 저장하려는 데이터입니다. 예를 들어 `Java` 변환이 최적화 프로그램에서 받는 필터 조건이 데이터일 수 있습니다.

```
public ResultAndMessage pushFilter(InfaExpression condition) {
    // Add code to absorb the filter
    // If filter is successfully absorbed return new ResultAndMessage(true, ""); and the optimizer
    // will remove the filter from the mapping
    // If the filter is not absorbed, return new ResultAndMessage(false, msg);
    return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

# 조이너 변환 최적화

조이너 변환은 런타임 시 중간 결과를 유지하기 위해 추가 공간을 필요로 하므로 성능을 저하시킬 수 있습니다.

조이너 변환 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**마스터 소스를 중복 키 값이 더 적은 소스로 지정합니다.**

데이터 통합 서비스가 정렬된 조이너 변환을 처리할 때 한 번에 백 개의 고유 키에 대한 행을 캐싱합니다. 마스터 소스에 키 값이 같은 행이 많이 포함된 경우 데이터 통합 서비스가 더 많은 행을 캐싱해야 하므로 성능이 저하될 수 있습니다.

**마스터 소스를 행이 더 적은 소스로 지정합니다.**

조이너 변환이 마스터 소스에 대해 세부 소스의 각 행을 비교합니다. 마스터의 행 수가 적을수록 조인 비교 횟수가 줄어들고 조인 프로세스가 빨라집니다.

**가능한 경우 데이터베이스에서 조인을 수행합니다.**

데이터베이스에서 조인을 수행하는 것이 매핑 실행 중에 조인을 수행하는 것보다 빠릅니다. 사용하는 데이터베이스 조인 유형이 성능에 영향을 미칠 수 있습니다. 일반 조인이 외부 조인보다 빠르며 결과 행 수도 적습니다. 서로 다른 두 데이터베이스나 플랫폼 파일 시스템의 테이블을 조인하는 경우와 같이, 데이터베이스에서 조인을 수행할 수 없는 경우도 있습니다.

**가능한 경우 정렬된 데이터를 조인하십시오.**

정렬된 입력을 사용하도록 조이너 변환을 구성합니다. 데이터 통합 서비스는 디스크 입력과 디스크 출력을 최소화하여 성능을 향상시킵니다. 대규모 데이터 집합으로 작업할 때 가장 큰 성능 향상을 얻을 수 있습니다. 비정렬 조이너 변환의 경우 행 수가 더 적은 소스를 마스터 소스로 지정하십시오.

**조인 조건을 최적화합니다.**

데이터 통합 서비스는 작은 그룹에서 행을 읽고 큰 그룹에서 일치하는 행을 찾은 다음 조인 작업을 수행하는 방식으로 한 조인 피연산자의 데이터 집합 크기를 줄입니다. 데이터 집합의 크기를 줄이면 데이터 통합 서비스가 더 이상 큰 그룹 소스에서 불필요한 행을 읽지 않게 되므로 매핑 성능이 향상됩니다. 데이터 통합 서비스는 조인 조건을 큰 그룹 소스로 이동하고 작은 그룹과 일치하는 행만 읽습니다.

**반 조인 최적화 방법을 사용합니다.**

한 입력 그룹에 다른 입력 그룹보다 훨씬 많은 행이 있으며 조인 조건을 기준으로 큰 그룹의 많은 행이 작은 그룹의 행과 일치하지 않는 경우 반 조인 최적화 방법을 사용하여 매핑 성능을 향상시킵니다.

## 조회 변환 최적화

조회 캐시 유형 및 조회 조건에 따라 조회 변환이 성능을 저하시킬 수 있습니다.

조회 변환 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**최적의 데이터베이스 드라이버를 사용합니다.**

데이터 통합 서비스는 원시 데이터베이스 드라이버나 ODBC 드라이버를 사용하여 조회 테이블에 연결할 수 있습니다. 원시 데이터베이스 드라이버가 ODBC 드라이버보다 나은 매핑 성능을 제공합니다.

**관계형 또는 플랫폼 파일 조회를 위해 조회 테이블을 캐싱합니다.**

관계형 또는 플랫폼 파일 소스의 조회 성능을 향상시키려면 변환에서 조회 캐싱을 활성화합니다. 캐싱을 활성화한 경우 데이터 통합 서비스가 조회 테이블을 캐싱합니다. 매핑을 실행하는 경우 데이터 통합 서비스가 조회 테이블 대신 조회 캐시를 쿼리합니다. 이 옵션을 활성화하지 않으면 데이터 통합 서비스가 행별로 조회 테이블을 쿼리합니다.

조회 테이블을 캐싱하는지 여부와 관계없이 조회 쿼리 및 처리 결과는 동일합니다. 하지만 작은 조회 테이블에서 조회 캐시를 사용하면 매핑 성능이 향상될 수 있습니다. 일반적으로 300MB 미만이 필요한 조회 테이블을 캐싱합니다.

**논리적 데이터 개체 조회를 위해 조회 테이블을 캐싱합니다.**

논리적 데이터 개체에서 조회 성능을 향상시키려면 데이터 통합 서비스에서 데이터 개체 캐싱을 활성화합니다. 데이터 개체 캐싱을 활성화한 경우 데이터 통합 서비스가 논리적 데이터 개체를 캐싱합니다. 데이터 개체 캐싱을 활성화하려면 매핑을 응용 프로그램에 배포하고, 논리적 데이터 개체의 캐싱을 활성화하고, `infacmd ms runmapping` 명령을 사용하여 매핑을 실행해야 합니다. 매핑을 실행하는 경우 데이터 통합 서비스가 논리적 데이터 개체 대신 데이터 개체 캐시를 쿼리합니다.

Developer tool에서 매핑을 실행하는 경우 조회 변환이 행별로 논리적 데이터 개체를 쿼리합니다.

**적절한 캐시 유형을 사용합니다.**

성능을 향상시키려면 다음과 같은 유형의 캐시를 사용하십시오.

- 공유 캐시. 여러 변환 사이에서 조회 캐시를 공유할 수 있습니다. 동일한 매핑의 변환 사이에서 명명되지 않은 캐시를 공유할 수 있습니다. 동일하거나 다른 매핑의 변환 사이에서 명명된 캐시를 공유할 수 있습니다.
- 지속형 캐시. 캐시 파일을 저장하고 재사용하려면 지속형 캐시를 사용하도록 변환을 구성할 수 있습니다. 매핑 실행 간에 조회 테이블이 바뀌지 않는 경우 이 기능을 사용합니다. 지속형 캐시를 사용하면 데이터 통합 서비스가 데이터베이스 대신 캐시 파일에서 메모리 캐시를 구축하므로 성능이 향상될 수 있습니다.

### 동시 캐시를 활성화합니다.

조희 변환을 포함하는 매핑을 처리하는 경우 데이터 통합 서비스는 캐싱된 조희 변환에서 첫 번째 데이터 행을 처리할 때 메모리에 캐시를 구축합니다. 매핑에 여러 개의 조희 변환이 있는 경우 데이터 통합 서비스는 조희 변환에 의해 첫 번째 데이터 행이 처리될 때 순차적으로 캐시를 생성합니다. 이로 인해 조희 변환 처리가 느려집니다.

동시 캐시를 활성화하여 성능을 향상시킬 수 있습니다. 추가 동시 파이프라인의 수가 1개 이상으로 설정되어 있으면 데이터 통합 서비스가 순차적이지 아니라 동시에 캐시를 구축합니다. 집계, 조이너 또는 분류기 변환 등과 같이, 완료에 시간이 걸릴 수 있는 많은 수의 활성 변환이 매핑에 포함된 경우 성능이 크게 향상됩니다. 여러 개의 동시 파이프라인을 활성화하면 데이터 통합 서비스가 더 이상 활성 매핑이 완료될 때까지 대기할 필요 없이 즉시 캐시를 구축할 수 있습니다. 파이프라인의 다른 조희 변환도 캐시를 동시에 구축합니다.

### 조희 조건 일치를 최적화합니다.

조희 변환은 조희 캐시 데이터를 조희 조건과 일치시킬 때 데이터를 정렬하고 처음 일치하는 값과 마지막 일치하는 값을 결정합니다. 이 변환이 조희 조건과 일치하는 임의 값을 반환하도록 구성할 수 있습니다. 일치하는 임의 값을 반환하도록 조희 변환을 구성하면 변환이 조희 조건과 일치하는 첫 번째 값을 반환합니다. 첫 번째 일치하는 값이나 마지막 일치하는 값을 반환하도록 구성한 경우와 달리 변환이 모든 포트를 인덱싱하지 않습니다.

변환이 모든 포트를 인덱싱하면 성능이 저하될 수 있는데, 일치하는 임의 값을 사용하면 이 작업을 하지 않으므로 성능이 향상될 수 있습니다.

### 캐싱되는 행 수를 줄입니다.

캐시에 포함되는 행 수를 줄여 성능을 향상시킬 수 있습니다. 조희 SQL 재정의 옵션을 사용하여 기본 SQL 문에 WHERE 절을 추가합니다. 동적 캐시를 사용하는 조희 변환에 WHERE 절을 추가할 경우 조희 변환 전에 필터 변환을 사용하여 WHERE 절과 일치하는 동적 캐시에 행을 전달합니다.

### ORDER BY 문의 재정의합니다.

기본적으로 데이터 통합 서비스는 캐싱된 조희에 대해 ORDER BY 문을 생성합니다. ORDER BY 문에는 모든 조희 포트가 포함됩니다. 성능을 향상시키려면 기본 ORDER BY 문을 억제하고 열 수가 더 적은 재정의 ORDER BY를 입력합니다.

재정의에 입력하는 것과 관계없이, 데이터 통합 서비스는 항상 ORDER BY 문을 생성합니다. 생성되는 ORDER BY 문을 억제하려면 ORDER BY 재정의의 다음에 대시 두 개('--')를 배치하십시오.

예를 들어 조희 변환에서 다음과 같은 조희 조건을 사용한다고 가정합니다.

```
ITEM_ID = IN_ITEM_ID  
PRICE <= IN_PRICE
```

이 조희 변환에는 매핑에 사용되는 조희 포트 3개(ITEM\_ID, ITEM\_NAME 및 PRICE)가 포함됩니다. ORDER BY 문을 입력할 때 조희 조건의 포트와 같은 순서로 열을 입력합니다. 또한, 모든 데이터베이스 예약어를 따옴표로 묶어야 합니다.

조희 SQL 재정의에 다음과 같은 조희 쿼리를 입력합니다.

```
SELECT ITEMS_DIM.ITEM_NAME, ITEMS_DIM.PRICE, ITEMS_DIM.ITEM_ID FROM ITEMS_DIM ORDER BY  
ITEMS_DIM.ITEM_ID, ITEMS_DIM.PRICE --
```

### 메모리가 더 많은 시스템을 사용합니다.

매핑 성능을 향상시키려면 많은 양의 메모리가 있는 데이터 통합 서비스 노드에서 매핑을 실행하십시오. 시스템에 무리가 없는 한도 내에서 인덱스 및 데이터 캐시 크기를 최대한 늘리십시오. 데이터 통합 서비스 노드에 충분한 메모리가 있다면 디스크에 페이징하지 않고 모든 데이터를 메모리에 유지할 수 있도록 캐시를 늘리십시오.

#### 조회 조건을 최적화합니다.

둘 이상의 조회 조건을 포함하는 경우 다음과 같은 순서로 조건을 배치하여 조회 성능을 최적화합니다.

- 같음(=)
- 보다 작음(<), 보다 큼(>), 작거나 같음(<=), 크거나 같음(>=)
- 같지 않음(!=)

#### 조회 열을 필터링합니다.

성능을 향상시키려면 필터 조건을 생성하여 조회 캐시가 구축될 때 소스에서 검색되는 조회 행 수를 줄이십시오.

#### 조회 테이블을 인덱싱합니다.

데이터 통합 서비스가 조회 조건 열에서 값을 쿼리하고 정렬하고 비교해야 합니다. 인덱스에는 조회 조건에 사용되는 모든 열이 포함되어야 합니다.

다음과 같은 유형의 조회에서 성능을 향상시킬 수 있습니다.

- 캐싱된 조회. 성능을 향상시키려면 조회 ORDER BY 문에서 열을 인덱싱합니다. 매핑 로그 파일에 ORDER BY 문이 포함됩니다.
- 캐싱되지 않은 조회. 성능을 향상시키려면 조회 조건에서 열을 인덱싱합니다. 데이터 통합 서비스는 조회 변환으로 전달되는 각 행에 대해 SELECT 문을 실행합니다.

#### 다중 조회를 최적화합니다.

매핑에 다중 조회가 포함된 경우 캐싱이 활성화되어 있고 힙 메모리가 충분해도 조회가 성능을 저하시킬 수 있습니다. 종합적인 성능을 향상시키려면 가장 많은 양의 데이터를 쿼리하는 조회 변환을 조정하십시오.

조회 테이블이 매핑의 소스 테이블과 동일한 데이터베이스에 있으며 캐싱을 사용하기 어려운 경우에는 조회 변환을 사용하는 대신 소스 데이터베이스에서 테이블을 조인합니다.

## 분류기 변환 최적화

데이터 통합 서비스 노드의 물리적 RAM이 데이터 정렬에 사용할 충분한 메모리를 할당하지 못하는 경우 분류기 변환이 성능을 저하시킬 수 있습니다.

분류기 변환 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

#### 충분한 메모리를 할당합니다.

최적의 성능을 얻으려면 분류기 캐시 크기를 데이터 통합 서비스 노드에서 사용 가능한 물리적 RAM 양과 같거나 작은 값으로 구성합니다. 분류기 변환을 사용하여 데이터를 정렬하려면 16MB 이상의 물리적 메모리를 할당하십시오. 기본적으로 분류기 캐시 크기는 16,777,216바이트로 설정됩니다. 데이터 통합 서비스가 데이터를 정렬할 수 있는 충분한 메모리를 할당하지 못하는 경우 매핑이 실패합니다.

수신 데이터의 양이 분류기 캐시 크기의 양보다 큰 경우 데이터 통합 서비스는 데이터를 분류기 변환 작업 디렉터리에 임시로 저장합니다. 작업 디렉터리에 데이터를 저장할 경우 데이터 통합 서비스에 적어도 수신 데이터 양의 두 배에 이르는 디스크 공간이 필요합니다.

# SQL 변환 최적화

데이터 통합 서비스는 매핑에서 새로운 쿼리를 처리할 때마다 **SQLPrepare**라는 함수를 호출하여 **SQL** 프로시저를 생성한 후 데이터베이스에 전달합니다. 입력 행마다 쿼리가 변경된다면 이 호출로 인해 성능이 저하될 수 있습니다.

SQL 변환 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**SQL 변환 쿼리에서 트랜잭션 문을 사용하지 마십시오.**

SQL 쿼리에 커밋 및 롤백 쿼리 문이 포함된 경우 데이터 통합 서비스가 각 커밋 및 롤백 후에 **SQL** 프로시저를 다시 생성해야 합니다. 성능을 최적화하려면 **SQL** 변환 쿼리에서 트랜잭션 문을 사용하지 마십시오.

**SQL 변환에서 초기 선택 또는 푸시인 필터 최적화 방법을 활성화합니다.**

성능을 향상시키려면 **SQL** 변환에서 초기 선택 또는 푸시인 최적화 방법을 활성화하는 것이 좋습니다.

## SQL 변환의 초기 선택 최적화

필터 조건이 통과 포트만 참조하고 **SQL** 변환에 부작용이 없는 경우 데이터 통합 서비스가 **SQL** 변환에서 초기 선택 최적화를 수행할 수 있습니다.

다음과 같은 상황에서는 **SQL** 변환에 부작용이 있습니다.

- **SQL** 쿼리가 데이터베이스를 업데이트합니다. **SQL** 쿼리에 **CREATE**, **DROP**, **INSERT**, **UPDATE**, **GRANT** 또는 **REVOKE** 문이 포함됩니다.
- 변환이 반환할 결과가 없는 **SELECT** 문에 대해 **NULL**을 반환합니다. 행이 통과 포트 값, **SQL** 오류 정보 또는 **NumRowsAffected** 필드를 포함할 수 있습니다.

## SQL 변환의 초기 선택 최적화 활성화

**SQL** 변환에 부작용이 없는 경우 **SQL** 변환에서 초기 선택 최적화를 활성화합니다.

1. **SQL** 변환 고급 속성에서 **데이터베이스 출력만 반환** 옵션을 활성화합니다.
2. 변환 고급 속성에서 **부작용 있음**을 선택 취소합니다.
3. 변환에 **NumAffectedRows** 포트가 있는 경우 해당 포트를 제거합니다.

## SQL 변환의 푸시인 최적화

푸시인 최적화에서는 데이터 통합 서비스가 매핑의 필터 변환에서 **SQL** 변환의 쿼리로 필터 논리를 푸시합니다.

**SQL** 변환에서 푸시인 최적화를 활성화할 경우 다음 규칙과 지침을 따르십시오.

- 변환 **SQL** 쿼리에 **SELECT** 문만 포함되어야 합니다.
- 변환 **SQL** 쿼리가 유효한 하위 쿼리여야 합니다.
- 필터 조건이 **SQL** 오류 또는 **NumRowsAffected** 필드를 참조할 수 없습니다.
- 출력 포트의 이름과 **SQL SELECT** 문의 열 이름이 일치해야 합니다. 필터 조건에서 출력 포트를 참조하는 경우 데이터 통합 서비스가 해당 포트 이름을 **SQL** 쿼리로 푸시합니다. 쿼리의 열이 출력 포트 이름과 일치하지 않는 경우 **SQL**에 별칭을 추가할 수 있습니다. 예를 들어 **SELECT mycolname1 AS portname1, mycolname2 AS portname2**와 같이 별칭을 추가합니다.
- 변환에 부작용이 있을 수 없습니다.



## SQL 변환의 푸시인 최적화 예제

SQL 변환이 고객 ID로 주문을 검색합니다. SQL 변환 후에 나타나는 필터 변환은 주문 액수가 1000보다 큰 행만 반환합니다.

데이터 통합 서비스가 다음과 같은 필터를 SQL 변환의 SELECT 문에 푸시합니다.

```
orderAmount > 1000
```

SQL 쿼리의 각 문이 필터를 포함하는 SELECT 문의 개별 하위 쿼리가 됩니다.

다음 쿼리 문은 원래 쿼리 문을 SELECT 문의 하위 쿼리로 보여 줍니다.

```
SELECT <customerID>, <orderAmount>, ... FROM (원래 쿼리 문) ALIAS WHERE <orderAmount> > 1000
```

SQL 쿼리에 여러 개의 문이 있는 경우 각 문이 별도의 하위 쿼리에 포함됩니다. 하위 쿼리는 WHERE 절이 포함된 동일한 구문을 갖습니다.

*customerID* 및 *orderAmount* 포트는 SQL 변환에서 출력 포트의 이름입니다. 하위 쿼리는 통과 포트, SQL 오류 또는 SQL 통계 포트를 포함하지 않습니다. SQL 변환에 여러 개의 필터를 푸시하는 경우 WHERE 절에 모든 필터가 포함됩니다.

## SQL 변환의 푸시인 최적화 활성화

SQL 변환 고급 속성 탭에서 속성을 구성하여 푸시인 최적화를 활성화합니다.

1. **부작용 있음**을 선택 취소합니다.
2. **데이터베이스 출력만 반환**을 활성화합니다.
3. **최대 출력 행 개수**를 0으로 설정합니다.
4. 푸시인 최적화를 활성화합니다.

# 변환 캐시

집계, 조이너, 조희, 순위 또는 분류기 변환을 사용하는 매핑을 실행하는 경우 데이터 통합 서비스가 메모리에 캐시를 작성하여 변환을 처리합니다. 데이터 통합 서비스는 공간이 더 필요할 경우 오버플로우 값을 디스크의 캐시 파일에 저장합니다.

변환 캐시 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**메모리에 캐시를 저장할 수 있는 충분한 공간을 할당하도록 변환을 구성합니다.**

집계, 조이너, 조희, 순위 또는 분류기 변환의 처리 시간을 향상시키려면 메모리에 캐시를 저장할 수 있는 충분한 공간을 할당하도록 변환을 구성합니다. 데이터 및 인덱스를 캐싱하는 데 필요한 양보다 많거나 같은 캐시 메모리 양을 구성하면 시스템 I/O 오버헤드가 줄어들어 성능이 향상됩니다. 데이터 통합 서비스가 캐시 파일을 디스크에 쓸 때 시스템 I/O 오버헤드로 인해 처리 시간이 증가합니다.

기본적으로 데이터 통합 서비스가 런타임 시 캐시 메모리 요구 사항을 자동으로 구성합니다. 자동 캐시 모드로 매핑을 실행한 후 변환에 대한 캐시 크기를 조정할 수 있습니다. 매핑 로그에서 변환 통계를 분석하여 메모리에서 변환 처리에 필요한 캐시 크기를 결정합니다. 매핑 로그에 지정된 값을 사용하도록 캐시 크기를 구성하면 할당된 메모리가 낭비되지 않도록 할 수 있습니다. 그러나 최적의 캐시 크기는 소스 데이터 크기에 따라 달라집니다. 후속 매핑이 실행된 다음 매핑 로그를 검토하여 캐시 크기에 대한 변경 사항을 모니터링합니다. 재사용 가능한 변환에 대해 특정 캐시 크기를 구성하는 경우 캐시 크기가 매핑의 각 변환 사용에 대해 최적인지 확인합니다.

## 변환 오류 제거

변환 오류가 데이터 통합 서비스의 성능을 저하시키는 경우가 많습니다. 변환 오류가 발생할 때마다 오류의 원인을 확인하고 오류를 발생시킨 행을 데이터 흐름에서 제거하기 위해 데이터 통합 서비스가 일시 중지됩니다. 데이터 통합 서비스는 일반적으로 데이터 통합 서비스 로그의 매핑 로그 파일에 행을 씁니다.

변환 오류 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**매핑 로그 파일을 검사하여 변환 오류가 발생한 위치를 확인하고 해당 변환 제약 조건을 평가합니다.**

데이터 통합 서비스에서 다른 변환 오류, 충돌하는 매핑 논리 및 오류로 설정된 모든 조건(예: null 입력)이 발생하면 변환 오류가 발생합니다. 매핑 로그 파일을 검사하여 변환 오류가 발생한 위치를 확인합니다. 오류가 특정 변환에서 집중적으로 발생한다면 해당 변환 제약 조건을 평가합니다.

**더 낮은 추적 수준을 구성합니다.**

많은 수의 변환 오류를 생성하는 매핑을 실행해야 한다면 더 낮은 추적 수준을 설정하여 성능을 향상시킬 수 있습니다. 하지만 이 방법은 변환 오류에 대한 장기적인 해결 방법이 될 수 없습니다.

## 변환 부작용

변환이 행을 반환하고 개체를 수정하는 경우 또는 다른 개체나 함수와 상호 작용하는 경우 변환에 부작용이 있습니다. 변환이 데이터베이스를 수정하거나, 합계에 더하거나, 예외를 발생시키거나, 전자 메일을 작성하거나, 부작용이 있는 다른 함수를 호출할 수 있습니다.

데이터 통합 서비스는 매핑을 최적화하기 전에 부작용이 있는 변환을 식별합니다. 데이터 통합 서비스는 변환에 부작용이 있는지 확인할 수 없는 경우 부작용이 있다고 가정합니다.

부작용이 있는 변환은 데이터 통합 서비스가 매핑을 최적화할 수 있는 시기를 제한합니다. 데이터 통합 서비스가 부작용이 있는 변환에 초기 선택, 분기 잘라내기, 글로벌 조건자 최적화 및 푸시인 최적화를 적용하면 매핑 결과가 바뀝니다. 초기 선택 및 푸시인 최적화에서는 필터 논리가 필터 변환에서 소스에 최대한 가깝게 이동됩니다. 필터가 부작용이 있는 함수 이전에 실행되면 매핑 결과가 변경됩니다.

예를 들어 변환이 고객 ID를 받아 주문 정보를 포함하는 행을 반환한다고 가정합니다. 또한, 이 변환은 파일에 주문을 씁니다. 변환이 주문을 파일에 쓰기 전에 데이터 통합 서비스가 필터 최적화를 적용하는 경우 필터가 매핑 후에 실행될 때보다 파일이 적은 수의 행을 받게 됩니다. 이 변환의 부작용은 주문 레코드를 파일에 쓰는 기능입니다.

다음과 같은 변환에 부작용이 있습니다.

- SQL 변환, 웹 서비스 소비자 변환 및 Java 변환(부작용 속성을 비활성화하지 않은 경우)
- ABORT() 또는 ERROR() 함수를 호출하거나, 전자 메일을 보내거나, 저장 프로시저를 호출하는 변환
- 파일 또는 데이터베이스에 쓰는 변환
- 변수 포트를 통해 개수를 유지하는 변환. 예: COUNT=COUNT+1

SQL 변환, 웹 서비스 소비자 변환 및 Java 변환에는 기본적으로 부작용이 있습니다. 부작용 없이 행을 처리하도록 변환을 구성하려면 **고급 속성**에서 **부작용 있음** 속성을 비활성화할 수 있습니다. 변환에 부작용이 없는 경우에는 변환에서 추가 속성을 구성하여 최적화를 활성화할 수 있습니다.

# 웹 서비스 소비자 변환 최적화

매핑이 웹 서비스를 여러 번 호출하는 경우 웹 서비스 소비자 변환이 성능을 저하시킬 수 있습니다.

웹 서비스 소비자 변환 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

## 쿠키 인증을 사용하도록 웹 서비스 소비자 변환을 구성합니다.

원격 웹 서비스 서버는 쿠키를 기반으로 웹 서비스 소비자 사용자를 추적합니다. 매핑이 웹 서비스를 여러 번 호출하는 경우 성능을 향상시킬 수 있습니다.

쿠키 포트를 웹 서비스 요청 메시지에 연결하면 웹 서비스 공급자가 응답 메시지에 쿠키 값을 반환합니다. 쿠키 값을 매핑의 다른 변환 다운스트림에 전달하거나 파일에 쿠키 값을 저장할 수 있습니다. 쿠키 값을 파일에 저장할 때 쿠키를 웹 서비스 소비자 변환의 입력으로 구성할 수 있습니다. 쿠키 출력 포트를 웹 서비스 소비자 변환 출력 그룹 중 하나에 연결할 수 있습니다.

## 웹 서비스 소비자 변환에서 초기 선택 또는 푸시인 필터 최적화 방법을 활성화합니다.

성능을 향상시키기 위해 데이터 통합 서비스가 웹 서비스 소비자 변환에 초기 선택 또는 푸시인 최적화 방법을 적용할 수 있습니다. 초기 선택 최적화를 적용하려면 웹 서비스에 부작용이 없어야 하며 웹 서비스가 결함을 오류로 취급할 수 없습니다. 푸시인 최적화를 적용하려면 웹 서비스에 부작용이 없어야 하고, 웹 서비스가 결함을 오류로 취급할 수 없으며, 필터 조건이 통과 포트를 참조해야 합니다.

웹 서비스가 웹 서비스 소비자 변환에 응답을 반환하는 외에 다른 기능을 수행하는 경우 웹 서비스에 부작용이 있습니다. 웹 서비스가 데이터베이스를 수정하거나, 파일에 쓰거나, 전자 메일을 작성하거나, 개수를 업데이트하거나, 부작용이 있는 다른 웹 서비스를 호출하는 경우 웹 서비스에 부작용이 있습니다.

## 여러 요청을 동시에 보내도록 웹 서비스 소비자 변환을 구성합니다.

매핑 성능을 개선하려면 여러 요청을 동시에 보내도록 웹 서비스 소비자 변환을 구성합니다. 웹 서비스 소비자 변환을 활성화하여 웹 서비스에 대한 여러 동시 연결을 생성할 때 메모리 소비 제한 및 동시 연결 수 제한을 설정할 수 있습니다.

## 웹 서비스 소비자 변환의 초기 선택 최적화

데이터 통합 서비스는 웹 서비스 소비자 변환에 초기 선택 최적화 방법을 적용할 때 매핑에서 웹 서비스 소비자 변환 이전에 있는 필터 조건을 최대한 소스에 가깝게 이동합니다.

## 웹 서비스 소비자 변환의 초기 선택 최적화 활성화

웹 서비스 소비자 변환에 부작용이 없고 변환이 결함을 오류로 처리하지 않을 경우 변환에 대한 초기 선택 최적화를 활성화합니다.

1. 웹 서비스 소비자 변환의 **고급 속성** 보기를 엽니다.
2. **결함을 오류로 처리**를 선택 취소합니다.
3. **부작용 있음**을 선택 취소합니다.

## 웹 서비스 소비자 변환의 푸시인 최적화

변환이 SQL 데이터 서비스의 가상 테이블에 있는 경우 웹 서비스 소비자 변환을 사용하여 푸시인 최적화를 구성할 수 있습니다.

매핑은 웹 서비스를 호출하여 최종 사용자 SQL 쿼리의 문에 따라 데이터 집합 또는 데이터의 하위 집합을 검색합니다. 최종 사용자 SQL 쿼리에는 선택적 필터 조건이 포함됩니다.

웹 서비스 소비자 변환은 푸시인 최적화를 통해 필터 포트의 필터 값을 검색합니다. 필터 포트는 푸시인 최적화를 구성할 때 필터 포트로 인식하는 연결되지 않은 입력 포트입니다. 필터 포트에는 최종 사용자 쿼리에 필터가 포함되지 않을 경우 웹 서비스가 모든 행을 반환하도록 하는 기본값이 있습니다. 필터 포트는 통과 포트가 아닙니다.

**참고:** 필터 필드는 웹 서비스 요청에서 루트 그룹의 일부여야 합니다.

필터 포트를 구성할 때는 웹 서비스 소비자 변환에서 웹 서비스 응답의 열 데이터를 수신할 출력 포트를 식별해야 합니다. 예를 들어 필터 포트가 **EmployeeID**라는 입력 포트일 경우 응답의 출력 포트는 **EmployeeNum**이라는 포트가 될 수 있습니다. **Developer tool**에서 가상 테이블 읽기의 필터 논리를 웹 서비스 소비자 요청으로 푸시하려면 입력 필터 포트와 출력 포트가 연결되어야 합니다. 일반적으로 웹 서비스 요청에 대한 입력 포트는 웹 서비스 응답의 출력 포트와 다릅니다.

필터 필드는 통과 포트가 될 수 없습니다. 필터 포트를 구성하면 포트의 기본값이 필터 조건의 값으로 변경되므로 통과 출력 포트 값도 변경됩니다. 출력 통과 포트에 기반하는 필터는 예기치 않은 결과를 반환합니다.

여러 필터 식을 웹 서비스 소비자 변환에 푸시할 수 있습니다. 각 필터 조건은 다음 형식이어야 합니다.

<필드> = <상수>

필터 조건은 AND로 조인되어야 합니다. 조건을 OR로 조인할 수 없습니다.

## 웹 서비스 소비자 변환의 푸시인 최적화 예제

SQL 데이터 서비스는 사용자로부터 수신한 SQL 쿼리에 따라 모든 고객에 대한 주문을 반환하거나 특정 고객에 대한 주문을 반환합니다.

데이터 서비스에는 다음 구성 요소의 논리적 데이터 개체가 포함됩니다.

### Customer 테이블

고객 정보가 포함된 Oracle 데이터베이스 테이블입니다.

### 웹 서비스 소비자 변환

웹 서비스를 호출하여 고객의 최신 주문을 검색하는 변환입니다. 웹 서비스 소비자 변환에는 **customerID** 및 **orderNum**에 대한 입력 포트가 있습니다. 이 변환에는 **Customer** 테이블로부터 수신한 고객 데이터가 포함된 통과 포트가 있습니다. **orderNum** 포트는 필터 포트이고 연결되지 않습니다. **orderNum**의 기본값은 "\*"입니다. 웹 서비스에서 웹 서비스 요청을 통해 이 값을 수신할 경우 모든 주문이 반환됩니다.

### Orders 가상 테이블

웹 서비스의 고객 및 주문 데이터를 수신하는 가상 테이블입니다. 최종 사용자는 이 테이블을 쿼리합니다.

**Orders**에는 고객 열, **orderID** 열 및 고객 및 주문 데이터가 포함됩니다.

최종 사용자는 다음 SQL 쿼리를 SQL 데이터 서비스에 전달합니다.

```
SELECT * from OrdersID where customer = 23 and orderID = 56
```

데이터 통합 서비스가 쿼리를 분할하여 매핑을 최적화합니다. 데이터 통합 서비스는 초기 선택 최적화를 사용하여 필터 논리 **customer = 23**을 **Customer** 테이블 읽기로 이동합니다. 데이터 통합 서비스는 푸시인 최적화를 사용하여 필터 논리 **orderID = 56**을 웹 서비스 소비자 변환 필터 포트에 푸시합니다. 웹 서비스 소비자 변환이 고객 **23**에 대한 **ordersID 56**을 검색합니다.

## 웹 서비스 소비자 변환의 푸시인 최적화 활성화

웹 서비스 소비자 변환에 부작용이 없고 변환이 결함을 오류로 처리하지 않을 경우 변환에 대한 푸시인 최적화를 활성화합니다.

1. 웹 서비스 소비자 변환의 **고급 속성** 보기를 엽니다.
2. **결함을 오류로 처리**를 선택 취소합니다.

3. **부작용 있음**을 선택 취소합니다.
4. **푸시인 최적화** 속성에서 **열기** 단추를 클릭합니다.
5. 입력 최적화 대화 상자에서 필터 포트 이름을 선택합니다.  
여러 필터 포트를 선택할 수 있습니다.
6. **출력** 열을 클릭합니다.
7. 각 필터 포트에 대해 필터링된 열을 웹 서비스 응답에 포함하는 출력 포트를 선택합니다.
8. 각 필터 포트에 대한 기본값을 입력합니다.  
**참고:** 웹 서비스 소비자 포트가 필터 포트가 아닌 경우 기본값을 구성할 수 없습니다.

## 제 5 장

# 매핑 최적화

이 장에 포함된 항목:

- [매핑 최적화 개요, 38](#)
- [최적화 방법, 39](#)
- [푸시다운 최적화, 45](#)
- [단일 패스 읽기, 47](#)
- [필터 최적화, 47](#)
- [데이터 유형 변환 최적화, 48](#)
- [오류 추적, 48](#)

## 매핑 최적화 개요

데이터 통합 서비스가 데이터를 효율적으로 변환하고 이동할 수 있게 하려면 매핑을 최적화하십시오. 매핑 수준 최적화는 구현에 시간이 걸릴 수 있지만 매핑 성능을 상당히 향상시킬 수 있습니다.

최적화 태스크는 일반 매핑, 논리적 데이터 개체 읽기 및 쓰기 매핑, 가상 테이블 매핑, 그리고 작업 매핑에 적용됩니다. 대상과 소스를 최적화한 후에는 매핑 수준 최적화에 집중하십시오.

매핑을 최적화하기 위해 다음과 같은 태스크를 수행할 수 있습니다.

- 최대한 많은 작업을 수행할 수 있는 최소 개수의 변환과 식으로 매핑을 구성합니다.
- 변환 간의 불필요한 링크를 삭제하여 이동되는 데이터 양을 최소화합니다.
- 데이터 통합 서비스가 매핑에 적용할 수 있는 최적화 방법을 결정하는 최적화 수준을 선택합니다. 매핑을 최적화할 때 데이터 통합 서비스는 처리할 데이터 양을 줄이려고 시도합니다. 예를 들어 데이터 통합 서비스는 초기 선택 최적화를 사용하여 필터를 소스에 더 가깝게 이동할 수 있고, 비용 기반 최적화 방법을 사용하여 조인 처리 순서를 변경할 수 있습니다.
- 데이터 통합 서비스가 소스 데이터베이스에 일부 또는 전체 변환 논리를 푸시다운할 수 있는지 여부를 결정할 수 있게 하려면 푸시다운 유형을 선택합니다.
- 데이터 통합 서비스가 논리적 데이터 개체를 캐싱하고 매핑을 실행할 때 미리 만들어진 논리적 데이터 개체에 액세스할 수 있도록 데이터 개체 캐싱을 구성합니다. 기본적으로 데이터 통합 서비스는 매핑을 실행할 때 소스 데이터를 추출하여 필요한 데이터 개체를 만듭니다. 데이터 통합 서비스가 미리 만들어진 데이터 개체에 액세스할 수 있으면 매핑 성능이 향상됩니다.
- SQL 변환, 웹 서비스 소비자 변환 및 Java 변환을 구성할 때 해당 변환에 부작용이 없는지 여부를 지정합니다. 일부 변환에는 최적화를 제한하는 부작용이 있습니다. 예를 들어 파일 또는 데이터베이스에 쓰거나, 개수를 늘리거나, 예외를 발생시키거나, 전자 메일을 쓰는 변환에는 부작용이 있을 수 있습니다. 대부분의 경우 데이터 통합 서비스가 최적화를 제한하는 부작용이 있는 변환을 식별합니다.

관련 항목:

- [“데이터 개체 캐싱” 페이지 58](#)

## 최적화 방법

데이터 통합 서비스는 맵에서 처리할 행 수를 줄이기 위해 최적화 방법을 적용합니다. 맵의 최적화 수준을 구성하여 데이터 통합 서비스가 적용하는 최적화 방법을 제한할 수 있습니다.

데이터 통합 서비스는 다음과 같은 최적화 방법을 적용할 수 있습니다.

- 푸시다운 최적화
- 초기 예측 최적화
- 초기 선택 최적화
- 분기 잘라내기 최적화
- 푸시인 최적화
- 조건자 최적화
- 글로벌 조건자 최적화
- 비용 기반 최적화
- 데이터십 조인 최적화
- 반 조인 최적화

데이터 통합 서비스가 한 맵에 동시에 여러 최적화 방법을 적용할 수 있습니다. 예를 들어, 보통 최적화 수준을 선택하는 경우 데이터 통합 서비스가 초기 예측 최적화, 조건자 최적화, 글로벌 조건자 최적화, 분기 잘라내기 최적화 및 초기 선택 최적화 또는 푸시인 최적화 방법을 적용합니다.

## 최적화 수준

데이터 통합 서비스는 사용자가 구성한 최적화 수준을 기반으로 맵을 최적화합니다. 기본값 이외의 최적화 수준을 사용하는 맵이 필요한 경우 최적화 수준 속성을 구성합니다.

다음과 같은 최적화 수준 중 하나를 선택할 수 있습니다.

### 자동

데이터 통합 서비스가 실행 모드와 맵 콘텐츠에 기반하여 최적화를 적용합니다.

### 없음

데이터 통합 서비스가 최적화를 적용하지 않습니다.

### 최소

데이터 통합 서비스가 초기 예측 최적화 방법을 적용합니다.

### 일반

데이터 통합 서비스가 초기 예측, 초기 선택, 분기 잘라내기, 푸시인, 글로벌 조건자 최적화 및 조건자 최적화 방법을 적용합니다.

### 전체

데이터 통합 서비스가 비용 기반, 초기 예측, 초기 선택, 분기 잘라내기, 조건자, 푸시인, 반 조인 및 데이터십 조인 최적화 방법을 적용합니다.

기본값은 자동입니다.

**실행** 메뉴나 **Developer tool**의 매핑 편집기에서 매핑을 실행하면 데이터 통합 서비스가 보통 최적화 수준을 적용합니다. **실행** 메뉴에서 매핑을 실행하면 데이터 통합 서비스가 매핑 구성의 최적화 수준을 적용합니다. 명령줄에서 매핑을 실행하면 데이터 통합 서비스가 응용 프로그램의 매핑 배포 속성에 있는 최적화 수준을 적용합니다.

**참고:** 데이터 통합 서비스가 최적화 수준이 있는 푸시다운 최적화 방법을 적용하지 않습니다. 매핑 런타임 속성의 매핑에 대해 푸시다운 최적화를 구성할 수 있습니다.

## 필터 최적화

필터 최적화는 매핑을 통과하는 행 수를 줄여 성능을 향상시킵니다. 데이터 통합 서비스에서는 초기 선택 최적화 또는 푸시인 최적화를 적용할 수 있습니다.

데이터 통합 서비스는 필터 최적화 방법을 적용할 때 매핑에서 가능한 소스에 근접한 위치로 필터를 이동합니다. 데이터 통합 서비스가 매핑에서 변환 앞으로 필터를 이동할 수 없는 경우에는 필터 논리를 변환으로 푸시할 수 있습니다.

## 초기 예측 최적화 방법

데이터 통합 서비스는 초기 예측 최적화 방법을 적용할 때 사용되지 않는 포트를 식별하여 해당 포트 간의 링크를 제거합니다.

초기 예측 최적화 방법은 변환 간에서 데이터 통합 서비스를 통해 이동되는 데이터의 양을 줄여 성능을 향상시킵니다. 데이터 통합 서비스는 매핑을 처리할 때 매핑의 모든 연결된 포트에서 제공되는 데이터를 한 변환에서 다른 변환으로 이동합니다. 대규모의 복잡한 매핑이나 중첩된 맵셋을 사용하는 매핑에서 일부 포트가 대상에 데이터를 제공하지 않을 수 있습니다. 데이터 통합 서비스는 대상에 데이터를 제공하지 않는 포트를 식별합니다. 이러한 사용되지 않는 포트를 식별한 후 데이터 통합 서비스는 매핑에서 모든 사용되지 않는 포트 사이의 링크를 제거합니다.

데이터 통합 서비스가 모든 링크를 제거하는 것은 아닙니다. 예를 들어 다음과 같은 링크는 제거하지 않습니다.

- 부작용이 있는 변환에 연결된 링크
- **ABORT()** 또는 **ERROR()** 함수를 호출하거나, 전자 메일을 보내거나, 저장 프로시저를 호출하는 변환에 연결된 링크

변환의 모든 포트가 사용되지 않음을 확인한 경우 데이터 통합 서비스는 가장 작은 데이터가 포함된 포트의 링크를 제외한 모든 변환 링크를 제거합니다. 하지만 매핑에서 사용되지 않는 변환을 제거하지는 않습니다.

개발자 도구는 기본적으로 이 최적화 방법을 활성화합니다.

## 조건자 최적화 방법

데이터 통합 서비스는 조건자 최적화 방법을 적용할 때 매핑이 생성하는 조건자 식을 검사한 후, 식을 간소화하거나 다시 작성하여 매핑 성능을 향상시킬 수 있는지 여부를 결정합니다.

데이터 통합 서비스는 매핑을 실행할 때 매핑 소스를 기준으로 쿼리를 생성하고 매핑 논리와 매핑 내 변환을 기반으로 쿼리 결과에 대한 연산을 수행합니다. 쿼리와 연산에 조건자 식이 포함되는 경우가 많습니다. 조건자 식은 데이터가 충족해야 하는 조건을 나타냅니다. 필터 변환과 조이너 변환의 필터 조건과 조인 조건이 조건자 식의 예입니다.

또한 조건자 최적화 방법에서 데이터 통합 서비스는 매핑에서 가능한 한 이른 시기에 조건자 식을 적용하여 매핑 성능을 향상시키려 합니다.

데이터 통합 서비스는 기존 조건자 식에서 관계를 유추하고 새 조건자 식을 생성합니다. 예를 들어 매핑에 조인 조건이 "**A=B**"인 조이너 변환과 필터 조건이 "**A>5**"인 필터 변환이 포함된다고 가정합니다. 이 경우 데이터 통합 서비스가 조인 조건에 "**B>5**" 조건을 추가할 수 있습니다.



조건자 최적화 방법과 초기 선택 최적화 방법을 모두 매핑에 적용할 수 있는 경우 데이터 통합 서비스가 두 방법을 함께 적용됩니다. 예를 들어 데이터 통합 서비스는 조건자 최적화 방법을 통해 새 필터 조건을 생성하는 동시에 초기 선택 방법을 통해 매핑의 업스트림으로 새 조건을 이동하려고 시도합니다. 두 최적화 방법을 모두 적용하면 한 방법만 적용하는 것에 비해 매핑 성능이 향상됩니다.

조건자 최적화 방법을 적용하여 성능이 향상되는 경우 데이터 통합 서비스가 조건자 최적화 방법을 적용합니다. 응용 프로그램에서 매핑 결과를 변경하거나 이 방법으로 매핑 성능이 저하되는 경우에는 데이터 통합 서비스가 이 방법을 적용하지 않습니다. 데이터 통합 서비스는 기본적으로 이 최적화 방법을 적용합니다.

## 조건자 최적화 규칙 및 지침

데이터 통합 서비스는 조건자 식을 다시 작성할 때 수학적 논리를 적용하여 식을 최적화합니다.

데이터 통합 서비스가 다음과 같은 작업 중 일부 또는 전부를 수행할 수 있습니다.

- 매핑의 전체 조건자 식에서 동일한 변수를 식별한 후 이 동일성을 기반으로 단순화된 식을 생성합니다.
- 매핑의 전체 조건자 식에서 중복되는 조건자를 식별하여 제거합니다.
- 선언적 절에서 하위 식을 추출하고 이러한 하위 식을 기반으로 여러 개의 단순화된 식을 생성합니다.
- 조건자 식을 정규화합니다.
- 매핑에서 가능한 한 이른 시기에 조건자 식을 적용합니다.

연결된 포트 사이에서 데이터 유형이 불일치하는 변환이 매핑에 포함된 경우 데이터 통합 서비스가 매핑에 조건자 최적화를 적용하지 않을 수 있습니다.

다음 조건 중 하나라도 일치하는 경우 데이터 통합 서비스가 변환에 조건자 최적화를 적용하지 않을 수 있습니다.

- 변환에 연결된 포트에 대한 명시적 기본값이 포함된 경우
- 변환에 부작용이 있는 경우
- 변환이 조건자 이동을 허용하지 않는 경우. 예를 들어 부작용이 있는 변환에 이 제한 사항이 있을 수 있습니다.

개발자 도구는 기본적으로 조건자 최적화 방법을 활성화합니다.

## 비용 기반 최적화 방법

비용 기반 최적화를 사용하면 데이터 통합 서비스가 매핑을 평가하고, 의미적으로 동등한 매핑을 생성한 다음, 가능한 최상의 성능으로 매핑을 실행합니다. 비용 기반 최적화는 인접한 내부 조인 및 전체 외부 조인 작업을 수행하는 매핑에 대한 런타임을 줄여줍니다.

의미적으로 동등한 매핑은 동일한 기능을 수행하고 동일한 결과를 생성하는 매핑입니다. 의미적으로 동등한 매핑을 생성하기 위해 데이터 통합 서비스는 원래 매핑을 조각으로 분할한 다음, 최적화할 수 있는 매핑 조각을 결정합니다.

최적화하는 동안 데이터 통합 서비스가 조각 내에서 변환을 추가하거나, 제거하거나, 순서를 다시 지정할 수 있습니다. 데이터 통합 서비스는 최적화된 조각이 원래 조각과 동일한 결과를 생성하는지 확인한 후 최적화된 조각을 사용하는 대체 매핑을 형성합니다.

또한 데이터 통합 서비스는 정렬된 병합 조인 성능이 중첩 루프 조인 성능보다 높다고 판단할 경우 정렬된 병합 조인을 적용할 수도 있습니다. 정렬된 병합 조인은 정렬 순서를 사용하여 조인을 수행하기 전에 2개의 데이터 집합을 정렬합니다. 중첩 루프 조인은 중첩 루프를 사용하여 2개의 데이터 집합을 조인합니다. 데이터 통합 서비스는 소스의 정렬 정보를 사용하거나 데이터 정렬 비용이 중첩 루프 조인을 처리하는 비용보다 낮은 경우 분류기 변환을 작성합니다.

모든 매핑 또는 거의 모든 매핑에 대해 원래 매핑과 의미적으로 동등한 매핑을 생성합니다. 데이터 통합 서비스는 프로파일링 통계나 데이터베이스 통계를 사용하여 원래 매핑과 각각의 대체 매핑에 대한 비용을 계산합니다.

그런 다음, 가장 빠르게 실행되는 매핑을 식별합니다. 데이터 통합 서비스는 최상의 대체 매핑에 대한 유효성 검사 확인을 수행하여 대체 매핑이 유효하며 원래 매핑과 동일한 결과를 생성하는지 확인합니다.

데이터 통합 서비스가 최상의 대체 매핑을 메모리에 캐싱합니다. 이제 매핑을 실행하면 데이터 통합 서비스가 대체 매핑을 검색하여 원래 매핑 대신 실행합니다.

Developer tool은 기본적으로 이 방법을 활성화하지 않습니다.

## 데이터십 조인 최적화 방법

데이터십 조인 최적화 방법은 더 큰 데이터 집합 옆의 더 작은 데이터 집합을 찾아 조인 처리 시간을 단축하려고 시도합니다. 데이터 통합 서비스는 두 테이블 간에 상당한 크기 차이가 있을 경우 데이터십 조인 최적화 방법을 적용하려고 시도합니다.

예를 들어 데이터 통합 서비스는 데이터십 조인 최적화 방법을 적용하여 10,000개의 행이 포함된 마스터 테이블을 1,000,000개의 행이 포함된 세부 테이블과 조인할 수 있습니다. 데이터십 조인을 수행하기 위해 데이터 통합 서비스는 더 큰 세부 테이블이 포함된 데이터베이스에 임시 준비 테이블을 작성합니다. 그런 다음 데이터 통합 서비스는 더 작은 마스터 테이블을 임시 테이블에 복사하고 임시 테이블의 데이터를 더 큰 세부 테이블의 데이터와 조인합니다. 데이터 통합 서비스가 조인 작업을 수행한 후 조이너 변환 논리가 데이터베이스에서 처리됩니다.

이 데이터십 조인 최적화 방법을 적용하기 전에, 데이터 통합 서비스가 분석을 수행하여 데이터십 조인 최적화가 가능하며 효과가 있는지 확인합니다. 분석에서 이 방법이 성능을 향상시킬 수 있다고 확인되면 데이터 통합 서비스가 매핑에 이 최적화 방법을 적용합니다. 그런 다음, 데이터 통합 서비스가 매핑을 다시 분석하여 데이터십 조인 최적화를 추가로 적용할 수 있는지 여부를 확인합니다. 해당하는 경우 추가적인 최적화를 수행합니다.

Developer tool은 기본적으로 이 방법을 활성화하지 않습니다.

## 성능 향상을 위한 데이터십 조인 요구 사항

데이터십 조인 최적화 방법이 항상 성능을 향상시키는 것은 아닙니다. 다음과 같은 요인이 데이터십 조인 최적화가 포함된 매핑 성능에 영향을 미칩니다.

- 조이너 변환의 마스터 소스에 세부 소스보다 훨씬 적은 수의 행이 있어야 합니다.
- 세부 소스가 최적화에 적합할 정도로 상당히 커야 합니다. 세부 소스가 충분히 크지 않으면 데이터 통합 서비스가 데이터십 조인 최적화 방법을 적용하지 않고 마스터와 세부 소스에서 모든 데이터를 읽는 것이 더 빠르다는 것을 확인합니다.

## 데이터십 조인 최적화 규칙 및 지침

조이너 변환이 다음과 같은 요구 사항을 충족하는 경우 데이터 통합 서비스가 변환에 데이터십 조인 최적화를 적용할 수 있습니다.

- 조인 유형이 일반, 마스터 외부 또는 세부 외부여야 합니다.
- 세부 파이프라인이 관계형 소스에서 제공되어야 합니다.
- 매핑이 대상 기반 커밋을 사용하는 경우 조이너 변환 범위가 모든 입력이어야 합니다.
- 마스터 파이프라인과 세부 파이프라인은 어떠한 변환도 공유할 수 없습니다.
- 매핑이 세부 소스와 조이너 변환 사이의 분기를 포함할 수 없습니다.
- 조인의 세부 측이 포함된 데이터베이스가 유니코드 인코딩을 지원하지 않는 IBM DB2 데이터베이스인 경우 데이터 통합 서비스가 데이터십 조인 최적화 방법을 적용하지 못합니다.

## 반 조인 최적화 방법

반 조인 최적화 방법에서는 매핑의 조인 작업을 수정하여 소스의 데이터 추출 양을 줄입니다.

한 입력 그룹에 다른 입력 그룹보다 훨씬 많은 행이 있으며 조인 조건을 기준으로 큰 그룹의 많은 행이 작은 그룹의 행과 일치하지 않는 경우 데이터 통합 서비스가 조이너 변환에 반 조인 최적화 방법을 적용합니다. 데이터 통합 서비스는 작은 그룹에서 행을 읽고 큰 그룹에서 일치하는 행을 찾은 다음 조인 작업을 수행하는 방식으로 반 조인 피연산자의 데이터 집합 크기를 줄입니다. 데이터 집합의 크기를 줄이면 데이터 통합 서비스가 더 이상 큰 그룹 소스에서 불필요한 행을 읽지 않게 되므로 매핑 성능이 향상됩니다. 데이터 통합 서비스는 조인 조건을 큰 그룹 소스로 이동하고 작은 그룹과 일치하는 행만 읽습니다.

이 반 조인 최적화 방법을 적용하기 전에, 데이터 통합 서비스가 분석을 수행하여 반 조인 최적화가 가능하며 효과가 있는지 확인합니다. 분석에서 이 방법이 성능을 향상시킬 수 있다고 확인되면 데이터 통합 서비스가 매핑에 이 최적화 방법을 적용합니다. 그런 다음, 데이터 통합 서비스가 매핑을 다시 분석하여 반 조인 최적화를 추가로 적용할 수 있는지 여부를 확인합니다. 해당하는 경우 추가적인 최적화를 수행합니다.

Developer tool은 기본적으로 이 방법을 활성화하지 않습니다.

### 성능 향상을 위한 반 조인 최적화 요구 사항

반 조인 최적화 방법이 항상 성능을 향상시키는 것은 아닙니다. 다음과 같은 요인이 반 조인 최적화가 포함된 매핑 성능에 영향을 미칩니다.

- 조이너 변환의 마스터 소스에 세부 소스보다 훨씬 적은 수의 행이 있어야 합니다.
- 세부 소스가 충분히 커 최적화에 적합해야 합니다. 데이터 통합 서비스가 반 조인 최적화를 적용하면 매핑 처리에 이로 인한 오버헤드 시간이 추가됩니다. 세부 소스가 작으면 반 조인 방법을 적용하는 데 걸리는 시간이 세부 소스의 모든 행을 처리하는 데 걸리는 시간보다 길어질 수 있습니다.
- 일반 조인 작업과 반 조인 작업의 시간 요구 사항을 정확하게 비교하기 위해서 데이터 통합 서비스가 조이너 변환에 대한 소스 행 수 통계를 구할 수 있어야 합니다.

### 반 조인 최적화 규칙 및 지침

조이너 변환이 다음과 같은 요구 사항을 충족하는 경우 데이터 통합 서비스가 변환에 반 조인 최적화를 적용할 수 있습니다.

- 조인 유형이 일반, 마스터 외부 또는 세부 외부여야 합니다. 조이너 변환은 전체 외부 조인을 수행할 수 없습니다.
- 세부 파이프라인이 관계형 소스에서 제공되어야 합니다.
- 조인 조건이 유효한 정렬-병합-조인 조건이어야 합니다. 즉, 각 절이 한 마스터 포트와 한 세부 포트에 대한 같은 조건이어야 합니다. 여러 절이 있는 경우에는 AND로 조인해야 합니다.
- 매핑이 대상 기반 커밋을 사용하지 않는 경우 조이너 변환 범위가 모든 입력이어야 합니다.
- 마스터 파이프라인과 세부 파이프라인은 어떠한 변환도 공유할 수 없습니다.
- 매핑이 세부 소스와 조이너 변환 사이의 분기를 포함할 수 없습니다.

## 초기 선택 최적화 방법

데이터 통합 서비스는 초기 선택 최적화 방법을 적용할 때 매핑에서 필터 변환을 분할하거나, 이동하거나, 제거합니다. 필터를 매핑 위쪽으로 소스에 더 가깝게 이동합니다.

필터 조건이 결합(conjunction)인 경우 데이터 통합 서비스가 필터 변환을 분할할 수 있습니다. 예를 들어 데이터 통합 서비스가 필터 조건 "A>100 AND B<50"을 더 단순한 두 개의 조건 즉, "A>100" 조건과 "B<50" 조건으로 분할할 수 있습니다. 데이터 통합 서비스가 필터를 분할할 때 단순화된 필터를 매핑 파이프라인 위쪽으로 소스에 더 가깝게 이동합니다. 또한 필터를 분할할 때 필터를 개별적으로 파이프라인 위쪽으로 이동합니다.

Developer tool에서 보통 또는 전체 최적화 수준을 선택하면 기본적으로 초기 선택 최적화 방법이 활성화됩니다. 필터 변환 앞에 나타나는 변환에 부작용이 있는 경우 데이터 통합 서비스가 초기 선택 최적화를 무시합니다. 데이터 통합 서비스는 SQL 변환, 웹 서비스 소비자 변환 및 Java 변환에 부작용이 있는지 여부를 결정할 수 없습니다. 부작용이 없는 경우 이러한 변환에 대한 초기 선택 최적화를 구성할 수 있습니다.

최적화가 성능을 향상시키지 않는 경우 초기 선택을 비활성화할 수 있습니다. 데이터 통합 서비스는 기본적으로 이 최적화 방법을 활성화합니다.

## 글로벌 조건자 최적화 방법

데이터 통합 서비스가 글로벌 조건자 최적화 방법을 사용하는 경우 매핑에서 가능한 한 이른 시기에 필터링 가능한 해당 행을 제거합니다. 따라서 매핑이 처리해야 하는 행 수가 감소합니다. 글로벌 조건자 최적화 방법에는 조건자 최적화 및 초기 선택 방법 모두가 포함됩니다.

예를 들어 매핑에 조인 조건이 "A=B"인 조이너 변환과 필터 조건이 "A>5"인 필터 변환이 포함된다고 가정합니다. 데이터 통합 서비스가 "B>5"를 조인 조건에 추가하고 소스에 더 가깝게 필터 변환을 이동할 수 있습니다.

글로벌 조건자 최적화 방법은 조건자 최적화 방법보다 더욱 효과적으로 조건자 식을 적용합니다. 글로벌 조건자 최적화 방법은 매핑 성능을 높이기 위해 식을 간소화하거나 다시 쓸 수 있는지 여부를 결정합니다. 또한 매핑 성능을 높이기 위해 매핑에서 가능한 한 이른 시기에 조건자 식을 적용하려고 시도합니다.

매핑에 중첩 조이너 또는 분기(각 분기의 필터 포함)가 있는 경우 글로벌 조건자 최적화 방법은 필터를 유추하고 소스에 더 가깝게 필터를 푸시합니다. 데이터 통합 서비스가 글로벌 조건자 최적화 방법을 사용하는 경우 필터를 분할하거나, 소스에 더 가깝게 필터를 이동하거나, 매핑에서 필터를 제거합니다.

## 분기 잘라내기 최적화 방법

데이터 통합 서비스는 매핑의 대상에 행을 제공하지 않는 변환에 분기 잘라내기 최적화 방법을 적용할 수 있습니다.

데이터 행에서 필터 조건이 FALSE로 평가되는 경우 데이터 통합 서비스가 필터 변환을 제거할 수 있습니다. 예를 들어 매핑에 두 관계형 소스에서 데이터를 필터링하는 두 필터 변환이 있다고 가정합니다. 한 필터 변환에는 필터 조건 Country=US가 있고 다른 필터 변환에는 필터 조건 Country=Canada가 있습니다. 합집합 변환은 두 관계형 소스를 조인하며 필터 조건 Country=US를 가집니다. 이 경우 데이터 통합 서비스가 필터 조건이 Country=Canada인 필터 변환을 매핑에서 제거할 수 있습니다.

보통 또는 전체 최적화 수준을 선택하면 개발자 도구에서 기본적으로 분기 잘라내기 최적화 방법을 활성화합니다. 최적화가 성능을 향상시키지 않는 경우 최적화 수준을 최소 또는 없음으로 설정하여 분기 잘라내기를 비활성화할 수 있습니다.

## 푸시인 최적화 방법

푸시인 최적화에서는 데이터 통합 서비스가 필터 변환 논리를 매핑에서 해당 필터 변환의 직전 업스트림 변환으로 이동합니다. 푸시인 최적화는 매핑을 통과하는 행 수를 줄여 성능을 향상시킵니다.

변환에 부작용이 있는 경우 데이터 통합 서비스가 필터 논리를 다른 변환으로 이동하지 않습니다. 데이터 통합 서비스는 SQL 변환, 웹 서비스 소비자 변환 및 Java 변환에 부작용이 있는지 여부를 결정할 수 없습니다. 하지만 사용자가 SQL 변환, 웹 서비스 소비자 변환 및 Java 변환에서 푸시인 최적화를 구성할 수 있습니다.

## 푸시다운 최적화

데이터 통합 서비스는 푸시다운 최적화를 적용할 때 소스 데이터베이스에 변환 논리를 푸시합니다. 데이터 통합 서비스는 변환 논리를 SQL 쿼리로 변환한 후 SQL 쿼리를 데이터베이스로 전송합니다. 소스 데이터베이스는 SQL 쿼리를 실행하여 변환을 처리합니다.

소스 데이터베이스가 데이터 통합 서비스보다 빠른 속도로 변환 논리를 처리할 수 있는 경우 푸시다운 최적화가 매핑 성능을 향상시킵니다. 또한 데이터 통합 서비스가 소스에서 훨씬 적은 데이터를 읽습니다.

데이터 통합 서비스가 소스 데이터베이스에 푸시하는 변환 논리의 양은 데이터베이스, 변환 논리 및 매핑 구성에 따라 달라집니다. 데이터베이스에 푸시할 수 없는 모든 변환 논리는 데이터 통합 서비스가 처리합니다.

푸시다운 최적화를 적용하면 데이터 통합 서비스가 소스에서 대상으로의 최적화된 매핑을 분석하거나 소스 데이터베이스에 푸시할 수 없는 다운스트림 변환에 도달할 때까지 분석합니다. 데이터 통합 서비스는 변환 논리를 푸시다운한 각 소스에 대해 **SELECT** 쿼리를 생성하고 실행합니다. 또한 대상이 데이터베이스에 푸시된 경우 데이터 통합 서비스가 **INSERT** 쿼리를 생성할 수 있습니다. 데이터 통합 서비스는 SQL 쿼리의 결과를 읽고 매핑의 나머지 변환을 처리합니다.

매핑 런타임 속성에서 푸시다운 유형을 선택하는 경우 데이터 통합 서비스가 매핑에 푸시다운 최적화를 적용합니다.

다음과 같은 푸시다운 유형을 선택할 수 있습니다.

- 없음. 매핑에 대한 푸시다운 유형을 선택하지 않습니다.
- 소스. 데이터 통합 서비스가 소스 데이터베이스에 대해 최대한 많은 변환 논리를 푸시다운하려고 시도합니다.
- 전체. 데이터 통합 서비스가 전체 변환 논리를 소스 데이터베이스에 푸시합니다.

또한 푸시다운 유형에 대한 문자열 매개 변수를 작성하고 다음 매개 변수 값을 사용할 수 있습니다.

- 없음
- 소스
- 전체

## 전체 푸시다운 최적화

데이터 통합 서비스가 전체 푸시다운 최적화를 적용하는 경우 매핑의 모든 변환 논리를 소스 데이터베이스로 푸시합니다. 매핑 런타임 속성에서 전체 푸시다운을 구성할 수 있습니다.

전체 푸시다운 최적화는 소스와 대상이 동일한 데이터베이스에 있거나 집계 및 필터 변환과 같은 변환이 소스 데이터베이스에서 처리되고 이동되는 데이터 양을 줄여주는 경우에 적합합니다. 예를 들어 매핑에 **Teradata** 소스 및 **Teradata** 대상이 포함되어 있는 경우 **Teradata** 소스 데이터베이스에서 **Teradata** 대상 데이터베이스로 처리하기 위한 모든 변환 논리를 푸시할 전체 푸시다운 최적화를 구성합니다.

전체 푸시다운에 대한 업데이트 전략 변환이 포함된 매핑을 구성하는 경우 해당 매핑에 대한 푸시다운 호환성을 결정해야 합니다.

데이터 통합 서비스는 다음과 같은 시나리오에서 업데이트 전략 변환이 포함된 매핑을 푸시다운할 수 있습니다.

- 업데이트 전략 변환에 연결된 대상 변환이 같은 키가 없는 여러 행을 받는 경우.
- 업데이트 전략 변환에 연결된 대상 변환이 순서를 다시 지정할 수 있는 같은 키가 있는 여러 행을 받는 경우.

데이터 통합 서비스는 다음과 같은 시나리오에서 업데이트 전략 변환이 포함된 매핑을 푸시다운할 수 없습니다.

- 업데이트 전략 변환에 연결된 대상 변환이 순서를 다시 지정할 수 없는 같은 키가 있는 여러 행을 받는 경우.

매핑의 푸시다운 호환성 매개 변수를 사용할 수도 있습니다. 다음 매개 변수 값을 사용할 수 있습니다.

- noMultipleRowsWithSameKeyOnTarget
- reorderAllowedForMultipleRowsWithSameKey
- reorderNotAllowedForRowsWithSameKey

데이터 통합 서비스는 다음 소스에 대해 전체 푸시다운 최적화를 사용할 수 있습니다.

- Amazon Redshift
- Greenplum
- IBM DB2
- Microsoft SQL Server
- Netezza
- Oracle
- SAP HANA
- Snowflake
- Teradata

## 소스 푸시다운

데이터 통합 서비스가 소스 푸시다운을 적용할 때 데이터 통합 서비스는 소스에서 대상으로의 매핑을 분석하거나 소스 데이터베이스에 푸시할 수 없는 다운스트림 변환에 도달할 때까지 분석합니다.

데이터 통합 서비스는 데이터베이스에 푸시할 수 있는 각 변환의 변환 논리에 따라 **SELECT** 문을 생성하고 실행합니다. 그런 다음 이 **SQL** 쿼리의 결과를 읽고 나머지 변환을 처리합니다.

소스와 대상이 서로 다른 데이터베이스에 있는 경우 소스 푸시다운을 사용하도록 매핑을 구성할 수 있습니다. 예를 들어 매핑에 **Teradata** 소스와 **Oracle** 대상이 포함되어 있는 경우 **Teradata** 소스에 대한 처리를 위해 일부 변환 논리를 푸시하도록 소스 푸시다운을 구성할 수 있습니다.

## 푸시다운 최적화 규칙 및 지침

데이터 통합 서비스가 소스 데이터베이스에 변환 논리를 푸시할 수 있습니다.

다음과 같은 규칙 및 지침이 푸시다운 최적화에 적용됩니다.

- 소스가 같은 데이터베이스 관리 시스템에 속하고 동일한 연결을 사용하는 경우 데이터 통합 서비스가 소스 데이터베이스에 조회 및 조이너 변환 논리를 푸시할 수 있습니다.
- 이진 데이터 유형이 있는 소스에는 데이터 통합 서비스가 변환 논리를 푸시할 수 없습니다.
- 데이터 통합 서비스는 **IBM DB2** 데이터 소스가 있으며 열 전체 자릿수가 10진수 데이터 유형에 대해 28~31자리 사이인 경우 푸시다운 최적화를 비활성화합니다.
- 데이터 통합 서비스는 기본적으로 **SQL** 데이터 서비스 또는 웹 서비스에 대한 푸시다운 최적화를 활성화합니다. **SQL** 데이터 서비스 또는 웹 서비스에 대한 푸시다운 최적화를 비활성화할 수 없습니다.
- 데이터 통합 서비스는 그룹화되지 않은 포트에서 집계 및 비집계 함수를 사용하는 식이 포함된 집계 변환을 푸시할 수 없습니다.



## 단일 패스 읽기

단일 패스 읽기를 사용하면 여러 대상을 사용자 지정된 데이터 개체 하나로 채울 수 있습니다. 동일한 소스를 사용하는 매핑 여러 개가 있는 경우 단일 패스 읽기를 사용하는 것이 좋습니다.

단일 패스 읽기 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**각 매핑의 변환 논리를 단일 매핑으로 결합하고 각 소스에 대해 사용자 지정된 단일 데이터 개체를 사용합니다.**

데이터 통합 서비스는 각 소스를 한 번만 읽고 데이터를 개별 파이프라인으로 보냅니다. 특정 행은 이러한 파이프라인 전체나 임의의 조합에서 사용되거나, 어떠한 파이프라인에서도 사용되지 않을 수 있습니다.

예를 들어 **Purchasing** 소스 테이블이 있으며 매일 이 소스를 사용하여 집계와 순위 지정을 수행한다고 가정합니다. 집계 변환과 순위 변환을 각각 별도의 매핑에 배치하면 데이터 통합 서비스가 동일한 소스 테이블을 두 번 읽어야 합니다. 하지만 집계와 매핑 논리를 단일 소스 한정자를 사용하여 한 매핑에 포함시키면 데이터 통합 서비스가 **Purchasing** 소스 테이블을 한 번만 읽고 해당 데이터를 별도의 파이프라인으로 보냅니다.

### 매핑에서 공통 함수 추출

단일 패스 읽기를 활용하기 위해 매핑을 변경할 때 매핑에서 공통 함수를 추출하여 기능을 최적화할 수 있습니다. 예를 들어 집계 변환과 순위 변환 모두의 **Price** 포트에서 백분율을 빼야 하는 경우 파이프라인을 분할하기 전에 백분율을 빼면 작업을 최소화할 수 있습니다. 식 변환을 사용하여 백분율을 뺄 수 있으며, 변환 후 매핑을 분할합니다.

## 필터 최적화

사용자 지정 데이터 개체 내에서 필터링하고 매핑의 초기 단계에 필터를 배치하여 매핑을 최적화할 수 있습니다.

필터 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**사용자 지정 데이터 개체에서 필터를 사용하여 소스의 행을 제거합니다.**

매핑에서 행을 필터링하면 데이터 흐름의 초기에 필터링을 통해 효율성을 향상시킬 수 있습니다. 사용자 지정 데이터 개체에서 필터를 사용하여 소스의 행을 제거합니다. 사용자 지정 데이터 개체는 관계형 소스에서 추출되는 행 집합을 제한합니다.

사용자 지정 데이터 개체에서 필터를 사용할 수 없다면 필터 변환을 사용하고 최대한 사용자 지정 데이터 개체에 가까운 위치로 해당 변환을 이동하여 데이터 흐름의 초기에 불필요한 데이터를 제거합니다. 필터 변환은 대상으로 전송되는 행 집합을 제한합니다.

**거부된 행을 유지할 필요가 없다면 업데이트 전략 변환에서 필터를 사용합니다.**

거부된 행을 유지할 필요가 없다면, 매핑 성능을 향상시키기 위해 필터 변환을 사용하여 업데이트 전략 변환에서 거부된 행을 삭제할 수도 있습니다.

**필터 조건에서 복잡한 식을 사용하지 마십시오.**

필터 조건에서 복잡한 식을 사용하지 마십시오. 필터 변환을 최적화하려면 필터 조건에서 단순한 정수 또는 **true/false** 식을 사용합니다.

필터 변환은 매핑 내에서 데이터를 필터링합니다. 필터 변환은 모든 유형의 소스에서 행을 필터링합니다. 사용자 지정 데이터 개체는 관계형 소스에서 행을 필터링합니다. 필터 변환은 모든 유형의 소스에서 행을 필터링합니다.

## 데이터 유형 변환 최적화

불필요한 데이터 유형 변환을 제거하여 성능을 향상시킬 수 있습니다. 예를 들어 매핑이 데이터를 정수 열에서 10진수 열로 이동한 다음 다시 정수 열로 이동하는 경우 불필요한 데이터 유형 변환 때문에 성능이 저하됩니다. 가능한 경우 매핑에서 불필요한 데이터 유형 변환을 제거하십시오.

데이터 유형 변환 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**조희 및 필터 변환을 사용하여 비교를 수행할 때 다른 데이터 유형 대신 정수 값을 사용합니다.**

예를 들어 많은 데이터베이스가 미국 우편 번호 정보를 Char 또는 Varchar 데이터 유형으로 저장합니다. 이 우편 번호 데이터를 정수 데이터 유형으로 변환하면 조희 데이터베이스가 우편 번호 94303-1234를 943031234로 저장합니다. 이렇게 하면 우편 번호에 기반한 조희 비교의 속도가 향상됩니다.

**포트 대 포트 변환을 통해 소스 날짜를 문자열로 변환하면 매핑 성능이 향상됩니다.**

대상의 포트를 그대로 문자열로 두거나 포트를 날짜/시간 포트로 변경할 수 있습니다.

## 오류 추적

성능을 향상시키려면 데이터 통합 서비스가 매핑을 실행하면서 생성하는 로그 이벤트의 수를 줄입니다. 매핑 구성 또는 매핑 배포 속성을 통해 매핑 최적화 수준을 업데이트하면 매핑 성능이 향상됩니다. 비용 기반 최적화 방법을 사용하여 매핑을 최적화합니다.

오류 추적 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**매핑 속성에서 추적 수준을 간단으로 설정합니다.**

매핑에 많은 수의 변환 오류가 포함되어 있고 이러한 오류를 해결할 필요가 없다면 매핑 속성에서 추적 수준을 간단으로 설정합니다. 이 추적 수준에서는 데이터 통합 서비스가 데이터 거부에 대한 오류 메시지나 행 수준 정보를 쓰지 않습니다.

매핑을 디버깅해야 하는 경우 추적 수준을 자세한 정보 표시로 설정하면 매핑을 실행할 때 상당한 성능 저하가 발생할 수 있습니다. 성능을 조정할 때에는 자세한 정보 표시 추적을 사용하지 마십시오. 이 매핑 추적 수준은 매핑 내의 모든 변환 관련 추적 수준을 재정의합니다. 높은 변환 오류 수준에 대한 장기간의 응답에는 사용하지 않는 것이 좋습니다.

**매핑의 최적화 수준을 변경합니다.**

매핑 실행에 과도한 시간이 걸리는 경우 매핑의 최적화 수준을 변경할 수 있습니다. 최적화 수준은 데이터 통합 서비스가 런타임에 매핑에 적용하는 최적화 방법을 결정합니다.

매핑 구성이나 매핑 배포 속성에서 매핑에 대한 최적화 수준을 설정합니다. 데이터 통합 서비스는 사용자가 매핑을 실행하는 방식에 따라 매핑에 서로 다른 최적화 수준을 적용합니다.

**비용 기반 최적화 방법을 사용합니다.**

비용 기반 최적화 방법을 사용하면 데이터 통합 서비스가 매핑을 평가하고, 의미적으로 동등한 매핑을 생성한 다음, 최상의 성능으로 매핑을 실행합니다. 이 방법은 다수의 조이너 변환이 포함된 매핑에서 가장 효과적이며, 인접한 비정렬 내부 조인 작업을 수행하는 매핑의 런타임을 줄여줍니다.

의미적으로 동등한 매핑은 동일한 기능을 수행하고 동일한 결과를 생성하는 매핑입니다. 의미적으로 동등한 매핑을 생성하기 위해 데이터 통합 서비스는 원래 매핑을 조각으로 분할한 다음, 최적화할 수 있는 매핑 조각을 결정합니다.



## 제 6 장

# 분할된 매핑 최적화

이 장에 포함된 항목:

- [분할된 매핑 최적화 개요, 49](#)
- [여러 개의 CPU 사용, 49](#)
- [최대 병렬도 값 증가, 50](#)
- [분할을 위한 플랫폼 파일 최적화, 50](#)
- [분할을 위해 관계형 데이터베이스 최적화, 51](#)
- [분할을 위한 변환 최적화, 53](#)

## 분할된 매핑 최적화 개요

분할 옵션을 사용할 수 있는 경우, 매핑을 실행하는 데이터 통합 서비스가 병렬도를 최대화하도록 할 수 있습니다. 병렬도를 최대화할 때 데이터 통합 서비스는 기본 데이터를 동적으로 파티션으로 나누고 모든 파티션을 동시에 처리합니다.

매핑이 큰 데이터 집합을 처리하거나 복잡한 계산을 수행하는 변환을 포함하는 경우 매핑이 처리하는 데 오래 걸릴 수 있고 데이터 처리량이 낮아질 수 있습니다. 이러한 매핑을 위해 분할을 활성화하면 데이터 통합 서비스가 추가적인 스레드를 사용하여 매핑을 처리합니다.

다음 태스크를 수행하여 분할된 매핑 성능을 최적화할 수 있습니다.

- 매핑이 실행되는 노드에서 여러 CPU를 사용합니다.
- 데이터 통합 서비스에 대한 최대 병렬도 값을 늘립니다.
- 플랫폼 파일 데이터 개체의 속성을 구성합니다.
- 분할을 최적화하도록 관계형 데이터베이스를 구성합니다.
- 변환의 속성을 구성합니다.

## 여러 개의 CPU 사용

처리 스레드 수를 늘리면 매핑이 실행되는 노드에서 로드가 늘어납니다. 노드의 CPU 대역폭이 충분하면 매핑에서 여러 데이터 행을 동시에 처리하여 매핑 성능을 최적화할 수 있습니다.

데이터 통합 서비스는 여러 CPU를 사용하여 다중 파티션을 포함하는 매핑을 처리할 수 있습니다. 서비스가 사용하는 CPU 수는 파티션 지점 수, 각 파이프라인 단계에 대해 작성된 스레드 수 및 매핑을 처리하는 데 필요한 리

소스 양 등의 요인에 따라 다릅니다. 단순 매핑은 두 파티션에서 더 빠르게 실행되지만, 일반적으로 단일 파티션에서 매핑이 실행될 경우에 비해 2배의 CPU가 필요합니다.

## 최대 병렬도 값 증가

최대 병렬도는 단일 파이프라인 단계를 처리할 수 있는 병렬 스레드의 최대 수를 결정합니다. 사용 가능한 하드웨어 리소스를 기반으로 데이터 통합 서비스에 대한 **최대 병렬도** 속성을 구성합니다. 최대 병렬도 값을 증가시키면 처리 시간의 양을 줄일 수 있습니다.

최대 병렬도 값을 증가시키는 경우 다음 지침을 고려하십시오.

### 사용 가능한 CPU 수에 따라 값을 증가시킵니다.

매핑이 실행되는 노드에서 사용 가능한 CPU 수에 따라 최대 병렬도 값을 높이십시오. 최대 병렬도 값을 늘리면 데이터 통합 서비스가 더 많은 스레드를 사용하여 매핑을 실행하고, 더 많은 CPU를 활용합니다. 단순 매핑은 두 파티션에서 더 빠르게 실행되지만, 일반적으로 단일 파티션에서 매핑이 실행될 경우에 비해 2배의 CPU가 필요합니다.

### 처리 스레드의 총 수를 고려하십시오.

최대 병렬도 값을 설정할 때 처리 스레드의 총 수를 고려하십시오. 복합 매핑으로 인해 추가적인 파티션 지점이 여러 개 사용되면 데이터 통합 서비스가 CPU에서 처리할 수 있는 것보다 많은 처리 스레드를 작성할 수 있습니다.

처리 스레드의 총 수는 최대 병렬도 값과 같습니다.

### 데이터 통합 서비스가 실행해야 하는 다른 작업을 고려하십시오.

각 매핑에서 많은 수의 스레드를 사용하도록 최대 병렬도를 구성하면 데이터 통합 서비스에서 추가 작업을 수행할 때 사용할 수 있는 스레드 수가 적어집니다.

### 필요에 따라 값을 매핑에 맞춰 변경하십시오.

기본적으로 각 매핑의 최대 병렬도는 자동으로 설정됩니다. 각 매핑은 데이터 통합 서비스에 대해 정의된 최대 병렬도 값을 사용합니다.

개발자는 **Developer tool**에서 매핑 런타임 속성의 최대 병렬도 값을 변경하여 특정 매핑에 대한 최대값을 정의할 수 있습니다. 데이터 통합 서비스 및 매핑에 대해 최대 병렬도를 다른 정수 값으로 설정할 경우 데이터 통합 서비스는 두 개 중에서 최소값을 사용합니다.

## 분할을 위한 플랫폼 파일 최적화

분할에 대해 활성화된 매핑이 플랫폼 파일 소스에서 읽거나 플랫폼 파일 대상으로 쓰는 경우 데이터 통합 서비스가 여러 스레드를 사용하여 플랫폼 파일을 대상으로 읽거나 쓸 수 있습니다.

### 분할을 위한 플랫폼 파일 소스 최적화

여러 스레드를 사용하여 플랫폼 파일에서 읽는 경우 성능을 최적화하려면 행 순서를 유지하는 대신 처리량을 최적화하도록 플랫폼 파일 데이터 개체를 구성합니다.

분할된 플랫폼 파일 소스에 대한 병목 현상을 줄이려면 다음과 같은 해결 방법을 고려하십시오.

**플랫 파일 데이터 개체에 대한 동시 읽기 분할을 구성하여 처리량을 최적화합니다.**

플랫 파일 데이터 개체 고급 속성에서 **동시 읽기 분할** 속성을 설정하여 처리량을 최적화합니다. 처리량을 최적화하는 경우 데이터 통합 서비스는 파일 또는 파일 목록에서 순차적으로 행을 읽지 않기 때문에 행 순서를 유지하지 않습니다.

## 분할을 위한 플랫 파일 대상 최적화

여러 스레드를 사용하여 플랫 파일에 쓰는 경우 성능을 최적화하려면 대상 출력을 별도의 파일에 쓰도록 파티션을 구성하고 여러 대상 디렉터리를 구성합니다.

분할된 플랫 파일 대상에 대한 병목 현상을 줄이려면 다음과 같은 해결 방법을 고려하십시오.

**대상 출력을 별도의 파일에 쓰도록 파티션을 구성합니다.**

플랫 파일 데이터 개체 고급 속성에서 **병합 유형** 속성을 **병합 안 함**으로 설정합니다. 데이터 통합 서비스가 각 파티션에 대해 별도의 파일에 대상 출력을 동시에 씁니다. 병합된 대상 데이터가 필요한 경우 동시 병합 유형은 순차 병합 유형보다 성능을 더욱 최적화합니다.

**여러 대상 디렉터리를 구성합니다.**

여러 스레드가 단일 디렉터리에 쓰는 경우 매핑에서 I/O(입력/출력) 경합으로 인해 병목 현상이 발생할 수 있습니다. 스레드가 동시에 파일 시스템에 데이터를 쓰는 경우 I/O 경합이 발생할 수 있습니다. 여러 디렉터리를 구성하는 경우 데이터 통합 서비스는 라운드 로빈 방식으로 각 스레드에 대해 출력 디렉터리를 확인합니다.

플랫 파일 데이터 개체에 대한 고급 속성에서 출력 파일 디렉터리를 구성합니다. 관리자가 **Administrator** 도구에서 데이터 통합 서비스의 **대상 디렉터리** 속성에 대해 세미콜론으로 구분하여 여러 디렉터리를 입력한 경우 기본 **TargetDir** 시스템 매개 변수 값을 사용합니다. 또는 다른 값을 입력하여 플랫 파일 데이터 개체와 관련된 여러 출력 파일 디렉터리를 구성할 수 있습니다.

## 분할을 위해 관계형 데이터베이스 최적화

분할에 대해 활성화된 매핑이 **IBM DB2 for LUW** 또는 **Oracle** 관계형 데이터베이스를 대상으로 읽거나 쓰는 경우 데이터 통합 서비스가 여러 스레드를 사용하여 관계형 소스를 읽거나 관계형 대상으로 쓸 수 있습니다.

여러 스레드를 사용하여 **DB2 for LUW** 또는 **Oracle** 관계형 데이터베이스 간에 읽거나 쓰는 경우 성능을 최적화하기 위해 소스 및 대상 테이블을 분할할 수 있습니다.

**참고:** 매핑이 **DB2 for LUW** 또는 **Oracle**이 아닌 관계형 데이터베이스를 대상으로 읽거나 쓰는 경우 데이터 통합 서비스가 하나의 판독기 스레드 또는 하나의 기록기 스레드를 사용합니다.

## 분할을 위해 소스 데이터베이스 최적화

여러 스레드를 사용하여 **DB2 for LUW** 또는 **Oracle** 소스 데이터베이스에서 읽는 경우의 성능을 최적화하려면 소스 테이블이 분할되어 있고 병렬 쿼리를 수락하도록 구성되어 있는지 확인합니다.

분할을 위해 소스 데이터베이스를 최적화하려면 다음 작업을 수행합니다.

**데이터베이스 파티션을 소스에 추가합니다.**

데이터베이스 파티션을 관계형 소스에 추가하여 해당 소스를 읽는 데이터 통합 서비스 쿼리의 속도를 향상시킵니다. 소스에 데이터베이스 파티션이 없을 경우 데이터 통합 서비스는 하나의 스레드를 사용하여 소스에서 읽습니다.

#### **병렬 쿼리를 활성화합니다.**

관계형 데이터베이스에는 해당 데이터베이스에 대한 병렬 쿼리를 활성화하는 옵션이 있을 수 있습니다. 이러한 옵션은 데이터베이스 설명서를 참조하십시오. 이러한 옵션이 활성화되어 있지 않으면 데이터 통합 서비스가 여러 개의 파티션 SELECT 문을 순차적으로 실행합니다.

#### **데이터를 여러 개의 테이블스페이스로 분리합니다.**

각 데이터베이스는 데이터를 여러 테이블스페이스로 분리하는 옵션을 제공합니다. 각 테이블스페이스는 고유한 파일 시스템을 참조할 수 있으므로, 파티션 간에 I/O 경합이 방지됩니다.

#### **데이터베이스에 대해 허용되는 세션의 최대 수를 증가시킵니다.**

데이터 통합 서비스는 각 파티션마다 소스 데이터베이스에 대한 연결을 별도로 작성합니다. 데이터베이스가 더 많은 수의 동시 연결을 처리할 수 있도록 최대 허용 세션 수를 증가시킵니다.

## **분할을 위해 대상 데이터베이스 최적화**

여러 스레드를 사용하여 DB2 for LUW 또는 Oracle 대상 데이터베이스에 기록하는 경우의 성능을 최적화하려면 대상 테이블이 분할되고 행을 병렬로 삽입하도록 구성되어 있는지 확인합니다.

분할을 위해 대상 데이터베이스를 최적화하려면 다음 작업을 수행합니다.

#### **DB2 for LUW 대상에 데이터베이스 파티션을 추가합니다.**

데이터 통합 서비스는 여러 스레드를 사용하여 데이터베이스 파티션이 없는 DB2 for LUW 대상에 기록할 수 있습니다. 하지만 대상에 데이터베이스 파티션이 있을 경우 로드 성능을 최적화할 수 있습니다. 이 경우 각 기록기 스레드가 데이터베이스 파티션이 있는 DB2 for LUW 노드에 연결됩니다. 모든 스레드가 단일 마스터 노드에 연결되는 대신 기록기 스레드가 서로 다른 DB2 for LUW 노드에 연결되기 때문에 성능이 향상됩니다.

#### **병렬 삽입을 활성화합니다.**

관계형 데이터베이스에는 해당 데이터베이스에 대한 병렬 삽입을 활성화하는 옵션이 있을 수 있습니다. 이러한 옵션은 데이터베이스 설명서를 참조하십시오. 예를 들어 Oracle 데이터베이스에서는 `db_writer_processes` 옵션을 설정하고 DB2 for LUW 데이터베이스에서는 `max_agents` 옵션을 설정하여 병렬 삽입을 활성화합니다.

#### **데이터를 여러 개의 테이블스페이스로 분리합니다.**

각 데이터베이스는 데이터를 여러 테이블스페이스로 분리하는 옵션을 제공합니다. 각 테이블스페이스는 고유한 파일 시스템을 참조할 수 있으므로, 파티션 간에 I/O 경합이 방지됩니다.

#### **데이터베이스에 대해 허용되는 세션의 최대 수를 증가시킵니다.**

데이터 통합 서비스는 각 파티션마다 대상 데이터베이스에 대한 연결을 별도로 작성합니다. 데이터베이스가 더 많은 수의 동시 연결을 처리할 수 있도록 최대 허용 세션 수를 증가시킵니다.

#### **데이터베이스 확장성을 개선하기 위해 옵션을 설정합니다.**

관계형 데이터베이스에는 확장성을 개선하는 옵션이 있을 수 있습니다. 예를 들어 Oracle 데이터베이스에서는 보관 기록 및 기간별 통계를 비활성화하면 확장성이 개선됩니다.

# 분할을 위한 변환 최적화

데이터 통합 서비스가 여러 스레드를 사용하여 집계, 조이너, 순위 또는 분류기 변환을 실행하는 경우 서비스는 캐시 분할을 사용하여 스레드에서 캐시 크기를 나눕니다. 캐시 분할을 위해 성능을 최적화하려면 여러 캐시 디렉터리를 구성합니다.

**참고:** 조회 변환은 단일 캐시 디렉터리만 사용할 수 있습니다.

분할된 집계, 조이너, 순위 및 분류기 변환에 대한 병목 현상을 줄이려면 다음과 같은 해결 방법을 고려하십시오.

## 여러 캐시 디렉터리를 구성합니다.

캐시 분할은 집계, 조이너, 순위 또는 분류기 변환을 처리하는 각 파티션에 대해 별개의 캐시를 작성합니다. 캐시 분할 중 각 파티션은 별도의 캐시에 서로 다른 데이터를 저장합니다. 각 캐시에는 해당 파티션에 필요한 행이 포함됩니다. 캐시 분할은 각 스레드에서 별개의 캐시를 병렬로 쿼리하기 때문에 매핑 성능을 최적화합니다.

캐시 크기가 변환을 실행하는 데 필요한 메모리 양보다 작은 경우 변환 스레드는 캐시 디렉터리에 기록하여 캐시 파일에 오버플로우 값을 저장합니다. 여러 스레드를 단일 디렉터리에 쓰는 경우 매핑에서 I/O 경합으로 인해 병목 현상이 발생할 수 있습니다. 스레드가 동시에 파일 시스템에 데이터를 쓰는 경우 I/O 경합이 발생할 수 있습니다. 여러 캐시 디렉터리를 구성하면 데이터 통합 서비스가 라운드 로빈 방식으로 각 변환 스레드에 대한 캐시 디렉터리를 결정합니다.

집계, 조이너 또는 순위 변환에서 **캐시 디렉터리** 고급 속성의 캐시 디렉터리를 구성합니다. 관리자가 **Administrator** 도구에서 데이터 통합 서비스의 **캐시 디렉터리** 속성에 대해 세미콜론으로 구분하여 여러 디렉터리를 입력한 경우 기본 **CacheDir** 시스템 매개 변수 값을 사용합니다. 또는 다른 값을 입력하여 변환과 관련된 여러 캐시 디렉터리를 구성할 수 있습니다.

분류기 변환에서 **작업 디렉터리** 고급 속성의 캐시 디렉터리를 구성합니다. 관리자가 **Administrator** 도구에서 데이터 통합 서비스의 **임시 디렉터리** 속성에 대해 세미콜론으로 구분하여 여러 디렉터리를 입력한 경우 기본 **TempDir** 시스템 매개 변수 값을 사용합니다. 또는 다른 값을 입력하여 변환과 관련된 여러 캐시 디렉터리를 구성할 수 있습니다.

## 제 7 장

# 런타임 최적화

이 장에 포함된 항목:

- [런타임 최적화 개요, 54](#)
- [응용 프로그램 서비스 최적화, 54](#)
- [모니터링 통계, 56](#)
- [메모리 할당, 57](#)
- [데이터 개체 캐싱, 58](#)
- [시스템 최적화, 61](#)

## 런타임 최적화 개요

성능 기능을 활성화하고 데이터 통합 서비스 속성을 조정하여 매핑 성능을 최적화할 수 있습니다.

**Administrator** 도구에서 다음과 같은 최적화 기법을 사용하여 요구 사항에 기반한 최상의 성능 결과를 얻을 수 있습니다.

- 응용 프로그램 서비스 프로세스를 최적화합니다.
- 모니터링 통계를 구성하여 시스템 병목 현상을 모니터링합니다.
- 최적의 시스템 성능을 얻을 수 있도록 메모리를 할당합니다.
- 데이터 개체 캐싱을 구성합니다.
- 시스템을 최적화하여 시스템 지연 시간과 디스크 액세스 속도 저하를 방지합니다.

## 응용 프로그램 서비스 최적화

성능에 영향을 미치는 응용 프로그램 서비스 프로세스를 최적화합니다. 분석 서비스, 데이터 통합 서비스 및 모델 리포지토리 서비스를 최적화할 수 있습니다.

### 분석 서비스 최적화

분석 서비스를 조정하여 성능을 최적화합니다. 분석 서비스 프로세스의 메모리 속성을 구성하고, 네트워크 대기 시간을 최소화하고, 분석 도구 플랫폼 파일 업로드 설정을 구성하여 서비스 성능을 향상시킬 수 있습니다.

분석 서비스 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**10MB보다 큰 플랫 파일을 업로드할 수 있는 네트워크 경로 위치에 연결하도록 분석 도구를 구성합니다.**

분석가가 10MB보다 큰 플랫 파일을 분석 도구가 실행되고 있는 시스템의 Informatica 설치 디렉터리에 업로드하는 경우 분석 서비스 프로세스 성능이 저하될 수 있습니다. 이 경우 디스크 공간과 네트워크 성능 모두 영향을 받을 수 있습니다.

**분석 도구에서 Informatica 설치 디렉터리로 10MB보다 작은 플랫 파일을 업로드하십시오.**

분석가가 10MB보다 큰 플랫 파일을 분석 도구에서 Informatica 설치 디렉터리로 업로드하는 경우 분석 서비스 프로세스 성능이 저하될 수 있습니다. 이 경우 디스크 공간과 네트워크 성능 모두 영향을 받을 수 있습니다.

**분석 서비스 프로세스의 최대 힙 크기 속성을 늘립니다.**

분석 서비스 프로세스가 동시에 로그인한 많은 수의 사용자를 처리하는 동안 대량의 메모리를 사용할 수 있습니다. 이 때문에 데이터 통합 서비스, 모델 리포지토리 서비스 등과 같은 서비스와 분석 서비스 사이에서 많은 수의 네트워크 연결이 열릴 수 있습니다.

Administrator 도구를 사용하여 고급 속성에서 분석 서비스 프로세스에 대한 최대 힙 크기 속성을 더 큰 값으로 구성합니다.

**큰 매핑 사양을 테이블로 내보내거나 플랫 파일로 내보낸 다음 파일을 잘라냅니다.**

분석가가 분석 도구에서 큰 매핑 사양으로 플랫 파일로 내보내는 경우 분석 서비스 프로세스가 성능을 저하시킬 수 있습니다.

## 데이터 통합 서비스 최적화

데이터 통합 서비스 프로세스를 조정하여 서비스 성능을 향상시킵니다. 데이터 통합 서비스 프로세스의 메모리 속성을 구성할 수 있습니다. 데이터 통합 서비스에서 실행되고 있는 각 웹 서비스와 SQL 데이터 서비스가 동시 요청을 처리하도록 구성할 수 있습니다.

데이터 통합 서비스 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**데이터 통합 서비스 프로세스의 최대 힙 크기 속성을 구성합니다.**

데이터 통합 서비스가 SQL 데이터 서비스 및 웹 서비스를 처리하는 동안 대량의 메모리를 사용할 수 있습니다.

Administrator 도구를 사용하여 고급 속성에서 데이터 통합 서비스 프로세스에 대한 최대 힙 크기 속성을 더 큰 값으로 구성합니다.

**웹 서비스에서 데이터 통합 서비스에 대한 DTM 활성 상태 유지 시간 속성을 구성합니다.**

데이터 통합 서비스는 시스템 리소스를 사용하여 각 웹 서비스 요청의 DTM 인스턴스를 생성합니다. 데이터 통합 서비스가 DTM 인스턴스 하나를 사용하여 둘 이상의 웹 서비스 요청을 처리하도록 구성합니다.

Administrator 도구를 사용하여 웹 서비스에서 데이터 통합 서비스에 대한 DTM 활성 상태 유지 시간 속성을 구성합니다.

**데이터 통합 프로세스 속성과 웹 서비스 및 SQL 데이터 서비스 속성에서 동시 요청에 대한 실행 옵션을 구성합니다.**

데이터 통합 서비스, 각 SQL 데이터 서비스 및 데이터 통합 서비스에서 실행되고 있는 각 웹 서비스는 각 동시 요청에 대해 시스템 리소스와 메모리 리소스를 사용합니다.

데이터 통합 서비스, 각 SQL 데이터 서비스 및 각 웹 서비스가 허용할 수 있는 동시 요청의 수를 구성하려면 데이터 통합 서비스 속성과 웹 서비스 속성을 구성합니다.

Administrator 도구를 사용하여 데이터 통합 서비스, 웹 서비스 및 SQL 데이터 서비스에 대해 다음과 같은 옵션과 속성을 구성합니다.

- 데이터 통합 서비스의 실행 옵션을 구성합니다.

- 데이터 통합 서비스 프로세스에 대한 SQL 속성의 각 SQL 데이터 서비스에 대해 최대 동시 연결 수 속성을 구성합니다.
- HTTP 구성 속성에서 각 웹 서비스에 대해 데이터 통합 서비스 프로세스에 대한 최대 백로그 요청 수 및 최대 동시 요청 수 속성을 구성합니다.

#### 웹 서비스 추적 수준을 해제합니다.

데이터 통합 서비스가 쓰고 유지 관리하는 웹 서비스 로그 파일의 수가 성능을 저하시킬 수 있습니다.

**Administrator** 도구를 사용하여 데이터 통합 서비스가 디스크에 저장하는 웹 서비스 런타임 로그 파일의 양이 줄어들도록 웹 서비스 추적 수준을 구성합니다.

## 모델 리포지토리 서비스 최적화

모델 리포지토리 서비스를 조정하여 성능을 향상시킵니다. 모델 리포지토리 서비스 프로세스의 메모리 속성을 구성하여 네트워크 대기 시간을 최소화할 수 있습니다.

모델 리포지토리 서비스 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

#### 모델 리포지토리 서비스와 같은 시스템에서 모델 리포지토리 데이터베이스를 호스팅합니다.

모델 리포지토리 데이터베이스가 원격 서버에서 호스팅되는 경우 모델 리포지토리 서비스 프로세스 성능이 저하될 수 있습니다. 대기 시간이 긴 네트워크에서 모델 리포지토리 서비스 간의 통신이 필요한 모델 리포지토리 서비스 작업이 있으면 모델 리포지토리 서비스 성능이 저하될 수 있습니다.

#### 모델 리포지토리 서비스 프로세스의 최대 힙 크기 속성을 늘립니다.

모델 리포지토리 서비스 프로세스가 동시에 로그인한 많은 수의 사용자를 처리하는 동안 대량의 메모리를 사용할 수 있습니다. 이 때문에 데이터 통합 서비스, 분석 서비스 등과 같은 서비스와 모델 리포지토리 서비스 사이에서 많은 수의 네트워크 연결이 열릴 수 있습니다.

**Administrator** 도구를 사용하여 고급 속성에서 모델 리포지토리 서비스 프로세스에 대한 최대 힙 크기 속성을 더 큰 값으로 구성합니다.

## 모니터링 통계

모니터링은 서비스 관리자가 수행하는 도메인 기능입니다. 서비스 관리자는 모델 리포지토리에 모니터링 구성을 저장합니다. **Administrator** 도구의 모니터 탭을 사용하여 선택한 서비스에서 실행되는 작업(실행 중, 실패, 취소됨 및 완료됨)의 총 수와 같은 시스템 병목 현상을 모니터링합니다.

모니터링 통계 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

#### 도메인을 구성하여 모니터링을 설정합니다.

모니터링을 설정하면 데이터 통합 서비스가 지속형 통계와 모니터링 보고서를 모델 리포지토리에 저장합니다. 지속형 통계는 이전에 실행된 통합 개체에 대한 기록 정보입니다. 모니터링 보고서는 통합 개체에 대한 주요 메트릭스를 보여 줍니다.

도메인에 대한 모니터링 설정을 구성하여 데이터 통합 서비스에 배포된 개체에 대한 런타임 통계를 저장하는 모델 리포지토리를 지정합니다. 모니터링 설정은 도메인의 모든 데이터 통합 서비스에 적용되며 서비스 성능에 영향을 줄 수 있습니다.



다음 테이블에는 서비스 성능에 영향을 줄 수 있는 모니터링 설정이 설명되어 있습니다.

옵션	설명
요약 기록 데이터 유지	모델 리포지토리가 평균 데이터를 저장하는 일 수입니다. 제거 기능이 비활성화된 경우 모델 리포지토리에서는 데이터를 무기한으로 저장합니다. 기본값은 180입니다. 최소값은 0입니다. 최대값은 366입니다.
세부 기록 데이터 유지	모델 리포지토리가 분 단위 데이터를 저장하는 일 수입니다. 제거 기능이 비활성화된 경우 모델 리포지토리에서는 데이터를 무기한으로 저장합니다. 기본값은 14입니다. 최소값은 1입니다. 최대값은 14입니다.
다음 기간마다 통계 제거	모델 리포지토리 서비스가 기록 데이터를 유지 옵션에 구성된 값보다 오래된 데이터를 제거하는 간격(일)입니다. 기본값은 1일입니다.
일마다	모델 리포지토리 서비스가 통계를 제거하는 시간입니다. 기본값은 오전 1시입니다.
최대 정렬 가능 레코드 수	<b>모니터</b> 탭에서 정렬할 수 있는 최대 레코드 수입니다. <b>모니터</b> 탭의 레코드 수가 이 값보다 큰 경우에는 <b>시작 시간</b> 및 <b>종료 시간</b> 을 기준으로만 정렬할 수 있습니다. 기본값은 3,000입니다.
최대 업데이트 알림 지연	데이터 통합 서비스가 통계를 모델 리포지토리에 저장하고 <b>모니터</b> 탭에 표시하기 전에 버퍼링하는 최대 시간(초)입니다. 데이터 통합 서비스가 통계를 모델 리포지토리에 저장하기 전에 예기치 않게 종료되면 통계가 손실됩니다. 기본값은 10입니다.
밀리초 표시	<b>모니터</b> 탭의 날짜 및 시간 필드용으로 밀리초를 포함합니다.

## 메모리 할당

매핑 성능을 최적화하려면 **Administrator** 도구에서 데이터 통합 서비스에 대한 메모리 속성을 구성합니다.

다음 테이블에는 매핑 서비스 모듈에 대한 요청당 최대 메모리 속성이 설명되어 있습니다.

속성	설명
요청당 최대 메모리	<p>요청당 최대 메모리 동작은 다음 데이터 통합 서비스 구성에 따라 다릅니다.</p> <ul style="list-style-type: none"> <li>서비스가 별도의 로컬 또는 원격 프로세스에서 작업을 실행하거나 서비스 프로세스 속성의 최대 메모리 크기가 0(기본값)입니다. 이 경우 요청당 최대 메모리는 데이터 통합 서비스가 단일 요청에서 자동 캐시 모드를 사용하는 모든 변환에 할당할 수 있는 최대 메모리 양(바이트)입니다. 서비스가 특정 캐시 크기를 가진 변환에 별도로 메모리를 할당합니다. 요청에서 사용한 총 메모리는 요청당 최대 메모리 값을 초과할 수 있습니다.</li> <li>서비스가 데이터 통합 서비스 프로세스에서 작업을 실행하고 서비스 프로세스 속성의 최대 메모리 크기가 0보다 큼니다. 이 경우 요청당 최대 메모리는 데이터 통합 서비스가 단일 요청에 할당할 수 있는 최대 메모리 양(바이트)입니다. 요청에서 사용한 총 메모리는 요청당 최대 메모리 값을 초과할 수 없습니다. 기본값은 536,870,912입니다.</li> </ul>

다음 테이블에는 데이터 통합 서비스에 대한 실행 옵션이 설명되어 있습니다.

속성	설명
최대 메모리 크기	<p>서비스가 데이터 통합 서비스 프로세스에서 작업을 실행할 때 데이터 통합 서비스가 모든 요청을 동시에 실행하는 데 할당할 수 있는 최대 메모리 양(단위: 바이트)입니다. 데이터 통합 서비스가 별도의 로컬 또는 원격 프로세스에서 작업을 실행하는 경우 서비스가 이 값을 무시합니다. 데이터 통합 서비스가 할당할 수 있는 메모리의 양을 제한하지 않으려면 이 속성을 0으로 설정합니다.</p> <p>값이 0보다 크면 데이터 통합 서비스가 이 속성을 사용하여 모든 요청을 동시에 실행하는 데 허용되는 최대 총 메모리를 계산합니다. 데이터 통합 서비스는 다음과 같은 방법으로 최대 총 메모리 값을 계산합니다.</p> <p>최대 메모리 크기 + 최대 힙 크기 + 프로그램 구성 요소를 로드하는 데 필요한 메모리 기본값은 0입니다.</p> <p><b>참고:</b> 프로필 또는 데이터 품질 매핑을 실행하는 경우 이 속성을 0으로 설정합니다.</p>

다음 테이블에는 데이터 통합 서비스 프로세스에 대한 최대 힙 크기 속성이 설명되어 있습니다.

속성	설명
최대 힙 크기	<p>데이터 통합 서비스를 실행하는 JVM(Java Virtual Machine)에 할당된 RAM 크기입니다. 이 속성을 사용하여 성능을 향상시킬 수 있습니다. 값에 다음 문자 중 하나를 추가하여 단위를 지정합니다.</p> <ul style="list-style-type: none"> <li>- b: 바이트.</li> <li>- k: 킬로바이트.</li> <li>- m: 메가바이트.</li> <li>- g: 기가바이트.</li> </ul> <p>기본값은 640MB입니다.</p> <p><b>참고:</b> 데이터 통합 서비스가 대량의 데이터를 처리해야 할 경우 힙 크기를 늘리는 것을 고려하십시오.</p>

## 데이터 개체 캐싱

데이터 통합 서비스는 미리 작성된 논리적 데이터 개체 및 가상 테이블에 액세스하기 위해 데이터 개체 캐싱을 사용합니다. 논리적 데이터 개체 및 가상 테이블을 포함하는 매핑, SQL 데이터 서비스 쿼리 및 웹 서비스 요청의 성능을 향상시키려면 데이터 개체 캐싱을 활성화합니다.

기본적으로 데이터 통합 서비스는 매핑, SQL 데이터 서비스 쿼리 또는 웹 서비스 요청을 실행할 때 소스 데이터를 추출하여 필요한 데이터 개체를 작성합니다. 데이터 개체 캐싱을 활성화한 경우 데이터 통합 서비스가 캐싱된 논리적 데이터 개체 및 가상 테이블을 사용할 수 있습니다.

응용 프로그램의 논리적 데이터 개체 및 가상 테이블에 대한 데이터 개체 캐싱을 구성하려면 다음 단계를 수행하십시오.

1. 데이터 통합 서비스에 대한 캐시 속성에서 데이터 개체 캐시 데이터베이스 연결을 구성합니다.
2. 응용 프로그램의 논리적 데이터 개체 또는 가상 테이블의 속성에서 캐싱을 활성화합니다.

기본적으로 데이터 통합 서비스의 데이터 개체 캐시 관리자 구성 요소가 데이터 개체 캐시 데이터베이스의 논리적 데이터 개체 및 가상 테이블에 대한 캐시 테이블을 관리합니다. 데이터 개체 캐시 관리자가 캐시를 관리하는 경우 새로 고칠 때마다 모든 데이터를 캐시 테이블에 삽입합니다. 캐시 테이블을 점차적으로 업데이트하려는 경우 데이터베이스 클라이언트 또는 기타 외부 도구를 사용하여 직접 캐시 테이블을 관리하도록 선택할 수 있습니다. 데이터 개체 캐싱을 활성화한 후 사용자 관리 캐시 테이블을 사용하도록 논리적 데이터 개체 또는 가상 테이블을 구성할 수 있습니다.

시간대가 포함된 타임스탬프 데이터 유형을 사용하고 IBM DB2 또는 Microsoft SQL Server에 대한 데이터 개체 캐싱을 활성화하려면 배포된 매핑의 날짜 시간 형식을 "YYYY-MM-DD HH24:MI:SS" 형식으로 설정합니다. 데이터 통합 서비스는 데이터를 초 단위까지 씁니다.

## 캐시 테이블에 대한 데이터 유형

데이터 통합 서비스는 캐싱된 개체를 포함하는 매핑, SQL 데이터 서비스 쿼리 및 웹 서비스 요청을 처리할 때 캐시 테이블의 데이터를 사용합니다. 데이터 통합 서비스에 필요한 캐시 테이블 데이터 유형이 캐싱된 개체 데이터 유형과 다를 수 있습니다.

데이터 개체 캐시 관리자가 데이터 통합 서비스에 필요한 데이터 유형으로 캐시 테이블을 생성합니다. 사용자 관리 캐시 테이블을 사용하는 경우 캐시 테이블이 데이터 통합 서비스에 필요한 데이터 유형을 사용하는지 확인하십시오.

## 가상 테이블 캐시 데이터 유형

다음 테이블에는 가상 테이블에 대한 캐시 테이블 데이터 유형이 나열되어 있습니다.

가상 테이블 데이터 유형	IBM DB2	Microsoft SQL Server	Oracle
Char	Vargraphic Dbclob(정밀도가 32672보다 큰 경우)	Nvarchar Ntext(정밀도가 4000보다 큰 경우)	Nvarchar2 Nclob(정밀도가 2000보다 큰 경우)
Bigint	Bigint	Bigint	숫자
부울	정수	Int	숫자
날짜	타임스탬프	Datetime2	타임스탬프
배정밀도	배정밀도	부동 소수점 수	타임스탬프
10진수	10진수	10진수	숫자
Int	정수	Int	숫자
시간	타임스탬프	Datetime2	타임스탬프
타임스탬프	타임스탬프	Datetime2	타임스탬프
Varbinary	Blob	이진 이미지(정밀도가 8000보다 큰 경우)	원시 Blob(정밀도가 2000보다 큰 경우)
Varchar	Vargraphic Dbclob(정밀도가 32672보다 큰 경우)	Nvarchar Ntext(정밀도가 4000보다 큰 경우)	Nvarchar2 Nclob(정밀도가 2000보다 큰 경우)

## 논리적 데이터 개체 캐시 데이터 유형

다음 테이블에는 논리적 데이터 개체에 대한 캐시 테이블 데이터 유형이 나열되어 있습니다.

논리적 데이터 개체 데이터 유형	DB2	Microsoft SQL Server	Oracle
Bigint	Bigint	Bigint	숫자
이진	Blob	이진 이미지(정밀도가 8000보다 큰 경우)	원시 Blob(정밀도가 2000보다 큰 경우)
날짜/시간	타임스탬프	Datetime2	타임스탬프
배정밀도	배정밀도	부동 소수점 수	숫자
10진수	10진수	10진수	숫자
정수	정수	Int	숫자
문자열	Vargraphic Dbclob(정밀도가 32672보다 큰 경우)	Nvarchar Ntext(정밀도가 4000보다 큰 경우)	Nvarchar2 Nclob(정밀도가 2000보다 큰 경우)
텍스트	Vargraphic Dbclob(정밀도가 32672보다 큰 경우)	Nvarchar Ntext(정밀도가 4000보다 큰 경우)	Nvarchar2 Nclob(정밀도가 2000보다 큰 경우)

## 데이터 개체 캐시 최적화

캐시 성능은 캐시 데이터베이스 성능과 매핑, SQL 데이터 서비스 및 웹 서비스 내 개체 구성에 따라 달라집니다.

캐시 성능을 향상시키려면 다음과 같은 해결 방법을 고려하십시오.

### 캐시 데이터베이스를 최적화합니다.

캐시의 최적 성능은 캐시 데이터베이스의 속도와 성능, 캐시 크기에 따라 달라집니다. 캐시 데이터베이스의 캐시 크기를 구성합니다.

데이터 개체 캐시 관리자가 새로 고침 작업을 위해 이전 캐시를 유지해야 하므로 캐시가 데이터 집합 두 개를 저장할 수 있을 정도로 커야 합니다. 다음 수식을 사용하여 필요한 최소 캐시 크기를 예측할 수 있습니다.

$$2 * \text{average data object size} * \text{number of data objects}$$

예를 들어 논리적 데이터 개체 20개와 가상 테이블 10개를 캐싱한다고 가정합니다. 평균 개체 크기가 15MB 이면 필요한 캐시 크기는  $2 * 15\text{MB} * (20 + 10)$ , 즉 900MB입니다.

캐시 테이블은 읽기 전용입니다. 최종 사용자가 SQL 명령으로 캐시 테이블을 업데이트할 수 없습니다.

### 논리적 데이터 개체의 기본 키와 외래 키를 정의합니다.

데이터 통합 서비스는 키가 포함된 논리적 데이터 개체의 캐시를 생성할 때 인덱스를 생성합니다. 인덱스는 캐시 데이터베이스에 대한 쿼리 성능을 향상시킬 수 있습니다.

### 매핑에 조인한 논리적 데이터 개체를 캐싱합니다.

캐싱된 논리적 데이터 개체를 조인할 때는 소스 데이터가 다른 데이터베이스에서 제공되더라도 데이터 통합 서비스가 조이너 변환 논리를 캐시 데이터베이스에 푸시다운할 수 있습니다.

**논리적 데이터 개체 또는 가상 테이블의 열에 기반하는 인덱스 캐시를 생성합니다.**

데이터 통합 서비스가 논리적 데이터 개체 또는 가상 테이블의 열에 기반한 인덱스 캐시를 생성하도록 구성합니다. 인덱스는 캐시 데이터베이스에 대한 쿼리 성능을 향상시킬 수 있습니다.

## 시스템 최적화

매핑이 비효율적인 연결이나 오버로드된 데이터 통합 서비스 프로세스 시스템에 의존하기 때문에 성능이 저하되는 경우가 많습니다. 라우터, 스위치, 네트워크 프로토콜, 많은 사용자로 인해 시스템 지연이 발생할 수 있습니다.

소스 및 대상 데이터베이스, 소스 및 대상 파일 시스템, 그리고 도메인의 노드에 대한 느린 디스크 액세스가 매핑 성능을 저하시킬 수 있습니다. 시스템 관리자에게 시스템의 하드 디스크를 평가하도록 요청하십시오.

시스템 최적화 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**네트워크 속도를 향상시킵니다.**

네트워크 연결이 느리면 매핑 성능이 저하될 수 있습니다. 시스템 관리자에게 네트워크가 최적의 속도로 실행되는지 확인하도록 요청하십시오. 데이터 통합 서비스 프로세스와 데이터베이스 사이의 네트워크 홉 수를 줄입니다.

**여러 개의 CPU를 사용합니다.**

여러 개의 CPU를 사용하여 여러 개의 매핑을 병렬로 실행할 수 있습니다.

**페이징을 줄입니다.**

운영 체제에서 실제 메모리가 부족해지면 실제 메모리를 확보하기 위해 디스크로 페이징하기 시작합니다. 데이터 통합 서비스 프로세스 시스템의 디스크 페이징을 최소화할 수 있도록 실제 메모리를 구성합니다.

**프로세서 바인딩을 사용합니다.**

다중 프로세서 UNIX 환경에서 데이터 통합 서비스가 대량의 시스템 리소스를 사용할 수 있습니다. 프로세서 바인딩을 사용하여 통합 서비스 프로세스의 프로세서 사용량을 제어하십시오. 소스 및 대상 데이터베이스가 같은 시스템에 있는 경우에도 프로세서 바인딩을 사용하여 데이터베이스가 사용하는 리소스를 제한하십시오.

## 제 8 장

# SQL 데이터 서비스 최적화

이 장에 포함된 항목:

- [SQL 데이터 서비스 최적화 개요, 62](#)
- [타사 클라이언트 도구 최적화, 63](#)
- [SQL 데이터 서비스 최적화 수준, 63](#)
- [SQL 데이터 서비스의 메모리 및 동시 요청에 대한 속성, 66](#)
- [SQL 데이터 서비스에 대한 결과 집합 캐시, 67](#)
- [가상 데이터를 임시 테이블에 보관, 68](#)

## SQL 데이터 서비스 최적화 개요

SQL 데이터 서비스를 최적화하여 최종 사용자가 타사 클라이언트 도구를 사용하여 SQL 데이터 서비스에 대한 SQL 쿼리를 실행할 때 성능을 향상시킬 수 있습니다. SQL 데이터 서비스가 가상 테이블 매핑을 사용하는 경우 소스, 변환 및 매핑을 최적화할 수 있습니다.

다음과 같은 최적화 기법을 사용하여 SQL 데이터 서비스를 최적화할 수 있습니다.

- 타사 클라이언트 도구를 최적화합니다.
- SQL 데이터 서비스 최적화 수준을 구성합니다.
- SQL 데이터 서비스 속성에서 데이터 통합 프로세스에 대한 동시 실행 및 메모리 속성을 구성합니다.
- SQL 데이터 서비스에 대한 데이터 개체 캐싱을 구성합니다.
- SQL 데이터 서비스에 대한 결과 집합 캐싱을 구성합니다.
- SQL 데이터 서비스에서 가상 테이블에 대한 제약 조건을 구성합니다.

관련 항목:

- [“데이터 개체 캐싱” 페이지 58](#)

## 타사 클라이언트 도구 최적화

SQL 데이터 서비스에 대해 SQL 쿼리를 처리하고 실행할 때 타사 클라이언트 도구가 성능을 저하시킬 수 있습니다. 최종 사용자가 SQL 데이터 서비스에 대해 SQL 쿼리를 실행할 때 사용할 수 있는 타사 클라이언트 도구를 최적화하십시오.

타사 클라이언트 도구 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**대규모 쿼리 결과를 디스크의 파일로 보냅니다.**

타사 클라이언트 도구가 대규모 쿼리 결과를 콘솔 창에 표시할 경우 성능이 저하될 수 있습니다.

**암호화를 비활성화하도록 타사 클라이언트 도구를 구성합니다.**

타사 클라이언트 도구가 쿼리를 결과를 가져오거나 표시할 때 데이터를 암호화하는 경우 성능이 저하될 수 있습니다.

**행 집합을 사전에 가져오도록 타사 클라이언트 도구를 구성합니다.**

타사 클라이언트 도구가 행을 한 번에 하나씩 가져오는 경우 성능이 저하될 수 있습니다.

**타사 클라이언트 도구를 구성하여 타사 클라이언트 도구가 처음 로드될 때 테이블에서 콘텐츠를 읽는 옵션을 비활성화합니다.**

타사 클라이언트 도구가 처음 로드될 때 테이블에서 콘텐츠를 읽도록 BLOB 및 CLOB 데이터 유형에 대한 데이터 유형 설정이 구성되어 있고 쿼리에 BLOB 및 CLOB 데이터 유형이 사용되지 않는 경우 성능이 저하될 수 있습니다.

**날짜, 시간 및 타임스탬프에 대해 기본 형식 및 변환 설정을 사용하도록 타사 클라이언트 도구를 구성합니다.**

날짜, 시간 및 타임스탬프에 대한 형식 및 변환 설정이 기본 형식이 아니라 사용자 지정 형식으로 설정되어 있는 경우 타사 클라이언트 도구가 성능을 저하시킬 수 있습니다.

**디버그 옵션을 비활성화하거나 디버그 안 함으로 설정합니다.**

쿼리를 실행하는 디버그 옵션이 추적으로 설정되어 있으면 타사 클라이언트 도구가 성능을 저하시킬 수 있습니다. 쿼리를 처리하는 동안 타사 클라이언트 도구가 디버그 파일에 더 많은 로그 메시지를 쓰기 때문에 성능이 저하되는 것입니다.

## SQL 데이터 서비스 최적화 수준

데이터 통합 서비스는 사용자가 구성한 최적화 수준을 기반으로 SQL 데이터 서비스를 최적화합니다. 보통 수준 이외의 최적화 수준을 사용하는 SQL 데이터 서비스가 필요한 경우 최적화 수준을 구성하십시오. 기본적으로 각 SQL 데이터 서비스는 보통 최적화 수준을 사용합니다.

최적화 수준에 따라 SQL 데이터 서비스에 대한 최적화된 쿼리가 생성되는 방법을 알고 싶다면 SQL 데이터 서비스에 대한 쿼리 계획을 보십시오. 쿼리 계획을 볼 경우 개발자 도구에 최적화 수준에 기반한 최적화된 쿼리의 그래픽 표현과 원래 쿼리의 그래픽 표현이 나타납니다.

다음과 같은 최적화 수준을 구성할 수 있습니다.

### 0(없음)

데이터 통합 서비스가 최적화를 적용하지 않습니다.

### 1(최소)

데이터 통합 서비스가 초기 예측 최적화 방법을 적용합니다.

### 2(보통)

데이터 통합 서비스가 초기 예측, 초기 선택, 분기 잘라내기, 푸시인, 글로벌 조건자 최적화 및 조건자 최적화 방법을 적용합니다.

### 3(전체)

데이터 통합 서비스가 비용 기반, 초기 예측, 초기 선택, 분기 잘라내기, 조건자, 푸시인, 반 조인 및 데이터십 조인 최적화 방법을 적용합니다.

기본값은 2(보통)입니다.

다음 방법 중 하나 이상을 사용하여 SQL 데이터 서비스에 대한 최적화 수준을 구성할 수 있습니다.

- SQL 데이터 서비스의 데이터 미리보기에 대한 최적화 수준을 구성합니다.
- 배포된 SQL 데이터 서비스에 대한 최적화 수준을 구성합니다.
- 배포된 SQL 데이터 서비스에서 실행되는 쿼리의 연결 문자열에 최적화 수준을 구성합니다.

## 데이터 미리보기에 대한 SQL 데이터 서비스 최적화 수준 구성

SQL 데이터 서비스의 출력을 미리 볼 경우 데이터 통합 서비스가 SQL 쿼리를 실행하기 위해 사용하는 최적화 수준을 구성합니다.

1. 개발자 도구에서 **실행 > 실행 대화 상자 열기**를 클릭합니다.

실행 대화 상자가 표시됩니다.

2. **데이터 뷰어 구성**을 클릭합니다.
3. **새로 만들기** 단추를 클릭합니다.
4. 데이터 뷰어 구성에 대한 이름을 입력합니다.
5. **고급** 탭을 클릭합니다.
6. 최적화 수준을 선택합니다.
7. **적용**을 클릭합니다.
8. Click **Close**

개발자 도구가 데이터 뷰어 구성을 생성합니다.

## 배포된 SQL 데이터 서비스에 대한 최적화 수준 구성

데이터 통합 서비스가 배포된 SQL 데이터 서비스에서 SQL 쿼리를 실행하기 위해 사용하는 최적화 수준을 구성합니다. SQL 데이터 서비스 연결에서 최적화 수준을 구성하여 단일 쿼리에 대한 최적화 수준을 재정의하도록 선택할 수 있습니다.

1. **Administrator** 도구에서 데이터 통합 서비스를 선택합니다.
2. **응용 프로그램** 보기를 클릭합니다.
3. 최적화 수준을 구성하려는 SQL 데이터 서비스가 포함된 응용 프로그램을 확장합니다.



4. SQL 데이터 서비스를 선택하고 다음 속성을 편집합니다.

속성	설명
최적화 수준	<p>데이터 통합 서비스에서 개체에 적용하는 최적화 수준입니다. 구성할 최적화 수준에 연결된 숫자 값을 입력합니다. 다음 숫자 값 중 하나를 입력할 수 있습니다.</p> <ul style="list-style-type: none"> <li>- 0. 데이터 통합 서비스가 최적화를 적용하지 않습니다.</li> <li>- 1. 데이터 통합 서비스가 초기 예측 최적화 방법을 적용합니다.</li> <li>- 2. 데이터 통합 서비스가 초기 예측, 초기 선택, 푸시인 및 조건자 최적화 방법을 적용합니다.</li> <li>- 3. 데이터 통합 서비스가 비용 기반, 초기 예측, 초기 선택, 푸시인, 조건자 및 반 조건 최적화 방법을 적용합니다.</li> </ul>

5. 데이터 통합 서비스가 쿼리를 실행하기 위해 사용하는 최적화 수준을 재정의하려면 JDBC URL 또는 ODBC 연결 문자열에 다음 항목을 추가합니다. `SQLDataServiceOptions.optimizeLevel= <numeric_optimizer_level>`.

## SQL 데이터 서비스 쿼리 계획

SQL 데이터 서비스에 대한 쿼리 계획을 표시할 경우 원래 쿼리의 그래픽 표현과 최적화된 쿼리의 그래픽 표현이 나타납니다. 그래픽 표현은 데이터 통합 서비스가 쿼리를 처리하는 방식을 설명합니다. 여기에는 변환과 데이터 통합 서비스가 각 변환을 처리하는 순서가 포함됩니다.

개발자 도구는 사용자가 설정한 최적화 수준을 사용하여 최적화된 쿼리를 생성합니다. 최적화된 쿼리는 데이터 통합 서비스가 쿼리를 실행할 때 쿼리를 표시합니다.

예를 들어 SQL 데이터 서비스에서 CUSTOMERS 가상 테이블을 쿼리한다고 가정합니다. 데이터 뷰어 보기에서 쿼리에 대한 최적화 수준을 보통으로 설정하는 기본 데이터 뷰어 구성 설정을 선택합니다.

데이터 뷰어 보기에서 다음과 같은 쿼리를 입력합니다.

```
select * from CUSTOMERS where CUSTOMER_ID > 150000 order by LAST_NAME
```

SQL 쿼리 계획을 볼 경우 개발자 도구에 다음과 같은 쿼리의 그래픽 표현이 나타납니다.



최적화되지 않은 보기에는 사용자가 입력한 쿼리가 표시됩니다. 개발자 도구에 WHERE 절은 필터 변환으로, ORDER BY 절은 분류기 변환으로 표시됩니다. 개발자 도구에서는 통과 식 변환을 사용하여 포트의 이름을 바꿉니다.

최적화된 쿼리를 볼 경우 개발자 도구에 다음과 같은 쿼리의 그래픽 표현이 나타납니다.



최적화된 보기는 데이터 통합 서비스가 실행하는 쿼리를 표시합니다. 최적화 수준이 보통이므로 데이터 통합 서비스가 필터 조건을 소스 데이터 개체에 푸시합니다. 필터 조건을 푸시하면 데이터 통합 서비스가 소스 데이터 개체에서 읽는 행 수가 줄어들기 때문에 쿼리 성능이 향상됩니다. 최적화되지 않은 쿼리와 유사하게 개발자 도구에는 ORDER BY 절이 분류기 변환으로 표시됩니다. 개발자 도구는 통과 식 변환을 사용하여 사용자가 논리적 변환에 지정한 데이터 유형을 적용합니다.

## SQL 쿼리 계획 보기

가상 테이블 데이터를 미리 볼 때 입력한 SQL 쿼리의 매핑 형식 표현을 보려면 SQL 쿼리 계획을 표시합니다.

1. 가상 테이블이 적어도 하나는 있는 SQL 데이터 서비스를 엽니다.
2. **데이터 뷰어** 보기를 클릭합니다.
3. **입력** 창에 SQL 쿼리를 입력합니다.
4. 선택적으로 쿼리에 적용하려는 최적화 수준이 포함된 데이터 뷰어 구성을 선택합니다.
5. **쿼리 계획 표시**를 클릭합니다.

개발자 도구에 **최적화되지 않음** 탭에서 입력한 상태로 쿼리에 대한 SQL 쿼리 계획이 표시됩니다.

6. 최적화된 쿼리를 보려면 **최적화** 탭을 클릭합니다.

개발자 도구에 최적화된 SQL 쿼리 계획이 표시됩니다.

## SQL 데이터 서비스의 메모리 및 동시 요청에 대한 속성

SQL 데이터 서비스 성능을 최적화하려면 **Administrator** 도구에서 데이터 통합 서비스에 대한 동시 요청 및 메모리 속성을 구성합니다.

다음 테이블에는 SQL 서비스 모듈에 대한 요청당 최대 메모리 속성이 설명되어 있습니다.

속성	설명
요청당 최대 메모리	<p>요청당 최대 메모리 동작은 다음 데이터 통합 서비스 구성에 따라 다릅니다.</p> <ul style="list-style-type: none"> <li>- 서비스가 별도의 로컬 또는 원격 프로세스에서 작업을 실행하거나 서비스 프로세스 속성의 최대 메모리 크기가 0(기본값)입니다.</li> </ul> <p>이 경우 요청당 최대 메모리는 데이터 통합 서비스가 단일 요청에서 자동 캐시 모드를 사용하는 모든 변환에 할당할 수 있는 최대 메모리 양(바이트)입니다. 서비스가 특정 캐시 크기를 가진 변환에 별도로 메모리를 할당합니다. 요청에서 사용한 총 메모리는 요청당 최대 메모리 값을 초과할 수 있습니다.</p> <ul style="list-style-type: none"> <li>- 서비스가 데이터 통합 서비스 프로세스에서 작업을 실행하고 서비스 프로세스 속성의 최대 메모리 크기가 0보다 큼니다.</li> </ul> <p>이 경우 요청당 최대 메모리는 데이터 통합 서비스가 단일 요청에 할당할 수 있는 최대 메모리 양(바이트)입니다. 요청에서 사용한 총 메모리는 요청당 최대 메모리 값을 초과할 수 없습니다.</p> <p>기본값은 50,000,000입니다.</p>

다음 테이블에는 데이터 통합 서비스 프로세스에 대한 최대 힙 크기 속성이 설명되어 있습니다.

속성	설명
최대 힙 크기	<p>데이터 통합 서비스를 실행하는 JVM(Java Virtual Machine)에 할당된 RAM 크기입니다. 이 속성을 사용하여 성능을 향상시킬 수 있습니다. 값에 다음 문자 중 하나를 추가하여 단위를 지정합니다.</p> <ul style="list-style-type: none"> <li>- b: 바이트.</li> <li>- k: 킬로바이트.</li> <li>- m: 메가바이트.</li> <li>- g: 기가바이트.</li> </ul> <p>기본값은 640MB입니다.</p> <p><b>참고:</b> 데이터 통합 서비스가 대량의 데이터를 처리해야 할 경우 힙 크기를 늘리는 것을 고려하십시오.</p>

다음 테이블에는 데이터 통합 서비스 프로세스에 대한 SQL 속성이 설명되어 있습니다.

속성	설명
최대 동시 연결 수	데이터 통합 서비스가 SQL 데이터 서비스에 대해 만들 수 있는 데이터베이스 연결 수를 제한합니다. 기본값은 100입니다.

다음 테이블에는 데이터 통합 서비스에 대한 실행 옵션이 설명되어 있습니다.

속성	설명
최대 주문형 실행 풀 크기	동시에 실행할 수 있는 주문형 작업의 최대 수입니다. 작업에는 데이터 미리 보기, 프로파일링 작업, REST 및 SQL 쿼리, 웹 서비스 요청 및 Developer tool에서 실행되는 매핑이 포함됩니다. 데이터 통합 서비스가 수신하는 모든 작업은 주문형 풀 크기에 영향을 줍니다. 충분한 리소스를 사용할 수 있는 경우 데이터 통합 서비스는 즉시 주문형 작업을 실행합니다. 그렇지 않은 경우 데이터 통합 서비스는 작업을 거부합니다. 기본값은 10입니다.
최대 원시 일괄 실행 풀 크기	원시 환경에서 동시에 실행할 수 있는 배포된 작업의 최대 수입니다. 충분한 리소스를 사용할 수 있는 경우 데이터 통합 서비스는 원시 매핑 작업을 대기열에서 원시 작업 풀로 이동합니다. 기본값은 10입니다.
최대 Hadoop 일괄 실행 풀 크기	Hadoop 환경에서 동시에 실행할 수 있는 배포된 작업의 최대 수입니다. 충분한 리소스를 사용할 수 있는 경우 데이터 통합 서비스는 Hadoop 작업을 대기열에서 Hadoop 작업 풀로 이동합니다. 기본값은 100입니다.
최대 메모리 크기	서비스가 데이터 통합 서비스 프로세스에서 작업을 실행할 때 데이터 통합 서비스가 모든 요청을 동시에 실행하는 데 할당할 수 있는 최대 메모리 양(단위: 바이트)입니다. 데이터 통합 서비스가 별도의 로컬 또는 원격 프로세스에서 작업을 실행하는 경우 서비스가 이 값을 무시합니다. 데이터 통합 서비스가 할당할 수 있는 메모리의 양을 제한하지 않으려면 이 속성을 0으로 설정합니다.  값이 0보다 크면 데이터 통합 서비스가 이 속성을 사용하여 모든 요청을 동시에 실행하는 데 허용되는 최대 총 메모리를 계산합니다. 데이터 통합 서비스는 다음과 같은 방법으로 최대 총 메모리 값을 계산합니다. 최대 메모리 크기 + 최대 힙 크기 + 프로그램 구성 요소를 로드하는 데 필요한 메모리 기본값은 0입니다. <b>참고:</b> 프로파일 또는 데이터 품질 매핑을 실행하는 경우 이 속성을 0으로 설정합니다.

## SQL 데이터 서비스에 대한 결과 집합 캐시

결과 집합 캐시를 구성하면 데이터 통합 서비스가 각 SQL 데이터 서비스 쿼리 및 웹 서비스 요청과 연결된 DTM 프로세스의 결과를 캐싱합니다. 데이터 통합 서비스는 사용자가 구성한 만료 기간 동안 결과를 캐싱합니다. 캐시가 만료되기 전에 클라이언트가 동일한 쿼리를 수행하면 데이터 통합 서비스가 캐싱된 결과를 반환합니다.

결과 집합 캐시 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**SQL 데이터 서비스에 대한 결과 집합 캐시를 구성합니다.**

결과 집합 캐시를 사용하면 데이터 통합 서비스가 SQL 데이터 서비스 쿼리에 대해 캐싱된 결과를 사용할 수 있습니다. 짧은 시간 내에 동일한 쿼리를 실행하는 사용자가 있다면 결과 집합 캐시를 사용하여 동일한 쿼리의 런타임을 줄일 수 있습니다.

데이터 통합 서비스가 캐싱된 결과를 사용하도록 설정하면 데이터 서비스 성능이 향상됩니다. 하지만 동일한 쿼리에 대한 데이터 서비스 처리 시간을 더욱 향상시키려면 메모리에 캐시를 저장할 수 있는 충분한 공간을 할당해야 합니다. 결과를 캐싱하는 데 필요한 양보다 많거나 같은 캐시 메모리 양을 구성하면 시스템 I/O

오버헤드가 줄어들어 성능이 향상됩니다. 데이터 통합 서비스가 캐시 파일을 디스크에 쓸 때 시스템 I/O 오버헤드로 인해 데이터 서비스 처리 시간이 증가합니다.

## SQL 데이터 서비스 결과 집합 캐시 속성

성능을 향상시키려면 데이터 통합 서비스에 대한 결과 집합 캐시 속성을 구성할 수 있습니다. 또한 SQL 데이터 서비스에 대해 결과 집합 캐시를 사용할 수 있는 기간(밀리초)을 구성할 수 있습니다.

다음 테이블에는 데이터 통합 서비스에 대한 결과 집합 캐시 속성이 설명되어 있습니다.

속성	설명
파일 이름 접두사	디스크에 저장되는 모든 결과 집합 캐시 파일의 이름에 대한 접두사입니다. 기본값은 RSCACHE입니다.
암호화 설정	결과 집합 캐시 파일이 128비트 AES 암호화를 사용하여 암호화되는지 여부를 나타냅니다. 유효한 값은 true 또는 false입니다. 기본값은 true입니다.

다음 테이블에는 SQL 데이터 서비스에서 결과 집합 캐시를 사용할 수 있는 기간(밀리초)을 구성하는 속성이 설명되어 있습니다.

속성	설명
결과 집합 캐시 만료 기간	결과 집합 캐시를 사용할 수 있는 기간(밀리초)입니다. -1로 설정하면 캐시가 만료되지 않습니다. 0으로 설정하면 결과 집합 캐싱이 비활성화됩니다. 만료 기간 변경은 기존 캐시에 적용되지 않습니다. 모든 캐시가 동일한 만료 기간을 사용하게 하려면 만료 기간을 변경한 후 결과 집합 캐시를 제거합니다. 기본값은 0입니다.

## SQL 데이터 서비스에 대한 결과 집합 캐싱 활성화

동일한 SQL 데이터 서비스 쿼리에 대해 캐싱된 결과를 사용하려면 데이터 통합 서비스가 결과 집합 캐싱을 사용하도록 구성합니다.

1. **Administrator** 도구에서 데이터 통합 서비스를 선택합니다.
2. **프로세스** 보기를 클릭하여 결과 집합 캐시 속성을 구성합니다.
3. **응용 프로그램** 보기를 클릭한 다음 SQL 데이터 서비스를 클릭하여 결과 집합 캐시 만료 속성을 구성합니다.

## 가상 데이터를 임시 테이블에 보관

임시 테이블은 중간의 임시 데이터를 저장하는 관계형 데이터베이스의 테이블입니다. 일반적으로 복잡한 쿼리를 실행할 때는 조인 정보와 같은 다량의 중간 데이터를 저장할 공간이 필요합니다. 임시 테이블을 구현하면 비즈니스 인텔리전스 도구가 SQL 데이터 서비스가 아닌 임시 테이블에서 이 데이터를 검색할 수 있습니다. 그러면 성능이 개선됩니다.

임시 테이블은 두 가지 면에서 보안을 개선하는 결과도 제공합니다. 첫째, 활성 세션의 사용자만 테이블에 액세스할 수 있습니다. 또한 세션이 활성 상태인 동안에만 테이블이 유지되고 연결이 종료되면 데이터베이스에서 테이블이 삭제됩니다.

## 임시 테이블 구현

임시 테이블을 사용하여 크고 복잡한 쿼리의 성능을 향상시킬 수 있습니다. 동일한 데이터 집합에 대해 관계형 데이터베이스에 있는 임시 테이블에 대한 쿼리가 SQL 데이터 서비스에 대한 반복적인 쿼리보다 더 빠르기 때문에 임시 테이블은 성능을 향상시킵니다.

성능 개선을 위해 임시 테이블을 구현하려면 Informatica 관리자과 비즈니스 인텔리전스 도구 개발자가 작업을 수행해야 합니다.

먼저 Informatica 관리자가 관계형 데이터베이스 연결을 작성하고 이 연결을 사용하도록 데이터 통합 서비스를 구성합니다.

그런 다음 비즈니스 인텔리전스 도구(예: IBM Cognos 또는 SAP Business Objects) 개발자가 비즈니스 인텔리전스 도구와 Informatica SQL 데이터 서비스 간 연결을 작성합니다. 연결은 Informatica ODBC 또는 JDBC 드라이버를 사용합니다.

이러한 연결이 활성화되면 비즈니스 인텔리전스 도구가 임시 테이블을 작성하고 사용하여 많은 양의 중간 데이터를 처리할 수 있습니다.

## 제 9 장

# 웹 서비스 최적화

이 장에 포함된 항목:

- [웹 서비스 최적화 개요, 70](#)
- [HTTP 요청 최적화, 71](#)
- [웹 서비스 메시지 압축, 71](#)
- [웹 서비스 최적화 수준, 71](#)
- [메모리 및 동시 요청에 대한 웹 서비스 속성, 73](#)
- [활성 DTM 인스턴스를 구성하는 웹 서비스 속성, 75](#)
- [웹 서비스 결과 집합 캐싱, 75](#)
- [웹 서비스 로그 관리, 76](#)

## 웹 서비스 최적화 개요

데이터 통합 서비스가 웹 서비스 요청을 실행하는 경우 웹 서비스를 최적화하여 성능을 향상시킬 수 있습니다. 메모리를 관리하고 동시 웹 서비스 요청을 처리하도록 데이터 통합 서비스를 조정합니다. 웹 서비스 성능을 향상시키려면 웹 서비스 메시지 압축을 사용하고, HTTP 요청을 최적화하고, 데이터 개체 및 결과 집합 캐시를 구성하고, 오류 로그 수준을 구성합니다.

다음과 같은 최적화 기법을 사용하여 웹 서비스를 최적화할 수 있습니다.

- HTTP 요청을 최적화합니다.
- 웹 서비스 메시지를 압축합니다.
- 웹 서비스 최적화 수준을 구성합니다.
- 웹 서비스 속성에서 데이터 통합 프로세스에 대한 동시 실행 및 메모리 속성을 구성합니다.
- 데이터 통합 서비스가 DTM 프로세스를 활성 상태로 유지하여 둘 이상의 웹 서비스 요청을 처리할 수 있도록 구성합니다.
- 웹 서비스에 대한 데이터 개체 캐싱을 구성합니다.
- 웹 서비스에 대한 결과 집합 캐싱을 구성합니다.
- 웹 서비스 런타임 오류 로그 수준을 구성합니다.

관련 항목:

- [“데이터 개체 캐싱” 페이지 58](#)

## HTTP 요청 최적화

HTTP 요청을 최적화하여 웹 서버에 대한 요청 수를 줄이십시오.

HTTP 요청 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**웹 서비스 클라이언트에 대한 HTTP 소켓 제한 시간을 줄입니다.**

소켓 제한 시간은 HTTP 요청 시간이 초과되기 전에 클라이언트가 대기하는 기간을 설정합니다. 소켓 제한 시간 값이 크면 웹 서비스 클라이언트가 멈출 수 있습니다.

## 웹 서비스 메시지 압축

공급자와 주고 받는 큰 웹 메시지를 압축하여 웹 서비스 성능을 최적화할 수 있습니다.

웹 서비스 메시지 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**웹 서비스 클라이언트에 대해 SOAP 메시지 압축을 활성화합니다.**

SOAP 메시지 압축을 사용하면 웹 서비스가 압축된 웹 서비스 클라이언트 메시지를 받을 수 있습니다. 웹 서비스는 웹 서비스 클라이언트에서 GZip 압축을 사용하는 SOAP 메시지를 받을 수 있습니다.

데이터 통합 서비스는 웹 서비스에서 응답을 받을 때 SOAP 메시지의 Content-Encoding HTTP 헤더를 확인하고 메시지를 디코딩합니다.

## 웹 서비스 최적화 수준

데이터 통합 서비스는 사용자가 구성한 최적화 수준을 기반으로 웹 서비스를 최적화합니다. 보통 수준 이외의 최적화 수준을 사용하는 웹 서비스가 필요한 경우 최적화 수준을 구성하십시오. 기본적으로 각 웹 서비스는 보통 최적화 수준을 사용합니다.

다음과 같은 최적화 수준 중 하나를 선택할 수 있습니다.

**0(없음)**

데이터 통합 서비스가 최적화를 적용하지 않습니다.

**1(최소)**

데이터 통합 서비스가 초기 예측 최적화 방법을 적용합니다.

**2(보통)**

데이터 통합 서비스가 초기 예측, 초기 선택, 분기 잘라내기, 푸시인, 글로벌 조건자 최적화 및 조건자 최적화 방법을 적용합니다.

**3(전체)**

데이터 통합 서비스가 비용 기반, 초기 예측, 초기 선택, 분기 잘라내기, 조건자, 푸시인, 반 조인 및 데이터셋 조인 최적화 방법을 적용합니다.

기본값은 2(보통)입니다.

다음 방법 중 하나 이상을 사용하여 웹 서비스에 대한 최적화 수준을 구성할 수 있습니다.

- 데이터 통합 서비스에 웹 서비스를 배포하기 전에 웹 서비스의 데이터 미리보기에 대한 최적화 수준을 구성합니다.
- 특정 데이터 통합 서비스에서 실행되는 배포된 웹 서비스에 대한 최적화 수준을 구성합니다.
- 배포된 웹 서비스에 대한 웹 서비스 요청의 헤더에서 최적화 수준을 구성합니다.

## 데이터 미리보기에 대한 웹 서비스 최적화 수준 구성

데이터 통합 서비스가 웹 서비스의 출력을 미리 보기 위해 사용하는 최적화 수준을 구성합니다.

1. 개발자 도구에서 **실행 > 실행 대화 상자 열기**를 클릭합니다.

실행 대화 상자가 표시됩니다.

2. **웹 서비스 구성**을 클릭합니다.
3. **새로 만들기** 단추를 클릭합니다.
4. 웹 서비스 구성의 이름을 입력합니다.
5. **고급** 탭을 클릭합니다.
6. 최적화 수준을 선택합니다.
7. **적용**을 클릭합니다.
8. **닫기**를 클릭합니다.

개발자 도구가 웹 서비스 구성을 생성합니다.

작업 매핑의 출력을 미리 보기 위해 데이터 뷰어를 실행할 때 사용하려는 최적화 수준이 포함된 웹 서비스 구성을 선택합니다.

## 배포된 웹 서비스에 대한 최적화 수준 구성

데이터 통합 서비스가 배포된 웹 서비스를 실행하기 위해 사용하는 최적화 수준을 구성합니다. 웹 서비스 SOAP 요청의 HTTP 헤더에서 최적화 수준을 구성하여 단일 요청에 대한 최적화 수준을 재정의하도록 선택할 수 있습니다.

1. **Administrator** 도구에서 데이터 통합 서비스를 선택합니다.
2. **응용 프로그램** 보기를 클릭합니다.
3. 최적화 수준을 구성하려는 웹 서비스가 포함된 응용 프로그램을 확장합니다.
4. 웹 서비스를 선택하고 다음 속성을 편집합니다.

속성	설명
최적화 수준	데이터 통합 서비스에서 개체에 적용하는 최적화 수준입니다. 구성할 최적화 수준에 연결된 숫자 값을 입력합니다. 다음 숫자 값 중 하나를 입력할 수 있습니다. <ul style="list-style-type: none"><li>- 0. 데이터 통합 서비스가 최적화를 적용하지 않습니다.</li><li>- 1. 데이터 통합 서비스가 초기 예측 최적화 방법을 적용합니다.</li><li>- 2. 데이터 통합 서비스가 초기 예측, 초기 선택, 푸시인 및 조건자 최적화 방법을 적용합니다.</li><li>- 3. 데이터 통합 서비스가 비용 기반, 초기 예측, 초기 선택, 푸시인, 조건자 및 반 조인 최적화 방법을 적용합니다.</li></ul>

5. 웹 서비스 요청에 대한 웹 서비스 최적화 수준을 재정의하려면 웹 서비스 SOAP 요청의 HTTP 헤더에 다음 항목을 포함시키십시오. `WebServiceOptions.optimizeLevel= <numeric_optimizer_level>`.



## 메모리 및 동시 요청에 대한 웹 서비스 속성

웹 서비스 성능을 최적화하려면 **Administrator** 도구에서 각 웹 서비스와 데이터 통합 서비스에 대한 동시 요청 및 메모리 속성을 구성합니다.

다음 테이블에는 웹 서비스 모듈에 대한 요청당 최대 메모리 속성이 설명되어 있습니다.

속성	설명
요청당 최대 메모리	<p>요청당 최대 메모리 동작은 다음 데이터 통합 서비스 구성에 따라 다릅니다.</p> <ul style="list-style-type: none"> <li>서비스가 별도의 로컬 또는 원격 프로세스에서 작업을 실행하거나 서비스 프로세스 속성의 최대 메모리 크기가 0(기본값)입니다.</li> <li>이 경우 요청당 최대 메모리는 데이터 통합 서비스가 단일 요청에서 자동 캐시 모드를 사용하는 모든 변환에 할당할 수 있는 최대 메모리 양(바이트)입니다. 서비스가 특정 캐시 크기를 가진 변환에 별도로 메모리를 할당합니다. 요청에서 사용한 총 메모리는 요청당 최대 메모리 값을 초과할 수 있습니다.</li> <li>서비스가 데이터 통합 서비스 프로세스에서 작업을 실행하고 서비스 프로세스 속성의 최대 메모리 크기가 0보다 큼니다.</li> <li>이 경우 요청당 최대 메모리는 데이터 통합 서비스가 단일 요청에 할당할 수 있는 최대 메모리 양(바이트)입니다. 요청에서 사용한 총 메모리는 요청당 최대 메모리 값을 초과할 수 없습니다.</li> </ul> <p>기본값은 50,000,000입니다.</p>

다음 테이블에는 데이터 통합 서비스에 대한 실행 옵션이 설명되어 있습니다.

속성	설명
최대 메모리 크기	<p>서비스가 데이터 통합 서비스 프로세스에서 작업을 실행할 때 데이터 통합 서비스가 모든 요청을 동시에 실행하는 데 할당할 수 있는 최대 메모리 양(단위: 바이트)입니다. 데이터 통합 서비스가 별도의 로컬 또는 원격 프로세스에서 작업을 실행하는 경우 서비스가 이 값을 무시합니다. 데이터 통합 서비스가 할당할 수 있는 메모리의 양을 제한하지 않으려면 이 속성을 0으로 설정합니다.</p> <p>값이 0보다 크면 데이터 통합 서비스가 이 속성을 사용하여 모든 요청을 동시에 실행하는 데 허용되는 최대 총 메모리를 계산합니다. 데이터 통합 서비스는 다음과 같은 방법으로 최대 총 메모리 값을 계산합니다.</p> <p>최대 메모리 크기 + 최대 힙 크기 + 프로그램 구성 요소를 로드하는 데 필요한 메모리</p> <p>기본값은 0입니다.</p> <p><b>참고:</b> 프로필 또는 데이터 품질 매핑을 실행하는 경우 이 속성을 0으로 설정합니다.</p>

다음 테이블에는 데이터 통합 서비스 프로세스의 HTTP 구성 속성이 설명되어 있습니다.

속성	설명
최대 백로그 요청 수	이 데이터 통합 서비스 프로세스에 대한 대기열에서 대기할 수 있는 최대 HTTP 또는 HTTPS 연결 수입니다. 기본값은 100입니다.
최대 동시 요청 수	<p>이 데이터 통합 서비스 프로세스에 연결될 수 있는 최대 HTTP 또는 HTTPS 연결 수입니다. 최소값은 4입니다. 기본값은 200입니다.</p> <p><b>참고:</b> 웹 서비스의 경우 이 속성은 데이터 통합 서비스가 데이터 통합 서비스 백로그로 요청을 전송하기 전에 수락할 수 있는 웹 서비스 요청 수에 영향을 미칩니다.</p>

다음 테이블에는 데이터 통합 서비스 프로세스에 대해 구성할 수 있는 최대 힙 크기 속성이 설명되어 있습니다.

속성	설명
최대 힙 크기	<p>데이터 통합 서비스를 실행하는 JVM(Java Virtual Machine)에 할당된 RAM 크기입니다. 이 속성을 사용하여 성능을 향상시킬 수 있습니다. 값에 다음 문자 중 하나를 추가하여 단위를 지정합니다.</p> <ul style="list-style-type: none"> <li>- b: 바이트.</li> <li>- k: 킬로바이트.</li> <li>- m: 메가바이트.</li> <li>- g: 기가바이트.</li> </ul> <p>기본값은 640MB입니다.</p> <p><b>참고:</b> 데이터 통합 서비스가 대량의 데이터를 처리해야 할 경우 힙 크기를 늘리는 것을 고려하십시오.</p>

## 동시 웹 서비스 요청에 대한 데이터 통합 서비스 구성 예제

데이터 통합 서비스가 동시 웹 서비스 요청을 처리하는 방법을 구성할 때 최대 동시 요청 수에 대한 값이 웹 서비스 및 데이터 통합 서비스 프로세스에 대한 값과 같은지 확인하십시오.

예를 들어 다음 구성에서 데이터 통합 서비스는 200개의 동시 HTTP 요청을 허용하지만 웹 서비스 동시 요청은 10개만 허용합니다.

속성 유형	속성 이름	구성
데이터 통합 서비스 프로세스	최대 동시 요청 수	200
데이터 통합 서비스 프로세스	최대 백로그 요청 수	500
데이터 통합 서비스	최대 주문형 실행 풀 크기	100
웹 서비스	최대 동시 요청 수	10

데이터 통합 서비스가 20개 웹 서비스 요청을 받은 경우 웹 서비스가 10개의 동시 요청만 허용하므로 10개 웹 서비스 요청이 실패합니다.

웹 서비스가 최대 동시 요청 수에 도달한 경우에도 웹 서비스 요청이 실패하지 않도록 데이터 통합 서비스 프로세스 및 웹 서비스에 대해서도 동일한 최대값을 구성하십시오. 데이터 통합 서비스로 전송된 요청 수가 최대 동시 요청 수 값을 초과하면 추가적인 요청을 처리할 때 데이터 통합 서비스를 사용할 수 있게 될 때까지 추가적인 요청이 백로그에 유지됩니다.

## 활성 DTM 인스턴스를 구성하는 웹 서비스 속성

성능을 향상시키려면 데이터 통합 서비스가 DTM 인스턴스를 활성 상태로 유지하여 둘 이상의 웹 서비스 요청을 처리할 수 있도록 구성합니다. Administrator 도구에서 데이터 통합 서비스에 대한 DTM 활성 상태 유지 시간 속성을 구성할 수 있습니다.

다음 테이블에는 DTM 활성 상태 유지 시간 속성이 설명되어 있습니다.

속성	설명
DTM 활성 상태 유지 시간	마지막 요청을 완료한 후 DTM 인스턴스가 열려 있는 시간(밀리초)입니다. 동일한 작업에 대해 발 행된 웹 서비스 요청은 열려 있는 인스턴스를 재사용할 수 있습니다. 요청을 처리하는 데 필요한 시간이 DTM 인스턴스에 대한 초기화 시간에 비해 작을 경우 활성 상태 유지 시간을 사용하면 성 능을 향상시킬 수 있습니다. 요청이 실패할 경우 DTM 인스턴스가 종료됩니다. 기본값은 5000입니다. <b>참고:</b> 기존 DTM 인스턴스를 사용하여 성능을 향상시키는 기능입니다. 각 요청에 대해 DTM 인스 턴스를 시작하려면 DIS에 추가적인 리소스가 필요합니다. DTM을 활성 상태로 유지하려면 메모리 를 사용해야 하므로, 이 옵션을 구성할 때 메모리 사용량을 고려해야 합니다.

## 웹 서비스 결과 집합 캐싱

결과 집합 캐싱을 구성하면 데이터 통합 서비스가 각 웹 서비스 요청과 연결된 DTM 프로세스의 결과를 캐싱합니  
다. 데이터 통합 서비스는 사용자가 구성한 만료 기간 동안 결과를 캐싱합니다. 캐시가 만료되기 전에 외부 클라  
이언트가 동일한 요청을 하면 데이터 통합 서비스가 캐싱된 결과를 반환합니다.

결과 집합 캐시 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

### 웹 서비스에 대한 결과 집합 캐시를 구성합니다.

결과 집합 캐싱을 사용하면 데이터 통합 서비스가 웹 서비스 요청에 대해 캐싱된 결과를 사용할 수 있습니  
다. 짧은 시간 내에 동일한 쿼리를 실행하는 사용자가 있다면 결과 집합 캐싱을 사용하여 동일한 쿼리의 런  
타임을 줄일 수 있습니다.

웹 서비스가 WSSecurity를 사용할 경우 데이터 통합 서비스가 사용자별로 웹 서비스에 대한 결과 집합 캐시  
를 저장합니다. 데이터 통합 서비스는 웹 서비스 요청의 사용자 이름 토큰에 제공된 사용자 이름을 사용하여  
캐시를 저장합니다. 데이터 통합 서비스는 사용자별 결과를 캐싱한 경우 웹 서비스 요청을 전송한 사용자에  
게 캐싱된 결과만 반환합니다.

## 웹 서비스에 대한 결과 집합 캐싱 활성화

동일한 웹 서비스 요청에 대해 캐싱된 결과를 사용하려면 데이터 통합 서비스가 결과 집합 캐싱을 사용하도록 구  
성합니다.

1. Administrator 도구에서 데이터 통합 서비스를 선택합니다.
2. 프로세스 보기를 클릭하여 결과 집합 캐시 속성을 구성합니다.
3. 응용 프로그램 보기, 웹 서비스, 작업을 차례로 클릭하고 웹 서비스 작업 속성에서 캐시 만료 기간을 구성합  
니다. 데이터 통합 서비스가 사용자별 결과를 캐싱하게 하려면 웹 서비스 속성에서 WS-Security를 활성화  
합니다.
4. 웹 서비스 작업이 결과 집합을 캐싱하도록 구성되어 있을 때 웹 서비스 요청에 대한 결과 집합 캐싱을 비활  
성화하려면 SOAP 요청의 HTTP 헤더에 다음 구문을 포함시킵니다.

```
WebServiceOptions.disableResultSetCache=true
```

## 웹 서비스 로그 관리

데이터 통합 서비스가 많은 수의 로그 파일을 쓰고, 유지 관리하는 경우 시스템 I/O 성능이 저하될 수 있습니다. 데이터 통합 서비스는 사용자가 구성한 추적 수준을 기반으로 웹 서비스 런타임 로그를 생성합니다. 데이터 통합 서비스가 쓰고, 유지 관리하는 로그 파일의 수를 관리하는 것이 좋습니다.

웹 서비스 로그 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

### 웹 서비스 추적 수준을 꺼짐으로 설정합니다.

배포된 웹 서비스에 대한 웹 서비스 속성을 구성할 때 로그 추적 수준을 지정할 수 있습니다. 추적 수준은 데이터 통합 서비스가 런타임 로그 위치에 쓰는 로그 유형을 결정합니다. 기본 웹 서비스 추적 수준은 정보입니다. 추적 수준이 최상이나 모두로 설정되어 있으면 데이터 통합 서비스가 로그 파일에 많은 로그를 쓰기 때문에 성능이 저하될 수 있습니다. 추적 수준을 최상이나 모두로 설정한 경우 성능에 미치는 영향은 웹 서비스가 HTTPS 및 WS-Security를 사용할 때 가장 큼니다.

### 더 이상 필요하지 않은 로그 파일을 보관합니다.

너무 많은 로그 파일을 저장하면 시스템 I/O가 영향을 받습니다. 기본적으로 데이터 통합 서비스는 다음 디렉터리에 웹 서비스 런타임 로그를 씁니다. <InformaticaInstallationDir>/tomcat/bin/disLogs/ws

**참고:** 로그를 비우기 위해 ws 폴더를 삭제한 경우 ws 폴더를 다시 생성해야 합니다. ws 폴더를 삭제하고 다시 생성할 때에는 먼저 데이터 통합 서비스를 중지하십시오.

### 데이터 통합 서비스에서 로그 건너뛰기 속성 활성화

웹 서비스 요청이 완료될 때 데이터 통합 서비스에서 로그 파일을 생성하지 않도록 하려면 **로그 건너뛰기** 속성을 활성화합니다. 데이터 통합 서비스에서 웹 서비스에 대한 추적 수준을 INFO 이상으로 설정해야 합니다.

## 제 10 장

# 연결 최적화

이 장에 포함된 항목:

- [연결 최적화 개요, 77](#)
- [연결 풀링, 77](#)
- [데이터베이스 네트워크 패킷 크기, 78](#)

## 연결 최적화 개요

연결을 최적화하여 성능을 개선할 수 있습니다. 데이터베이스 연결의 유휴 연결 인스턴스 풀을 관리할 수 있습니다. 네트워크 패킷 크기를 늘리면 네트워크를 통해 한 번에 더 많은 데이터 패킷을 전송할 수 있습니다.

다음 기법을 사용하여 연결을 최적화할 수 있습니다.

- 연결 풀링 최적화
- 데이터베이스 네트워크 패킷 크기 최적화

## 연결 풀링

연결 풀링은 데이터 통합 서비스에 사용되는 데이터베이스 연결 정보를 캐싱하기 위한 프레임워크로, 캐싱된 연결 정보를 재사용하여 성능을 향상시킵니다.

연결 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**데이터베이스 연결에 대해 연결 풀링을 활성화합니다.**

연결 풀링을 활성화하여 연결 성능을 최적화합니다. 데이터베이스 연결의 유휴 연결 인스턴스를 관리할 수 있습니다. 연결 풀은 사용자가 구성한 풀링 속성을 기반으로 유휴 연결 인스턴스를 유지합니다. 최대 및 최소 유휴 연결 수와 유휴 연결의 최대 대기 시간을 조정할 수 있습니다.

## 연결 개체의 풀링 속성

연결 풀링 속성은 데이터베이스 연결의 **풀링** 보기에서 편집할 수 있습니다.

연결 풀 라이브러리의 수는 실행 중인 데이터 통합 서비스 프로세스 또는 DTM 프로세스의 수에 따라 다릅니다. 각각의 데이터 통합 서비스 프로세스 또는 DTM 프로세스가 고유한 연결 풀 라이브러리를 유지 관리합니다. 풀링 속성의 값은 각 연결 풀 라이브러리에 대한 것입니다.

예를 들어 최대 연결 수를 15로 설정한 경우 각 연결 풀 라이브러리는 최대 15개의 유휴 연결을 풀에 유지할 수 있습니다. 데이터 통합 서비스가 별도의 로컬 프로세스에서 작업을 실행하고 세 개의 DTM 프로세스가 실행되고 있는 경우 최대 45개의 유휴 연결 인스턴스가 있을 수 있습니다.

유휴 연결 인스턴스의 전체 수를 줄이려면 최소 연결 수를 0으로 설정하고 각 데이터베이스 연결의 최대 유휴 시간을 줄입니다.

다음 목록에는 데이터베이스 연결의 **풀링** 보기에서 편집할 수 있는 데이터베이스 연결 풀링 속성이 설명되어 있습니다.

#### 연결 풀링 활성화

연결 풀링을 활성화합니다. 연결 풀링을 활성화하면 각 연결 풀의 유휴 연결 인스턴스가 메모리에 유지됩니다. 유휴 연결의 풀을 삭제하려면 데이터 통합 서비스를 다시 시작해야 합니다.

연결 풀링을 비활성화하면 DTM 프로세스 또는 데이터 통합 서비스 프로세스가 모든 풀링 활동을 중지합니다. DTM 프로세스 또는 데이터 통합 서비스 프로세스가 작업을 처리할 때마다 연결 인스턴스를 작성하고 작업 처리가 끝나면 인스턴스를 삭제합니다.

DB2 for i5/OS, DB2 for z/OS, IBM DB2, Microsoft SQL Server, Oracle 및 ODBC 연결의 경우 기본값은 활성화됩니다. Adabas, IMS, Sequential 및 VSAM 연결의 경우 기본값은 비활성화됩니다.

#### 최소 연결 수

최대 유휴 시간에 도달한 후 데이터베이스 연결을 위해 풀에 유지되는 유휴 연결 인스턴스의 최소 수입니다. 이 값은 최대 유휴 연결 인스턴스 수보다 작거나 같게 설정합니다. 기본값은 0입니다.

#### 최대 연결 수

최소 유휴 시간에 도달하기 전에 데이터베이스 연결을 위해 풀에 유지되는 유휴 연결 인스턴스의 최대 수입니다. 이 값은 최소 유휴 연결 인스턴스 수보다 크게 설정합니다. 기본값은 15입니다.

#### 최대 유휴 시간

최소 연결 인스턴스 수를 초과하는 연결 인스턴스가 연결 풀에서 삭제되기 전에 유휴 상태로 유지될 수 있는 시간(초)입니다. 연결 인스턴스가 최소 유휴 연결 인스턴스 수를 초과하지 않는 경우에는 연결 풀이 유휴 시간을 무시합니다. 기본값은 120입니다.

## 데이터베이스 네트워크 패킷 크기

Oracle, Sybase ASE 또는 Microsoft SQL Server 대상에 대해 읽기나 쓰기를 수행하는 경우 대상 데이터베이스를 기준으로 네트워크 패킷 크기를 늘려 성능을 향상시킬 수 있습니다. 네트워크 패킷 크기를 늘리면 네트워크를 통해 한 번에 더 많은 데이터 패킷을 전송할 수 있습니다.

데이터베이스 네트워크 패킷 크기와 관련된 병목 현상이 있는 경우 다음과 같은 해결 방법을 고려하십시오.

**Oracle 데이터베이스의 데이터베이스 네트워크 패킷 크기를 늘립니다.**

listener.ora 및 tnsnames.ora에서 데이터베이스 서버 네트워크 패킷 크기를 늘릴 수 있습니다. 필요한 경우 데이터베이스 설명서에서 패킷 크기 늘리기에 대한 추가 정보를 참조하십시오.

**Sybase ASE 데이터베이스의 데이터베이스 네트워크 패킷 크기를 늘립니다.**

패킷 크기를 늘리는 방법에 대한 자세한 내용은 데이터베이스 설명서를 참조하십시오. 데이터베이스 서버 패킷 크기를 반영하려면 데이터 통합 서비스의 관계형 연결 개체에서 Sybase ASE에 대한 패킷 크기도 변경해야 합니다.

Microsoft SQL Server 데이터베이스의 데이터베이스 네트워크 패킷 크기를 늘립니다.

패킷 크기를 늘리는 방법에 대한 자세한 내용은 데이터베이스 설명서를 참조하십시오. 데이터베이스 서버 패킷 크기를 반영하려면 데이터 통합 서비스의 관계형 연결 개체에서 Microsoft SQL Server에 대한 패킷 크기도 변경해야 합니다.

# 인덱스

## B

- 푸시다운 최적화
  - 설명 [45](#)
- 푸시다운 최적화 방법
  - 소스 푸시다운 [46](#)
  - 전체 푸시다운 [45](#)
- 푸시인 최적화
  - SQL 변환 [32](#)
  - SQL 변환에서 활성화 [33](#)
  - 설명 [44](#)
  - 웹 서비스 소비자 변환 [35](#)
- 플랫 파일
  - 분할을 위한 대상 최적화 [51](#)
  - 분할을 위한 소스 최적화 [50](#)
- 플랫 파일 대상
  - 대상 최적화 [14](#)
- 플랫 파일 소스
  - 소스 최적화 [18](#)
- 필터 최적화
  - 매핑 최적화 [47](#)
- 필터 포트
  - 웹 서비스 소비자 변환 [35](#)
- 활성 DTM 인스턴스
  - 웹 서비스 [75](#)
- 힌트
  - 쿼리 보기 [20](#)

## H

- HTTP 요청 최적화
  - 웹 서비스 최적화 [71](#)

## J

- Java 변환
  - 변환 최적화 [26](#)
- JDBC 드라이버
  - 런타임 최적화 [63](#)

## N

- 논리적 데이터 개체
  - 데이터베이스에서 캐싱 [58](#)
- 단일 패스 읽기
  - 매핑 최적화 [47](#)
- 대량 로드
  - 대상 최적화 [15](#)
- 대상 최적화
  - Oracle 데이터베이스 최적화 [15](#)
  - 대량 로드 [15](#)
  - 데이터베이스 검사점 간격 [15](#)

- 대상 최적화 (계속)
  - 플랫 파일 대상 [14](#)
- 데이터 개체 캐시
  - 구성 [58](#)
  - 사용자 관리 테이블 [58](#)
  - 설명 [58](#)
  - 인덱스 캐시 [58](#)
  - 최적화 [60](#)
  - 테이블 데이터 유형 [59](#)
- 데이터 유형 변환 최적화
  - 매핑 최적화 [48](#)
- 데이터 통합 서비스
  - SQL 데이터 서비스 결과 집합 캐시 [67](#)
  - 웹 서비스 결과 집합 캐시 [75](#)
- 데이터 통합 서비스 최적화
  - 런타임 최적화 [55](#)
- 데이터베이스
  - 분할을 위한 대상 최적화 [52](#)
  - 분할을 위한 소스 최적화 [51](#)
- 데이터베이스 검사점 간격
  - 대상 최적화 [15](#)
- 데이터베이스 네트워크 패킷 크기
  - 연결 최적화 [78](#)
- 데이터베이스 힌트
  - Developer 도구에서 입력 [20](#)
- 데이터십 조인 최적화
  - 설명 [42](#)
- 동시 요청
  - SQL 데이터 서비스 [66](#)
  - 웹 서비스 [73](#)
- 런타임 최적화
  - 데이터 통합 서비스 최적화 [55](#)
  - 모니터링 통계 [56](#)
  - 모델 리포지토리 서비스 최적화 [56](#)
  - 분석 서비스 최적화 [54](#)
  - 시스템 최적화 [61](#)
- 매핑
  - 글로벌 조건자 최적화 방법 [44](#)
  - 분할된 최적화 [49](#)
  - 조건자 최적화 방법 [40](#)
  - 최적화 방법 [39](#)
- 매핑 최적화
  - 단일 패스 읽기 [47](#)
  - 데이터 유형 변환 최적화 [48](#)
  - 식 최적화 [24](#)
  - 오류 추적 [48](#)
  - 필터 최적화 [47](#)
- 메모리 할당
  - SQL 데이터 서비스 [66](#)
  - 동시 요청 [66](#)
  - 웹 서비스 [75](#)
  - 활성 DTM 인스턴스 [75](#)
- 모니터링 통계
  - 런타임 최적화 [56](#)



모델 리포지토리 서비스 최적화

런타임 최적화 [56](#)

반 조인 최적화

설명 [43](#)

변환

분할을 위한 최적화 [53](#)

변환 오류 제거

변환 최적화 [34](#)

변환 최적화

Java 변환 [26](#)

SQL 변환 [32](#)

변환 오류 제거 [34](#)

변환 캐시 [33](#)

분류기 변환 [31](#)

웹 서비스 소비자 변환 [35](#)

조이너 변환 [28](#)

조희 변환 [29](#)

집계 변환 [23](#)

변환 캐시

변환 최적화 [33](#)

병목 현상

UNIX의 경우 [11](#)

Windows의 경우 [11](#)

보통 최적화 수준

설명 [39](#)

부작용

SQL 변환 [32](#)

설명 [34](#)

웹 서비스 소비자 변환 [35](#)

부작용 있음

변환 속성 설명 [34](#)

분기 잘라내기 최적화

설명 [44](#)

분류기 변환

변환 최적화 [31](#)

분석 서비스 최적화

런타임 최적화 [54](#)

분할

다중 CPU [49](#)

대상 데이터베이스 최적화 [52](#)

변환 최적화 [53](#)

소스 데이터베이스 최적화 [51](#)

최적화 [49](#)

플랫 파일 대상 최적화 [51](#)

플랫 파일 소스 최적화 [50](#)

비용 기반 최적화

설명 [41](#)

사용자 지정 데이터 개체

소스 최적화 [22](#)

성능 조정

글로벌 조건자 최적화 방법 [44](#)

데이터십 조인 최적화 방법 [42](#)

반 조인 최적화 방법 [43](#)

분기 잘라내기 최적화 방법 [44](#)

비용 기반 최적화 방법 [41](#)

조건자 최적화 방법 [40](#)

초기 선택 최적화 방법 [43](#)

초기 예측 최적화 방법 [40](#)

최적화 방법 [39](#)

최적화 수준 [39](#)

푸시다운 최적화 방법 [45](#)

푸시인 최적화 방법 [44](#)

프로세스 개요 [9](#)

소스 최적화

Oracle 데이터베이스 최적화 [22](#)

고유 항목 선택 [19](#)

사용자 지정 데이터 개체 [22](#)

소스 최적화 (계속)

제약 조건 [21](#)

조건부 필터 [19](#)

쿼리 최적화 [18](#)

플랫 파일 소스 [18](#)

시스템

UNIX의 병목 현상, 식별 [11](#)

Windows의 병목 현상, 식별 [11](#)

시스템 최적화

런타임 최적화 [61](#)

식 최적화

매핑 최적화 [24](#)

연결 최적화

데이터베이스 네트워크 패킷 크기 [78](#)

연결 풀링 [77](#)

연결 풀링

속성 [77](#)

연결 최적화 [77](#)

오류 추적

매핑 최적화 [48](#)

오류 추적 수준

웹 서비스 로그 관리 [76](#)

웹 서비스

동시 요청 [73](#)

메모리 할당 [75](#)

웹 서비스 결과 집합 캐시

데이터 통합 서비스 [75](#)

웹 서비스 로그 관리

오류 추적 수준 [76](#)

웹 서비스 메시지 압축

웹 서비스 최적화 [71](#)

웹 서비스 소비자 변환

변환 최적화 [35](#)

초기 선택 최적화 [35](#)

푸시인 최적화 [35](#)

푸시인 최적화 활성화 [36](#)

필터 최적화 [35](#)

웹 서비스 최적화

HTTP 요청 최적화 [71](#)

웹 서비스 메시지 압축 [71](#)

임시 테이블

설명 [68](#)

전체 최적화 수준

설명 [39](#)

제약 조건

소스 최적화 [21](#)

제약 조건 구성 [21](#)

조건부 필터

소스 최적화 [19](#)

조이너 변환

변환 최적화 [28](#)

조희 변환

변환 최적화 [29](#)

집계 변환

변환 최적화 [23](#)

초기 선택 최적화

SQL 변환 [32](#)

설명 [43](#)

웹 서비스 소비자 변환 [35](#)

초기 예측 최적화

설명 [40](#)

최대 병렬도

늘리기 [50](#)

최소 최적화 수준

설명 [39](#)

최적화

데이터십 조인 최적화 방법 [42](#)

## 최적화 (계속)

- 매핑 성능 방법 [39](#)
- 반 조인 최적화 방법 [43](#)
- 부작용 [34](#)
- 분기 잘라내기 최적화 방법 [44](#)
- 비용 기반 최적화 방법 [41](#)
- 초기 선택 최적화 방법 [43](#)
- 초기 예측 최적화 방법 [40](#)
- 푸시다운 최적화 방법 [45](#)
- 푸시인 최적화 방법 [44](#)
- 최적화 수준
  - 설명 [39](#)
- 쿼리 보기
  - 힌트 구성 [20](#)
- 쿼리 최적화
  - 소스 최적화 [18](#)
- 타사 클라이언트 도구
  - 런타임 최적화 [63](#)

## O

- Oracle 데이터베이스 최적화
  - 대상 최적화 [15](#)
  - 소스 최적화 [22](#)

## S

- SQL 데이터 서비스
  - 메모리 할당 [66](#)
- SQL 데이터 서비스 결과 집합 캐시
  - 데이터 통합 서비스 [67](#)
- SQL 데이터 서비스 최적화
  - JDBC 드라이버 [63](#)
  - 타사 클라이언트 도구 [63](#)

- SQL 데이터 서비스에 대한 결과 집합 캐시 활성화
  - 결과 집합 캐시 [68](#)

- SQL 변환
  - 변환 최적화 [32](#)
  - 초기 선택 최적화 [32](#)
  - 푸시인 최적화 [32](#)
  - 푸시인 최적화 속성 [33](#)

- SQL 쿼리 계획
  - 보기 [66](#)
- SQL 힌트
  - Developer 도구에서 입력 [20](#)

## U

- UNIX
  - 시스템 병목 현상 [11](#)

## W

- Windows
  - 병목 현상 [11](#)

## ㄱ

- 가상 테이블
  - 데이터베이스에서 캐싱 [58](#)
- 결과 집합 캐시
  - SQL 데이터 서비스에 대한 결과 집합 캐시 활성화 [68](#)
  - 결과 집합 캐시 속성 [68](#)
- 결과 집합 캐시 속성
  - 런타임 최적화 [68](#)
- 고유 항목 선택
  - 소스 최적화 [19](#)