



Informatica® PowerCenter
10.5.6

성능 조정 가이드

Informatica PowerCenter 성능 조정 가이드

10.5.6

2024년5월

© 저작권 Informatica LLC 2000, 2024

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, Informatica 로고 및 PowerCenter는 미국과 전 세계 여러 관할 국가에서 Informatica LLC의 상표 또는 등록 상표입니다. Informatica 상표의 현재 목록은 <https://www.informatica.com/trademarks.html>에서 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 설명서 중 일부는 타사 저작권의 적용을 받으며, 이에 국한되지 않습니다. 저작권 DataDirect Technologies. 모든 권리 보유. 저작권 (c) Sun Microsystems. 모든 권리 보유. 저작권 (c) RSA Security Inc. 모든 권리 보유. 저작권 (c) Ordinal Technology Corp. 모든 권리 보유. 저작권 (c) Aandacht c.v. 모든 권리 보유. 저작권 Genivia, Inc. 모든 권리 보유. 저작권 Isomorphic Software. 모든 권리 보유. 저작권 (c) Meta Integration Technology, Inc. 모든 권리 보유. 저작권 (c) Intalio. 모든 권리 보유. 저작권 (c) Oracle. 모든 권리 보유. 저작권 (c) Adobe Systems Incorporated. 모든 권리 보유. 저작권 (c) DataArt, Inc. 모든 권리 보유. 저작권 (c) ComponentSource. 모든 권리 보유. 저작권 (c) Microsoft Corporation. 모든 권리 보유. 저작권 (c) Rogue Wave Software, Inc. 모든 권리 보유. 저작권 (c) Teradata Corporation. 모든 권리 보유. 저작권 (c) Yahoo! Inc. 모든 권리 보유. 저작권 (c) Glyph & Cog, LLC. 모든 권리 보유. 저작권 (c) Thinkmap, Inc. 모든 권리 보유. 저작권 (c) Clearpace Software Limited. 모든 권리 보유. 저작권 (c) Information Builders, Inc. 모든 권리 보유. 저작권 (c) OSS Nokalva, Inc. 모든 권리 보유. 저작권 Edifecs, Inc. 모든 권리 보유. 저작권 Cleo Communications, Inc. 모든 권리 보유. 저작권 (c) International Organization for Standardization 1986. 모든 권리 보유. 저작권 (c) ej-technologies GmbH. 모든 권리 보유. 저작권 (c) Jaspersoft Corporation. 모든 권리 보유. 저작권 (c) International Business Machines Corporation. 모든 권리 보유. 저작권 (c) yWorks GmbH. 모든 권리 보유. 저작권 (c) Lucent Technologies. 모든 권리 보유. 저작권 (c) University of Toronto. 모든 권리 보유. 저작권 (c) Daniel Veillard. 모든 권리 보유. 저작권 (c) Uniconic, Inc. 저작권 IBM Corp. 모든 권리 보유. 저작권 (c) MicroQuill Software Publishing, Inc. 모든 권리 보유. 저작권 (c) PassMark Software Pty Ltd. 모든 권리 보유. 저작권 (c) LogiXML, Inc. 모든 권리 보유. 저작권 (c) 2003-2010 Lorenzi Davide, 모든 권리 보유. 저작권 (c) Red Hat, Inc. 모든 권리 보유. 저작권 (c) The Board of Trustees of the Leland Stanford Junior University. 모든 권리 보유. 저작권 (c) EMC Corporation. 모든 권리 보유. 저작권 (c) Flexera Software. 모든 권리 보유. 저작권 (c) Jinfonet Software. 모든 권리 보유. 저작권 (c) Apple Inc. 모든 권리 보유. 저작권 (c) Telerik Inc. 모든 권리 보유. 저작권 (c) BEA Systems. 모든 권리 보유. 저작권 (c) PDFlib GmbH. 모든 권리 보유. 저작권 (c) Orientation in Objects GmbH. 모든 권리 보유. 저작권 (c) Tanuki Software, Ltd. 모든 권리 보유. 저작권 (c) Ricebridge. 모든 권리 보유. 저작권 (c) Sencha, Inc. 모든 권리 보유. 저작권 (c) Scalable Systems, Inc. 모든 권리 보유. 저작권 (c) jQWidgets. 모든 권리 보유. 저작권 (c) Tableau Software, Inc. 모든 권리 보유. 저작권 (c) MaxMind, Inc. 모든 권리 보유. 저작권 (c) TMate Software s.r.o. 모든 권리 보유. 저작권 (c) MapR Technologies Inc. 모든 권리 보유. 저작권 (c) Amazon Corporate LLC. 모든 권리 보유. 저작권 (c) Highsoft. 모든 권리 보유. 저작권 (c) Python Software Foundation. 모든 권리 보유. 저작권 (c) BeOpen.com. 모든 권리 보유. 저작권 (c) CNRI. 모든 권리 보유.

이 제품에는 Apache Software Foundation(<http://www.apache.org/>)에서 개발한 소프트웨어 및/또는 Apache License의 다양한 버전("라이선스")에 따라 사용이 허가 된 기타 소프트웨어가 포함되어 있습니다. <http://www.apache.org/licenses/>에서 이러한 라이선스의 복사본을 얻을 수 있습니다. 관련 법규 또는 서면 동의에 명시되어 있지 않은 경우, 이러한 라이선스에 따라 배포되는 소프트웨어는 어떠한 종류의 명시적이거나 묵시적인 보증 또는 조건 없이 "있는 그대로" 배포됩니다. 사용 권한에 대한 특정 언어별 라이선스 및 해당 라이선스에 따른 제한 사항을 참조하십시오.

이 제품에는 Mozilla(<http://www.mozilla.org/>)에서 개발한 소프트웨어, JBoss Group, LLC(저작권 JBoss Group, LLC, 모든 권리 보유.)가 저작권을 소유한 소프트웨어, Bruno Lowagie and Paulo Soares(저작권 (c) 1999-2006 by Bruno Lowagie and Paulo Soares)가 저작권을 소유한 소프트웨어 및 GNU Lesser General Public License Agreement(<http://www.gnu.org/licenses/lgpl.html>)의 다양한 버전에 따라 라이선스가 부여된 기타 소프트웨어가 포함되어 있습니다. 해당 정보는 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 이에 국한되지 않는 어떠한 종류의 명시적이거나 묵시적인 보증 없이 "있는 그대로" 제공되며, Informatica는 어떠한 책임도 지지 않습니다.

이 제품에는 Douglas C. Schmidt와 Washington University, University of California, Irvine, Vanderbilt University의 연구팀(저작권 ((c)) 1993-2006, 모든 권리 보유.)이 저작권을 소유한 ACE(TM) 및 TAO(TM) 소프트웨어가 포함되어 있습니다.

이 제품에는 OpenSSL Toolkit(저작권 The OpenSSL Project. 모든 권리 보유.)에서 사용할 수 있도록 OpenSSL Project에서 개발한 소프트웨어가 포함되어 있으며 이 소프트웨어의 재배포는 <http://www.openssl.org> 및 <http://www.openssl.org/source/license.html>의 조항에 따라 변경될 수 있습니다.

이 제품에는 Curl 소프트웨어(저작권 1996-2013, Daniel Stenberg, <daniel@haxx.se>. 모든 권리 보유.)가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한 사항은 <http://curl.haxx.se/docs/copyright.html>에 명시된 조항에 따라 변경될 수 있습니다. 위와 같은 저작권 고지 및 이러한 허가 고지가 모든 제품에 표시되어 있는 경우 목적 및 사용권 유무에 관계없이 이 소프트웨어는 사용, 수정 및 배포할 수 있는 사용 권한이 부여됩니다.

이 제품에는 MetaStuff, Ltd(저작권 2001-2005 ((C)) MetaStuff, Ltd. 모든 권리 보유.)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://www.dom4j.org/license.html>의 조항에 따라 변경될 수 있습니다.

이 제품에는 Per Bothner(저작권 (c) 1996-2006 Per Bothner. 모든 권리 보유.)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이러한 정보를 사용할 수 있는 권리는 <http://www.gnu.org/software/kawa/Software-License.html>의 라이선스에 설명되어 있습니다.

이 제품에는 OSSP UUID 소프트웨어(저작권 (c) 2002 Ralf S. Engelschall, 저작권 (c) 2002 The OSSP Project 저작권 (c) 2002 Cable & Wireless Deutschland)가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://www.opensource.org/licenses/mit-license.php>의 조항에 따라 변경될 수 있습니다.

이 제품에는 Boost(<http://www.boost.org/>)에서 개발하거나 Boost 소프트웨어 라이선스에 따라 개발된 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 http://www.boost.org/LICENSE_1_0.txt의 조항에 따라 변경될 수 있습니다.

이 제품에는 University of Cambridge(저작권 (c) 1997-2007 University of Cambridge)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://www.pcre.org/license.txt>의 조항에 따라 변경될 수 있습니다.

이 제품에는 Eclipse Foundation(저작권 (c) 2007 The Eclipse Foundation. 모든 권리 보유.)이 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://www.eclipse.org/org/documents/epl-v10.php> 및 <http://www.eclipse.org/org/documents/edl-v10.php>의 조항에 따라 변경될 수 있습니다.

이 제품에는 <http://www.tcl.tk/software/tcltk/license.html>, <http://www.bosrup.com/web/overlib/?License>, <http://www.stlport.org/doc/license.html>, <http://asm.ow2.org/license.html>, <http://www.cryptix.org/LICENSE.TXT>, <http://hsqldb.org/web/hsqLicense.html>, <http://httpunit.sourceforge.net/doc/license.html>, <http://jung.sourceforge.net/license.txt>, http://www.gzip.org/zlib_license.html, <http://www.openldap.org/software/release/license.html>, <http://www.libssh2.org>, <http://slf4j.org/license.html>, <http://www.sente.ch/software/OpenSourceLicense.html>, <http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3-license-agreement>, <http://antlr.org/license.html>, <http://aopalliance.sourceforge.net/>, <http://www.bouncycastle.org/licence.html>, <http://www.jgraph.com/jgraphdownload.html>, <http://www.jcraft.com/jsch/LICENSE.txt>, http://jotm.objectweb.org/bsd_license.html, <http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>, <http://www.slf4j.org/license.html>, <http://nanoxml.sourceforge.net/orig/copyright.html>, <http://www.json.org/license.html>, <http://forge.ow2.org/projects/javaservice/>, <http://www.postgresql.org/about/licence.html>, <http://www.sqlite.org/copyright.html>, <http://www.tcl.tk/software/tcltk/license.html>, <http://www.jaxen.org/faq.html>, <http://www.jdom.org/docs/faq.html>, <http://www.slf4j.org/license.html>, <http://www.iodbc.org/dataspace/iodbc/wiki/IODBC/License>, <http://www.keplerproject.org/md5/license.html>, <http://www.toedter.com/en/jcalendar/license.html>, <http://www.edankert.com/bounce/index.html>, <http://www.net-snmp.org/about/license.html>, <http://www.openmdx.org/#FAQ>, http://www.php.net/license/3_01.txt, <http://srp.stanford.edu/license.txt>, <http://www.schneier.com/blowfish.html>, <http://www.jmock.org/license.html>, <http://xsom.java.net>, <http://benalman.com/about/license/>, <https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js>, <http://>

www.h2database.com/html/license.html#summary, <http://jsoncpp.sourceforge.net/LICENSE>, <http://jdbc.postgresql.org/license.html>, <http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto>, <https://github.com/rantav/hector/blob/master/LICENSE>, <http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html>, <http://jibx.sourceforge.net/jibx-license.html>, <https://github.com/lyokato/libgeohash/blob/master/LICENSE>, <https://github.com/hjiang/jsonxx/blob/master/LICENSE>, <https://code.google.com/p/lz4/>, <https://github.com/jedisct1/libsodium/blob/master/LICENSE>, <http://one-jar.sourceforge.net/index.php?page=documents&file=license>, <https://github.com/EsotericSoftware/kryo/blob/master/license.txt>, <http://www.scala-lang.org/license.html>, <https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt>, <http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html>, <https://aws.amazon.com/asl/>, <https://github.com/twbs/bootstrap/blob/master/LICENSE> 및 <https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt>.

이 제품에는 Academic Free License(<http://www.opensource.org/licenses/afl-3.0.php>), Common Development and Distribution License(<http://www.opensource.org/licenses/cddl1.php>), Common Public License (<http://www.opensource.org/licenses/cpl1.0.php>), Sun Binary Code License Agreement Supplemental License Terms, BSD License(<http://www.opensource.org/licenses/bsd-license.php>), 새 BSD License(<http://opensource.org/licenses/BSD-3-Clause>), MIT License(<http://www.opensource.org/licenses/mit-license.php>), Artistic License(<http://www.opensource.org/licenses/artistic-license-1.0>) 및 Initial Developer's Public License 버전 1.0(<http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/>)에 따라 라이선스가 부여된 소프트웨어가 포함되어 있습니다.

이 제품에는 Joe Walnes와 XStream Committers(저작권 (c) 2003-2006 Joe Walnes, 2006-2007 XStream Committers. 모든 권리 보유.)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://xstream.codehaus.org/license.html>의 조항에 따라 변경될 수 있습니다. 이 제품에는 Indiana University Extreme! Lab에서 개발한 소프트웨어가 포함되어 있습니다. 자세한 내용을 확인하려면 <http://www.extreme.indiana.edu/>를 방문하십시오.

이 제품에는 Frank Balluffi 및 Markus Moeller(저작권 (c) 2013 Frank Balluffi and Markus Moeller. 모든 권리 보유.)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한 사항은 MIT license에 명시된 조항에 따라 변경될 수 있습니다.

<https://www.informatica.com/legal/patents.html>에서 특허를 참조하십시오.

고지 사항: Informatica LLC는 비침해, 상품성 또는 특정 목적에 따른 사용에 대한 묵시적 보증을 포함하여 이에 국한되지 않는 어떠한 종류의 명시적이거나 묵시적인 보증 없이 이 문서를 "있는 그대로" 제공합니다. Informatica LLC는 이 소프트웨어나 문서에 오류가 없음을 보장하지 않습니다. 이 소프트웨어나 설명서에 제공된 정보에는 기술적 오류나 인쇄 오류가 있을 수 있습니다. 이 소프트웨어 및 설명서의 정보는 언제든지 예고 없이 변경될 수 있습니다.

고지 사항

이 Informatica 제품(이하 "소프트웨어")에는 Progress Software Corporation(이하 "DataDirect")의 운영 회사인 DataDirect Technologies의 특정 드라이버(이하 "DataDirect Drivers")가 포함되어 있습니다. 이러한 드라이버에는 다음 조건이 적용됩니다.

1. DataDirect Drivers는 상품성, 특정 목적에의 적합성 및 비침해에 대한 묵시적 보증을 포함하여 이에 국한되지 않는 어떠한 종류의 명시적이거나 묵시적인 보증 없이 "있는 그대로" 제공됩니다.
2. DataDirect 또는 그 타사 공급자는 손해의 발생 가능성을 사전에 알고 있었는지 여부에 관계없이 ODBC 드라이버의 사용으로 발생하는 직접, 간접, 부수적, 특별, 결과적 또는 기타 손해에 대해 어떠한 경우에도 최종 사용자에게 책임을 지지 않습니다. 이러한 제한 사항은 계약 위반, 보증 불이행, 과실, 무과실 책임, 허위 진술 및 기타 불법 행위를 포함하여 이에 국한되지 않는 모든 소송 사유에 적용됩니다.

이 설명서의 정보는 예고 없이 변경될 수 있습니다. 이 문서에서 문제가 발견되는 경우 infa_documentation@informatica.com으로 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: 2024-08-07

목차

서문	9
Informatica 리소스	9
Informatica 네트워크	9
Informatica 기술 자료	9
Informatica 설명서	9
Informatica Product Availability Matrix	10
Informatica Velocity	10
Informatica Marketplace	10
Informatica 글로벌 고객 지원 센터	10
장 1: 성능 조정 개요	11
성능 조정 개요	11
장 2: 병목 현상	12
병목 현상 개요	12
스레드 통계 사용	13
스레드 통계를 기반으로 병목 현상 제거	13
예제	13
대상 병목 현상	14
대상 병목 현상 식별	14
대상 병목 현상 제거	14
소스 병목 현상	14
소스 병목 현상 식별	14
소스 병목 현상 제거	15
매핑 병목 현상	16
매핑 병목 현상 식별	16
매핑 병목 현상 제거	16
세션 병목 현상	16
세션 병목 현상 식별	16
세션 병목 현상 제거	16
시스템 병목 현상	17
시스템 병목 현상 식별	17
시스템 병목 현상 제거	18
장 3: 대상 최적화	19
플랫 파일 대상 최적화	19
인덱스 및 키 제약 조건 삭제	19
데이터베이스 검사점 간격 늘리기	20
대량 로드 사용	20
외부 로더 사용	20

교착 상태 최소화.....	21
데이터베이스 네트워크 패킷 크기 늘리기.....	21
Oracle 대상 데이터베이스 최적화.....	21

장 4: 소스 최적화..... 22

쿼리 최적화.....	22
조건부 필터 사용.....	23
데이터베이스 네트워크 패킷 크기 늘리기.....	23
Oracle 데이터베이스 소스에 연결.....	23
Teradata FastExport 사용.....	23
Sybase 또는 Microsoft SQL Server 테이블 조인을 위해 tempdb 사용.....	24

장 5: 매핑 최적화..... 25

매핑 최적화 개요.....	25
플랫 파일 소스 최적화.....	25
순차 정렬 버퍼 길이 최소화.....	25
구분자로 분리된 플랫 파일 소스 최적화.....	26
XML 및 플랫 파일 소스 최적화.....	26
단일 패스 읽기 구성.....	26
통과 매핑 최적화.....	26
필터 최적화.....	27
데이터 유형 변환 최적화.....	27
식 최적화.....	27
공통 논리 추출.....	28
집계 함수 호출 최소화.....	28
로컬 변수로 공통 식 바꾸기.....	28
문자열 연산 대신 숫자 연산 선택.....	28
Char-Char 및 Char-Varchar 비교 최적화.....	28
LOOKUP 대신 DECODE 선택.....	28
함수 대신 연산자 사용.....	28
IIF 함수 최적화.....	29
식 평가.....	29
외부 프로시저 최적화.....	30

장 6: 변환 최적화..... 31

집계 변환 최적화.....	31
단순한 열을 기준으로 그룹화.....	31
정렬된 입력 사용.....	32
중분 집계 사용.....	32
집계 전 데이터 필터링.....	32
포트 연결 제한.....	32
사용자 지정 변환 최적화.....	32

조이너 변환 최적화.....	33
조희 변환 최적화.....	33
최적의 데이터베이스 드라이버 사용.....	33
조희 테이블 캐싱.....	34
조희 조건 최적화.....	35
조희 행 필터링.....	35
조희 테이블 인덱싱.....	35
다중 조희 최적화.....	36
파이프라인 조희 변환 작성.....	36
노멀라이저 변환 최적화.....	36
시퀀스 생성기 변환 최적화.....	36
분류기 변환 최적화.....	37
메모리 할당.....	37
파티션의 작업 디렉터리.....	37
유니코드 모드.....	37
소스 한정자 변환 최적화.....	38
SQL 변환 최적화.....	38
XML 변환 최적화.....	38
변환 오류 제거.....	38

장 7: 세션 최적화..... 40

그리드.....	40
푸시다운 최적화.....	41
동시 세션 및 워크플로우.....	41
버퍼 메모리.....	41
DTM 버퍼 크기 늘리기.....	42
버퍼 블록 크기 최적화.....	42
캐시.....	42
연결된 포트 수 제한.....	43
캐시 디렉터리 위치.....	43
캐시 크기 늘리기.....	43
64비트 버전의 PowerCenter 사용.....	43
대상 기반 커밋.....	44
실시간 처리.....	44
플러시 대기 시간.....	44
소스 기반 커밋.....	44
준비 영역.....	44
로그 파일.....	44
오류 추적.....	45
세션 이후 전자 메일.....	45

장 8: 그리드 배포 최적화.....	46
그리드 배포 최적화 개요.....	46
파일 저장.....	46
높은 대역폭 공유 파일 시스템 파일.....	47
낮은 대역폭 공유 파일 시스템 파일.....	47
로컬 저장소 파일.....	47
공유 파일 시스템 사용.....	47
공유 파일 시스템 구성.....	48
CPU 및 메모리 사용량 균형 조정.....	48
PowerCenter 매핑 및 세션 구성.....	48
파일 시스템 전체에 파일 배포.....	49
파일 배포 세션 구성.....	50
시퀀스 생성기 변환 최적화.....	51
 장 9: PowerCenter 구성 요소 최적화.....	 52
PowerCenter 구성 요소 최적화 개요.....	52
PowerCenter 리포지토리 성능 최적화.....	52
리포지토리 서비스 프로세스 및 리포지토리의 위치.....	52
개체 쿼리의 조건 순서 지정.....	53
단일 노드 DB2 데이터베이스 테이블스페이스 사용.....	53
데이터베이스 스키마 최적화.....	53
리포지토리 서비스에 대한 개체 캐싱.....	54
복원력 최적화.....	54
통합 서비스 성능 최적화.....	54
원시 및 ODBC 드라이버 사용.....	55
ASCII 데이터 이동 모드에서 통합 서비스 실행.....	55
리포지토리 서비스에 대해 PowerCenter 메타데이터 캐싱.....	55
 장 10: 시스템 최적화.....	 56
시스템 최적화 개요.....	56
네트워크 속도 향상.....	57
여러 개의 CPU 사용.....	57
페이징 감소.....	57
프로세서 바인딩 사용.....	57
 장 11: 파이프라인 파티션 사용.....	 59
파이프라인 파티션 사용 개요.....	59
파티션 수 늘리기.....	59
성능이 가장 뛰어난 파티션 유형 선택.....	60
여러 개의 CPU 사용.....	60
분할을 위해 소스 데이터베이스 최적화.....	61

데이터베이스 조정.....	61
정렬된 데이터 그룹화.....	61
단일 정렬된 쿼리 최적화.....	62
분할을 위해 대상 데이터베이스 최적화.....	62
부록 A: 성능 카운터.....	63
성능 카운터 개요.....	63
Errorrows 카운터.....	63
Readfromcache 및 Writetocache 카운터.....	64
Readfromdisk 및 Writetodisk 카운터.....	64
Rowsinlookupcache 카운터.....	65
인덱스.....	66

서문

런타임 병목 현상 및 최적의 성능을 위한 조정 방법에 대해 알아보려면 *PowerCenter® 성능 조정 가이드*를 참조하십시오.

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.com을 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.com에서 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)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 네트워크를 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 네트워크에 대한 온라인 지원 리소스를 찾으려면 <https://network.informatica.com>으로 이동하고 eSupport 옵션을 선택하십시오.

제 1 장

성능 조정 개요

이 장에 포함된 항목:

- [성능 조정 개요, 11](#)

성능 조정 개요

성능 조정의 목표는 성능 병목 현상을 제거하여 세션 성능을 최적화하는 것입니다. 세션 성능을 조정하려면 세션 성능에 만족할 때까지 성능 병목 현상을 식별하고 제거한 후 다음 세션 병목 현상을 식별하는 단계를 반복해야 합니다. 세션 성능을 조정할 때 테스트 로드 옵션을 사용하여 세션을 실행할 수 있습니다.

모든 병목 현상을 조정하는 경우 세션에서 파이프라인 파티션의 수를 늘려 세션 성능을 추가로 최적화할 수 있습니다. 세션을 처리하는 중에 파티션을 추가하면 더 많은 시스템 하드웨어를 활용하여 성능을 향상시킬 수 있습니다.

성능을 향상시킬 수 있는 최적의 방법을 결정하는 것은 복잡할 수 있으므로 한 번에 하나의 변수를 변경하고 변경 전후에 세션의 시간을 기록합니다. 세션 성능이 향상되지 않으면 최초 구성으로 돌아갈 수 있습니다.

세션 성능을 향상시키려면 다음 태스크를 완료하십시오.

1. **대상을 최적화합니다.** 통합 서비스가 대상에 효율적으로 씩니다.
2. **소스를 최적화합니다.** 통합 서비스가 소스 데이터를 효율적으로 읽습니다.
3. **매핑을 최적화합니다.** 통합 서비스가 데이터를 효율적으로 변환하고 이동합니다.
4. **변환을 최적화합니다.** 통합 서비스가 매핑에서 변환을 효율적으로 처리합니다.
5. **세션을 최적화합니다.** 통합 서비스가 세션을 더 신속하게 실행합니다.
6. **그리드 배포를 최적화합니다.** 통합 서비스가 그리드에서 최적의 성능으로 실행됩니다.
7. **PowerCenter 구성 요소를 최적화합니다.** 통합 서비스 및 리포지토리 서비스가 최적화된 상태로 작동합니다.
8. **시스템을 최적화합니다.** PowerCenter 서비스 프로세스가 더 신속하게 실행됩니다.

제 2 장

병목 현상

이 장에 포함된 항목:

- [병목 현상 개요, 12](#)
- [스레드 통계 사용, 13](#)
- [대상 병목 현상, 14](#)
- [소스 병목 현상, 14](#)
- [매핑 병목 현상, 16](#)
- [세션 병목 현상, 16](#)
- [시스템 병목 현상, 17](#)

병목 현상 개요

성능 조정의 첫 번째 단계는 성능 병목 현상을 식별하는 것입니다. 성능 병목 현상은 소스 및 대상 데이터베이스, 매핑, 세션 및 시스템에서 발생할 수 있습니다. 이에 대한 전략은 성능에 만족할 때까지 성능 병목 현상을 식별하고 제거한 후 다음 성능 병목 현상을 식별하는 단계를 반복하는 것입니다.

다음과 같은 순서로 성능 병목 현상을 찾으십시오.

1. 대상
2. 소스
3. 매핑
4. 세션
5. 시스템

다음 방법을 사용하여 성능 병목 현상을 찾습니다.

- **테스트 세션을 실행합니다.** 플랫폼 파일 소스에서 읽거나 플랫폼 파일 대상에 쓰는 테스트 세션을 구성하여 소스 및 대상 병목 현상을 식별할 수 있습니다.
- **성능 세부 정보를 분석합니다.** 성능 카운터와 같은 성능 세부 정보를 분석하여 세션 성능이 저하되는 지점을 확인합니다.
- **스레드 통계를 분석합니다.** 최적의 파티션 지점 수를 결정하려면 스레드 통계를 분석합니다.
- **시스템 성능을 모니터링합니다.** 시스템 모니터링 도구를 사용하여 CPU 사용률, I/O 대기 시간 및 시스템 병목 현상 식별을 위한 페이지를 볼 수 있습니다. 또한 워크플로우 모니터를 사용하여 시스템 리소스 사용량을 볼 수도 있습니다.

스레드 통계 사용

세션 로그의 스레드 통계를 사용하여 소스, 대상 또는 변환 병목 현상을 식별할 수 있습니다. 기본적으로, 통합 서비스는 세션 처리를 위해 각각 하나씩의 판독기 스레드, 변환 스레드 및 기록기 스레드를 사용합니다. 사용률이 가장 높은 스레드가 세션의 병목 현상을 식별합니다.

세션 로그는 다음 스레드 통계를 제공합니다.

- **런타임.** 런타임. 스레드가 실행되는 시간입니다.
- **유휴 시간.** 스레드가 유휴 상태인 시간입니다. 여기에는 스레드가 응용 프로그램 내에서 다른 스레드 처리를 대기하는 시간이 포함됩니다. 유휴 시간에는 스레드가 통합 서비스에 의해 차단된 시간이 포함되지만 운영 체제에 의해 차단된 시간은 포함되지 않습니다.
- **사용 시간.** 다음 공식으로 얻어지는 스레드의 런타임 백분율입니다.
$$(\text{run time} - \text{idle time}) / \text{run time} \times 100$$

총 런타임이 짧은 경우(예: 60초 미만) 높은 사용률을 무시할 수 있습니다. 이것이 반드시 병목 현상을 나타내는 것은 아닙니다.
- **스레드 작업 시간.** 통합 서비스가 스레드에서 각 변환을 처리하는 데 걸리는 시간의 백분율입니다. 세션 로그는 변환 스레드 작업 시간에 대해 다음 정보를 표시합니다.

```
Thread work time breakdown:  
  <transformation name>: <number> percent  
  <transformation name>: <number> percent  
  <transformation name>: <number> percent
```

변환에 적은 시간이 걸린 경우 해당 시간은 세션 로그에 포함되지 않습니다. 세션이 짧은 시간 동안 실행되어 스레드에 정확한 통계가 없는 경우 세션 로그는 통계가 정확하지 않음을 보고합니다.

스레드 통계를 기반으로 병목 현상 제거

스레드 통계를 기반으로 병목 현상을 제거하려면 다음 태스크를 완료하십시오.

- 판독기 또는 기록기 스레드의 사용률이 100%라면 소스 또는 대상 포트에서 문자열 데이터 유형을 사용합니다. 문자열이 아닌 포트에는 더 많은 처리가 필요합니다.
- 변환 스레드의 사용률이 100%라면 세그먼트에서 파티션 지점을 추가합니다. 매핑에 파티션 지점을 추가하면 통합 서비스가 세션에 사용하는 변환 스레드의 수를 늘립니다. 하지만 시스템이 이미 최대 용량으로 또는 이에 근접한 상태로 실행 중이라면 스레드를 추가하지 마십시오.
- 변환 하나가 다른 변환보다 더 많은 처리 시간을 필요로 하는 경우 해당 변환에 통과 파티션 지점을 추가합니다.

예제

세션을 실행할 때 세션 로그는 다음 텍스트와 유사한 스레드 통계 및 실행 정보를 나열합니다.

```
***** RUN INFO FOR TGT LOAD ORDER GROUP [1], CONCURRENT SET [1] *****  
Thread [READER_1_1_1] created for [the read stage] of partition point [SQ_two_gig_file_32B_rows] has  
completed.  
  Total Run Time = [505.871140] secs  
  Total Idle Time = [457.038313] secs  
  Busy Percentage = [9.653215]  
Thread [TRANSF_1_1_1] created for [the transformation stage] of partition point  
[SQ_two_gig_file_32B_rows] has completed.  
  Total Run Time = [506.230461] secs  
  Total Idle Time = [1.390318] secs  
  Busy Percentage = [99.725359]  
Thread work time breakdown:  
  LKP_ADDRESS: 25.000000 percent  
  SRT_ADDRESS: 21.551724 percent  
  RTR_ZIP_CODE: 53.448276 percent
```

```
Thread [WRITER_1_*_1] created for [the write stage] of partition point [scratch_out_32B] has completed.  
Total Run Time = [507.027212] secs  
Total Idle Time = [384.632435] secs  
Busy Percentage = [24.139686]
```

이 세션 로그에서 변환 스레드의 총 실행 시간은 506초이고 사용률은 99.7%입니다. 이것은 변환 스레드가 506초 동안 유휴 상태가 아니었음을 의미합니다. 관독기 및 기록기의 사용률은 이보다 훨씬 작은 약 9.6% 및 24%입니다. 이 세션에서 매핑에 병목 현상을 유발하는 것은 변환 스레드입니다.

변환 스레드에서 어떤 변환이 병목 현상을 유발하는지 판별하려면 스레드 작업 시간 세부 내역에서 각 변환의 사용률을 확인합니다. 이 세션 로그에서 변환 RTR_ZIP_CODE의 사용률은 53%였습니다.

대상 병목 현상

가장 일반적인 성능 병목 현상은 통합 서비스가 대상 데이터베이스에 쓸 때 발생합니다. 짧은 검사점 간격, 작은 데이터베이스 패킷 크기 또는 로드가 과중한 작업 중의 문제가 대상 병목 현상을 유발할 수 있습니다.

대상 병목 현상 식별

대상 병목 현상을 식별하려면 다음 태스크를 완료하십시오.

- 플랫폼 파일 대상에 쓸 세션의 사본을 구성합니다. 세션 성능이 상당히 향상된다면 대상 병목 현상이 있는 것입니다. 세션이 이미 플랫폼 파일 대상에 쓰는 경우 대개 대상 병목 현상이 없습니다.
- 세션 로그에서 스레드 통계를 읽습니다. 통합 서비스가 변환 또는 관독기 스레드보다 기록기 스레드에서 더 많은 시간을 소모한다면 대상 병목 현상이 있는 것입니다.

대상 병목 현상 제거

대상 병목 현상을 제거하려면 다음 태스크를 완료하십시오.

- 데이터베이스 관리자가 쿼리 최적화를 통해 데이터베이스 성능을 최적화합니다.
- 데이터베이스 네트워크 패킷 크기를 최적화합니다.
- 인덱스 및 키 제약 조건을 구성합니다.

관련 항목:

- [“대상 최적화” 페이지 19](#)

소스 병목 현상

통합 서비스가 소스 데이터베이스에서 읽을 때 성능 병목 현상이 발생할 수 있습니다. 불충분한 쿼리 또는 작은 데이터베이스 네트워크 패킷 크기가 소스 병목 현상을 유발할 수 있습니다.

소스 병목 현상 식별

세션 로그에서 스레드 통계를 확인하면 소스 병목 현상이 있는지 판별할 수 있습니다. 통합 서비스가 변환 또는 기록기 스레드보다 관독기 스레드에서 더 많은 시간을 소모한다면 소스 병목 현상이 있는 것입니다.

세션이 관계형 소스에서 읽는 경우 다음 방법을 사용하여 소스 병목 현상을 확인하십시오.

- 필터 변환
- 읽기 테스트 매핑
- 데이터베이스 쿼리

세션이 플랫폼 파일 소스에서 읽는 경우 대개 소스 병목 현상이 없습니다.

필터 변환 사용

매핑에서 필터 변환을 사용하여 소스 데이터를 읽는 데 걸리는 시간을 측정할 수 있습니다.

각 소스 한정자 뒤에 필터 변환을 추가합니다. 데이터가 필터 변환을 통과하여 처리되지 않도록 필터 조건을 **false**로 설정합니다. 새 세션을 실행하는 데 걸리는 시간이 거의 동일하게 유지되면 소스 병목 현상이 있는 것입니다.

읽기 테스트 매핑 사용

소스 병목 현상 식별을 위해 읽기 테스트 매핑을 작성할 수 있습니다. 읽기 테스트 매핑은 매핑에서 변환을 제거하여 읽기 쿼리를 격리합니다.

읽기 테스트 매핑을 작성하려면 다음 단계를 완료하십시오.

1. 원래 매핑의 사본을 만듭니다.
2. 복사한 매핑에서 소스, 소스 한정자, 사용자 지정 조인 또는 쿼리만 유지합니다.
3. 모든 변환을 제거합니다.
4. 소스 한정자를 파일 대상에 연결합니다.

읽기 테스트 매핑에 대해 세션을 실행합니다. 세션 성능이 원래 세션의 성능과 비슷하다면 소스 병목 현상이 있는 것입니다.

데이터베이스 쿼리 사용

소스 병목 현상을 식별하려면 소스 데이터베이스에 대해 직접 읽기 쿼리를 실행합니다.

세션 로그에서 직접 읽기 쿼리를 복사합니다. **isql** 같은 쿼리 도구를 사용하여 소스 데이터베이스에 대해 쿼리를 실행합니다. **Windows**의 경우, 쿼리의 결과를 파일에 로드할 수 있습니다. **UNIX**의 경우, 쿼리의 결과를 **/dev/null**에 로드할 수 있습니다.

쿼리 실행 시간과 쿼리가 첫 번째 행을 반환하는 데 걸리는 시간을 측정합니다.

소스 병목 현상 제거

소스 병목 현상을 제거하려면 다음 태스크를 완료하십시오.

- 통합 서비스가 플랫폼 파일 소스에서 읽는 경우 통합 서비스가 읽는 줄당 바이트 수를 설정합니다.
- 데이터베이스 관리자가 쿼리 최적화를 통해 데이터베이스 성능을 최적화합니다.
- 데이터베이스 네트워크 패킷 크기를 최적화합니다.
- 인덱스 및 키 제약 조건을 구성합니다.
- 데이터베이스 쿼리의 두 시간 측정 간에 긴 지연이 있는 경우 최적화 프로그램 힌트를 사용할 수 있습니다.

관련 항목:

- [“소스 최적화” 페이지 22](#)

매핑 병목 현상

소스 병목 현상이나 대상 병목 현상이 없음을 확인했다면 매핑 병목 현상을 확인해야 합니다.

매핑 병목 현상 식별

매핑 병목 현상을 식별하려면 다음 태스크를 완료하십시오.

- 세션 로그에서 스레드 통계와 작업 시간 통계를 읽습니다. 통합 서비스가 기록기 또는 관독기 스레드보다 변환 스레드에서 더 많은 시간을 소모한다면 변환 병목 현상이 있는 것입니다. 통합 서비스가 하나의 변환에서 더 많은 시간을 소비한다면 해당 변환이 변환 스레드에서 병목 현상을 유발하는 것입니다.
- 성능 카운터를 분석합니다. 높은 **errorrows** 및 **rowsinlookupcache** 카운터는 매핑 병목 현상을 나타냅니다.
- 각 대상 정의 앞에 필터 변환을 추가합니다. 대상 테이블에 데이터가 로드되지 않도록 필터 조건을 **false**로 설정합니다. 새 세션을 실행하는 데 걸리는 시간이 원래 세션의 경우와 같다면 매핑 병목 현상이 있는 것입니다.

매핑 병목 현상 제거

매핑 병목 현상을 제거하려면 매핑에서 변환 설정을 최적화합니다.

관련 항목:

- [“매핑 최적화” 페이지 25](#)

세션 병목 현상

소스, 대상 또는 매핑 병목 현상이 없다면 세션 병목 현상일 수 있습니다. 작은 캐시 크기, 적은 버퍼 메모리, 짧은 커밋 간격 등으로 인해 세션 병목 현상이 발생할 수 있습니다.

세션 병목 현상 식별

세션 병목 현상을 식별하려면 성능 세부 정보를 분석합니다. 성능 세부 정보는 입력 행, 출력 행 및 오류 행 수와 같은 각 변환에 대한 정보를 표시합니다.

세션 병목 현상 제거

세션 병목 현상을 제거하려면 세션을 최적화합니다.

관련 항목:

- [“세션 최적화” 페이지 40](#)

시스템 병목 현상

소스, 대상, 매핑 및 세션을 조정한 후 시스템 병목 현상 방지를 위해 시스템을 조정합니다. 통합 서비스는 시스템 리소스를 사용하여 변환을 처리하고, 세션을 실행하고, 데이터를 읽고 씁니다. 또한 통합 서비스는 시스템 리소스를 사용하여 집계, 조이너, 조회, 분류기, XML, 순위와 같은 변환을 위한 캐시 파일을 작성합니다.

시스템 병목 현상 식별

워크플로우 모니터에서 시스템 리소스 사용량을 볼 수 있습니다. 시스템 도구를 사용하여 Windows 및 UNIX 시스템을 모니터링할 수 있습니다.

시스템 병목 현상 식별을 위해 워크플로우 모니터 사용

통합 서비스에서 태스크 프로세스를 실행 중인 경우 워크플로우 모니터의 통합 서비스 속성에서 시스템의 스왑 사용량, CPU 사용량 및 메모리 사용량을 확인할 수 있습니다. 성능 문제를 식별하려면 다음 통합 서비스 속성을 사용하십시오.

- **CPU%**. CPU 사용량 백분율에는 시스템에서 실행되고 있는 기타 외부 태스크가 포함됩니다.
- **메모리 사용량**. 메모리 사용량 백분율에는 시스템에서 실행 중인 기타 외부 태스크가 포함됩니다. 메모리 사용량이 95%에 근접하면 시스템에서 실행 중인 태스크가 워크플로우 모니터에 표시된 용량을 사용 중인지 또는 메모리 누수가 있는지 확인하십시오. 문제를 해결하려면 시스템 도구를 사용하여 세션을 실행하기 전후의 메모리 사용량을 확인한 다음 그 결과를 세션이 실행되고 있는 동안의 메모리 사용량과 비교합니다.
- **스왑 사용량**. 스왑 사용량은 가능한 메모리 누수 또는 많은 수의 동시 태스크에 따른 페이지ング의 결과입니다.

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.

속성	설명
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**는 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.

시스템 병목 현상 제거

시스템 병목 현상을 제거하려면 다음 태스크를 완료하십시오.

- CPU 사용량이 80%를 초과하는 경우 동시 실행 태스크의 수를 확인합니다. 태스크를 다른 노드에 배포하도록 그리드 사용 또는 로드 변경을 고려합니다. 로드를 줄일 수 없는 경우에는 프로세서를 추가합니다.
- 스왑이 발생하면 실제 메모리를 늘리거나 디스크에서 메모리 집약적인 응용 프로그램의 수를 줄입니다.
- 과도한 메모리 사용(스래싱)이 발생하는 경우 실제 메모리를 추가합니다.
- 시간의 백분율이 높으면 디스크에 쓰는 대신 인메모리 캐시를 사용하도록 **PowerCenter**의 캐시를 조정합니다. 캐시를 조정 중이고, 요청이 계속 대기열에 있으며, 디스크 사용률이 최소 50%라면 다른 디스크 장치를 추가하거나 더 빠른 디스크 장치로 업그레이드합니다. 세션의 각 파티션에 대해 별도의 디스크를 사용할 수도 있습니다.
- 실제 디스크 대기열 길이가 2보다 크면 다른 디스크 장치를 추가하거나 디스크 장치를 업그레이드합니다. 관독기, 기록기 및 변환 스트레드에 대해 별도의 디스크를 사용할 수도 있습니다.
- 네트워크 대역폭을 개선합니다.
- UNIX 시스템을 조정하는 경우 주요 데이터베이스 시스템의 서버를 조정합니다.
- I/O(%wio)에서 대기하는 백분율 시간이 높은 경우 활용도가 낮은 다른 디스크를 사용합니다. 예를 들어 소스 데이터, 대상 데이터, 조회, 순위 및 집계 캐시 파일이 모두 동일한 디스크에 있는 경우 파일들을 다른 디스크에 넣습니다.

관련 항목:

- [“페이징 감소” 페이지 57](#)
- [“시스템 최적화” 페이지 56](#)

제 3 장

대상 최적화

이 장에 포함된 항목:

- [플랫 파일 대상 최적화, 19](#)
- [인덱스 및 키 제약 조건 삭제, 19](#)
- [데이터베이스 검사점 간격 늘리기, 20](#)
- [대량 로드 사용, 20](#)
- [외부 로더 사용, 20](#)
- [교착 상태 최소화, 21](#)
- [데이터베이스 네트워크 패킷 크기 늘리기, 21](#)
- [Oracle 대상 데이터베이스 최적화, 21](#)

플랫 파일 대상 최적화

플랫 파일 대상에 대해 공유 저장소 디렉터리를 사용하는 경우 다른 태스크를 수행하지 않고 파일 저장 및 관리에만 사용되는 시스템에 공유 저장소 디렉터리를 배치함으로써 세션 성능을 최적화할 수 있습니다.

통합 서비스가 단일 노드에서 실행되고 있고 세션이 플랫 파일 대상에 쓰는 경우 통합 서비스 프로세스 노드에 대해 로컬인 플랫 파일 대상에 씌으로써 세션 성능을 최적화할 수 있습니다.

인덱스 및 키 제약 조건 삭제

대상 테이블에서 키 제약 조건 또는 인덱스를 정의하는 경우 해당 테이블의 로드 속도가 느려집니다. 성능을 향상시키려면 세션을 실행하기 전에 인덱스 및 키 제약 조건을 삭제합니다. 세션 완료 후에 삭제한 인덱스 및 키 제약 조건을 다시 작성할 수 있습니다.

인덱스 및 키 제약 조건을 정기적으로 삭제하고 다시 작성하기로 결정한 경우 다음 방법을 사용하여 세션을 실행할 때마다 이러한 작업을 수행할 수 있습니다.

- 사전 및 사후 로드 저장 프로시저를 사용합니다.
- 사전 세션 및 사후 세션 SQL 명령을 사용합니다.

참고: 성능을 최적화하려면 필요한 경우 제약 조건 기반의 로드만 사용합니다.

데이터베이스 검사점 간격 늘리기

데이터베이스에서 검사점이 실행될 때마다 통합 서비스가 대기해야 하므로 성능이 저하됩니다. 검사점 수를 줄여 성능을 향상시키려면 데이터베이스에서 검사점 간격을 늘리십시오.

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

대량 로드 사용

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

대량 로드를 사용하는 경우 통합 서비스가 데이터베이스 로그를 바이패스하는데, 이로 인해 성능이 향상됩니다. 하지만 데이터베이스 로그에 쓰지 않기 때문에 대상 데이터베이스에서 롤백을 수행할 수 없습니다. 결과적으로 복구를 수행할 수 없게 됩니다. 대량 로드를 사용할 경우 향상된 세션 능력의 중요성을 불완전한 세션을 복구하는 기능과 비교하여 결정해야 합니다.

Microsoft SQL Server 또는 Oracle 대상에 대량으로 로드할 때 성능을 높이려면 커밋 간격을 길게 정의하십시오. Microsoft SQL Server 및 Oracle은 각 커밋 후에 새로운 대량 로드 트랜잭션을 시작합니다. 커밋 간격을 늘리면 대량 로드 트랜잭션의 수가 줄어 성능이 향상됩니다.

관련 항목:

- [“대상 기반 커밋” 페이지 44](#)

외부 로더 사용

세션 성능을 높이려면 다음 유형의 대상 데이터베이스에 대해 외부 로더를 사용하도록 PowerCenter를 구성합니다.

- IBM DB2 EE 또는 EEE
- Oracle

다중 파티션이 포함된 파이프라인을 사용하여 Oracle 데이터베이스에 데이터를 로드하는 경우 파이프라인에 사용하는 것과 동일한 수의 파티션으로 Oracle 대상 테이블을 작성하면 성능을 높일 수 있습니다.

- Sybase IQ

Sybase IQ 데이터베이스가 UNIX 시스템의 통합 서비스 프로세스에 대해 로컬인 경우 명명된 파이프에서 대상 테이블로 직접 데이터를 로드하면 성능을 높일 수 있습니다. 그리드에서 통합 서비스를 실행하는 경우 리소스를 확인하도록 로드 균형 조정기를 구성하고, Sybase IQ 리소스를 만들고, 그리드의 모든 노드에서 리소스를 사용할 수 있도록 합니다. 그런 다음 워크플로우 관리자에서 Sybase IQ 리소스를 적용 가능한 세션에 할당합니다.

- Teradata

교착 상태 최소화

통합 서비스가 대상에 쓰기를 시도할 때 교착 상태가 발생하는 경우 교착 상태는 동일한 대상 연결 그룹에 있는 대상에만 영향을 미칩니다. 통합 서비스는 다른 대상 연결 그룹의 대상에 계속 씁니다.

교착 상태가 발생하는 경우 세션 성능이 저하될 수 있습니다. 세션 성능을 향상시키기 위해 통합 서비스가 세션의 대상에 쓰는 데 사용하는 대상 연결 그룹의 수를 늘릴 수 있습니다. 세션의 각 대상에 대해 서로 다른 대상 연결을 사용하려면 각 대상 인스턴스에 대해 서로 다른 데이터베이스 연결 이름을 사용합니다. 각 연결 이름에 대해 같은 연결 정보를 지정할 수 있습니다.

데이터베이스 네트워크 패킷 크기 늘리기

Oracle, Sybase ASE 또는 Microsoft SQL Server 대상에 쓰는 경우 네트워크 패킷 크기를 늘려 성능을 향상시킬 수 있습니다. 네트워크 패킷 크기를 늘리면 네트워크를 통해 한 번에 더 많은 데이터 패킷을 전송할 수 있습니다. 어떤 데이터베이스에 쓰는지 기준으로 네트워크 패킷 크기를 늘리십시오.

- **Oracle.** 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 및 Microsoft SQL Server.** 패킷 크기를 늘리는 방법에 대한 자세한 내용은 데이터베이스 설명서를 참조하십시오.

Sybase ASE 또는 Microsoft SQL Server의 경우, 데이터베이스 서버 패킷 크기를 반영하려면 워크플로우 관리자에서 관계형 연결 개체의 패킷 크기도 변경해야 합니다.

Oracle 대상 데이터베이스 최적화

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

Oracle 데이터베이스에 쓸 때에는 데이터베이스 개체의 저장소 절을 확인합니다. 테이블에서 큰 초기 값과 다음 값을 사용 중인지 확인합니다. 또한 데이터베이스가 테이블 및 인덱스 데이터를 가급적 서로 다른 디스크의 개별 테이블스페이스에 저장해야 합니다.

Oracle 데이터베이스에 쓸 경우 데이터베이스가 로드되는 동안 롤백 또는 실행 취소 세그먼트가 사용됩니다.

Oracle 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.

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

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

Oracle 데이터베이스 최적화에 대한 자세한 내용은 Oracle 설명서를 참조하십시오.

제 4 장

소스 최적화

이 장에 포함된 항목:

- [쿼리 최적화, 22](#)
- [조건부 필터 사용, 23](#)
- [데이터베이스 네트워크 패킷 크기 늘리기, 23](#)
- [Oracle 데이터베이스 소스에 연결, 23](#)
- [Teradata FastExport 사용, 23](#)
- [Sybase 또는 Microsoft SQL Server 테이블 조인을 위해 tempdb 사용, 24](#)

쿼리 최적화

세션이 소스 한정자 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.

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.

데이터 통합 서비스가 데이터 읽기에 사용하는 쿼리가 세션 로그에 나타납니다. 또한 소스 한정자 변환에서도 이 쿼리를 찾을 수 있습니다. 데이터베이스 관리자에게 쿼리를 분석하여 소스 테이블에 대한 최적화 프로그램 힌트와 인덱스를 생성하게 하십시오.

쿼리를 실행한 시점과 **PowerCenter**가 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 통합 서비스가 쿼리 실행과 병렬로 행을 처리할 수 있습니다.

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

조건부 필터 사용

인덱스가 누락된 경우 소스 데이터베이스에 대한 단순한 소스 필터가 성능에 악영향을 미칠 수 있습니다. 소스 한정자에서 **PowerCenter** 조건부 필터를 사용하여 성능을 향상시킬 수 있습니다.

성능 향상을 위해 **PowerCenter** 조건부 필터를 사용해야 하는지 여부는 세션에 따라 다릅니다. 예를 들어 여러 세션이 동일한 소스에서 동시에 읽는 경우 **PowerCenter** 조건부 필터가 성능을 향상시킬 수 있습니다.

하지만 일부 세션은 소스 데이터베이스에서 소스 데이터를 필터링하면 성능이 더 향상될 수 있습니다. 데이터베이스 필터와 **PowerCenter** 필터 두 가지로 세션을 테스트하여 어떤 필터가 성능을 향상시키는지 확인하십시오.

데이터베이스 네트워크 패킷 크기 늘리기

Oracle, Sybase ASE 또는 Microsoft SQL Server 소스에서 읽는 경우 네트워크 패킷 크기를 늘려 성능을 향상시킬 수 있습니다. 네트워크 패킷 크기를 늘리면 네트워크를 통해 한 번에 더 많은 데이터 패킷을 전송할 수 있습니다. 어떤 데이터베이스에서 읽는지를 기준으로 네트워크 패킷 크기를 늘리십시오.

- **Oracle.** 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 및 Microsoft SQL Server.** 패킷 크기를 늘리는 방법에 대한 자세한 내용은 데이터베이스 설명서를 참조하십시오.

Sybase ASE 또는 Microsoft SQL Server의 경우, 데이터베이스 서버 패킷 크기를 반영하려면 워크플로우 관리자에서 관계형 연결 개체의 패킷 크기도 변경해야 합니다.

Oracle 데이터베이스 소스에 연결

통합 서비스를 단일 노드에서 실행하고 있고 Oracle 인스턴스가 통합 서비스 프로세스 노드에 대해 로컬인 경우 IPC 프로토콜을 사용하여 Oracle 데이터베이스에 연결하면 성능을 최적화할 수 있습니다. listener.ora 및 tnsnames.ora에서 Oracle 데이터베이스 연결을 설정할 수 있습니다.

Teradata FastExport 사용

FastExport는 Teradata 데이터베이스에서 대량의 데이터를 빠르게 내보내기 위해 여러 Teradata 세션을 사용하는 유틸리티입니다. Teradata 소스를 빠르게 읽기 위해 FastExport를 사용하는 PowerCenter 세션을 작성할 수 있습니다. FastExport를 사용하려면 Teradata 소스 데이터베이스로 매핑을 작성합니다. 세션에서 관계형 관독기 대신 FastExport 관독기를 사용합니다. 세션에서 내보내려는 Teradata 테이블에 대해 FastExport 연결을 사용합니다.

Sybase 또는 Microsoft SQL Server 테이블 조인을 위해 tempdb 사용

Sybase 또는 Microsoft SQL Server 데이터베이스에서 대규모 테이블을 조인할 때 **tempdb**를 인메모리 데이터 베이스로 작성하여 충분한 메모리를 할당하면 성능을 향상시킬 수 있습니다. 자세한 내용은 Sybase 또는 Microsoft SQL Server 설명서를 참조하십시오.

제 5 장

매핑 최적화

이 장에 포함된 항목:

- [매핑 최적화 개요, 25](#)
- [플랫 파일 소스 최적화, 25](#)
- [단일 패스 읽기 구성, 26](#)
- [통과 매핑 최적화, 26](#)
- [필터 최적화, 27](#)
- [데이터 유형 변환 최적화, 27](#)
- [식 최적화, 27](#)
- [외부 프로시저 최적화, 30](#)

매핑 최적화 개요

매핑 수준 최적화는 구현에 시간이 걸릴 수 있지만 세션 성능을 상당히 향상시킬 수 있습니다. 대상과 소스를 최적화한 후에는 매핑 수준 최적화에 집중하십시오.

일반적으로 매핑 최적화를 위해 매핑에서 변환의 수를 줄이고 변환 간의 불필요한 링크를 삭제합니다. 최대한 많은 작업을 수행할 수 있는 최소 개수의 변환과 식으로 매핑을 구성합니다. 변환 간의 불필요한 링크를 삭제하여 이동되는 데이터 양을 최소화합니다.

플랫 파일 소스 최적화

플랫 파일 소스를 최적화하려면 다음 태스크를 완료하십시오.

- 순차 정렬 버퍼 길이를 최적화합니다.
- 구분자로 분리된 플랫 파일 소스를 최적화합니다.
- XML 및 플랫 파일 소스를 최적화합니다.

순차 정렬 버퍼 길이 최적화

세션이 플랫 파일 소스에서 읽는 경우 통합 서비스가 각 줄에서 읽는 바이트 수를 설정하면 세션 성능을 향상시킬 수 있습니다. 기본적으로 통합 서비스는 각 줄에서 1024바이트를 읽습니다. 소스 파일의 각 줄이 기본 설정보다 짧은 경우 세션 속성에서 순차 정렬 버퍼 길이를 줄일 수 있습니다.

구분자로 분리된 플랫 파일 소스 최적화

소스가 구분자로 분리된 플랫 파일인 경우 소스 파일의 데이터 열을 분리하는 구분자 문자를 지정해야 합니다. 이스케이프 문자도 지정해야 합니다. 구분자 문자 앞에 이스케이프 문자를 포함하면 통합 서비스가 구분자 문자를 정규 문자로 읽습니다. 소스 파일에 따옴표 또는 이스케이프 문자가 포함되어 있지 않으면 세션 성능이 향상될 수 있습니다.

XML 및 플랫 파일 소스 최적화

XML 파일은 보통 태그 정보로 인해 플랫 파일보다 큼니다. XML 파일의 크기는 XML 파일의 태그 수준에 따라 달라집니다. 태그가 더 많으면 파일 크기가 더 큼니다. 그 결과, 통합 서비스가 XML 소스를 읽고 캐싱하는 데 시간이 더 걸릴 수 있습니다.

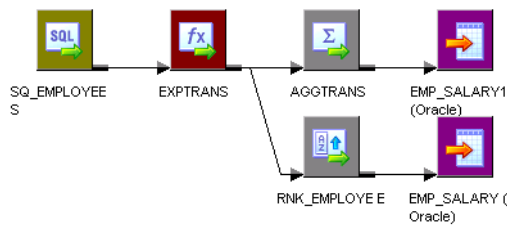
단일 패스 읽기 구성

단일 패스 읽기를 사용하면 여러 대상을 소스 한정자 하나로 채울 수 있습니다. 동일한 소스를 사용하는 세션 여러 개가 있는 경우 단일 패스 읽기를 사용합니다. 매핑 하나에서 각 매핑에 대한 변환 논리를 결합하고 각 소스에 대해 소스 한정자 하나를 사용할 수 있습니다. 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 매핑과 세션에 배치하면 통합 서비스가 동일한 소스 테이블을 두 번 읽어야 합니다. 하지만 집계와 매핑 논리를 단일 소스 한정자를 사용하여 한 매핑에 포함시키면 통합 서비스가 **Purchasing** 소스 테이블을 한 번만 읽고 해당 데이터를 별도의 파이프라인으로 보냅니다.

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.

다음 그림은 식 변환 후 매핑이 분할되는 단일 패스 읽기를 보여줍니다.



통과 매핑 최적화

통과 매핑의 성능을 최적화할 수 있습니다. 기타 변환 없이 소스에서 대상으로 직접 통과하려면 소스 한정자 변환을 대상에 직접 연결합니다. 시작 마법사를 사용하여 통과 매핑을 작성하는 경우 마법사가 소스 한정자 변환과 대상 사이에서 식 변환을 작성합니다.

필터 최적화

데이터를 필터링하려면 다음 변환 중 하나를 사용하십시오.

- **소스 한정자 변환.** 소스 한정자 변환은 관계형 소스의 열을 필터링합니다.
- **필터 변환.** 필터 변환은 매핑 내에서 데이터를 필터링합니다. 필터 변환은 모든 유형의 소스에서 행을 필터링합니다.

매핑에서 행을 필터링하면 데이터 흐름의 초기에 필터링을 통해 효율성을 향상시킬 수 있습니다. 소스 한정자 변환에서 필터를 사용하여 소스의 행을 제거합니다. 소스 한정자 변환은 관계형 소스에서 추출되는 행 집합을 제한합니다.

소스 한정자 변환에서 필터를 사용할 수 없다면 필터 변환을 사용하고 최대한 소스 한정자 변환에 가까운 위치로 해당 변환을 이동하여 데이터 흐름의 초기에 불필요한 데이터를 제거합니다. 필터 변환은 대상으로 전송되는 행 집합을 제한합니다.

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

참고: 거부된 행을 유지할 필요가 없다면 필터 변환 또는 라우터 변환을 사용하여 업데이트 전략 변환에서 거부된 행을 삭제할 수도 있습니다.

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.

시스템 성능을 향상시키려면 다음 데이터 유형 변환을 사용하십시오.

- **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 세션 성능이 향상됩니다.** 대상의 포트를 그대로 문자열로 두거나 포트를 날짜/시간 포트로 변경할 수 있습니다.

식 최적화

변환에 사용된 식을 최적화할 수도 있습니다. 가능한 경우 느린 식을 격리하고 단순화합니다.

느린 식을 격리하려면 다음 태스크를 완료하십시오.

1. 매핑에서 식을 하나씩 제거합니다.
2. 변환 없이 매핑을 실행하는 데 걸리는 시간을 확인하기 위해 매핑을 실행합니다.

세션 런타임에 상당한 차이가 있는 경우 느린 식을 최적화할 방법을 찾습니다.

공통 논리 추출

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 호출 최소화

식을 작성할 때 가능한 한 집계 함수 호출 횟수를 줄입니다. 집계 함수 호출을 사용할 때마다 통합 서비스가 데이터를 검색하고 그룹화해야 합니다. 예를 들어 다음 식에서 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 통합 서비스가 COLUMN_A와 COLUMN_B를 더한 다음 둘 모두의 합계를 구합니다.

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

로컬 변수로 공통 식 바꾸기

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 연산 대신 숫자 연산 선택

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 비교 최적화

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

LOOKUP 대신 DECODE 선택

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

함수 대신 연산자 사용

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 함수 최적화

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,  
                                ))))))))
```

이 식에는 8개의 IIF, 16개의 AND 및 최소 24개의 비교가 필요합니다.

IIF 함수를 사용한다면 식을 다음과 같이 다시 쓸 수 있습니다.

```
IIF(FLG_A='Y', VAL_A, 0.0)+ IIF(FLG_B='Y', VAL_B, 0.0)+ IIF(FLG_C='Y', VAL_C, 0.0)
```

여기에는 3개의 IIF, 2개의 비교, 2개의 추가가 있으며 결과적으로 세션이 빨라집니다.

식 평가

어떤 식이 성능을 저하시키는지 확실하지 않은 경우에는 식 성능을 평가하여 문제점을 확인합니다.

식 성능을 평가하려면 다음 단계를 수행합니다.

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

외부 프로시저 최적화

외부 프로시저가 입력 그룹에서 읽기를 대체해야 하는 경우 입력 데이터를 차단해야 합니다. 차단 기능이 없으면 프로시저 코드를 버퍼 수신 데이터에 써야 합니다. 입력 데이터를 버퍼링하는 대신 차단하면 일반적으로 세션 성능을 높일 수 있습니다.

예를 들어 두 개의 입력 그룹으로 외부 프로시저를 작성해야 한다고 가정합니다. 외부 프로시저가 첫 번째 입력 그룹에서 행을 읽은 다음 두 번째 입력 그룹에서 행을 읽습니다. 차단을 사용하는 경우 외부 프로시저 코드를 써서 다른 입력 그룹에서 데이터를 처리하는 동안 한 입력 그룹의 데이터 흐름을 차단할 수 있습니다. 데이터를 차단하기 위해 외부 프로시저 코드를 쓸 경우 프로시저는 소스 데이터를 버퍼에 복사할 필요가 없기 때문에 성능이 향상됩니다. 하지만 버퍼를 할당하고 데이터를 처리할 준비가 될 때까지 한 입력 그룹의 데이터를 버퍼에 복사하도록 외부 프로시저를 쓸 수 있습니다. 소스 데이터를 버퍼에 복사하면 성능이 저하됩니다.

제 6 장

변환 최적화

이 장에 포함된 항목:

- [집계 변환 최적화, 31](#)
- [사용자 지정 변환 최적화, 32](#)
- [조이너 변환 최적화, 33](#)
- [조희 변환 최적화, 33](#)
- [노멀라이저 변환 최적화, 36](#)
- [시퀀스 생성기 변환 최적화, 36](#)
- [분류기 변환 최적화, 37](#)
- [소스 한정자 변환 최적화, 38](#)
- [SQL 변환 최적화, 38](#)
- [XML 변환 최적화, 38](#)
- [변환 오류 제거, 38](#)

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 변환의 성능을 최적화하려면 다음 지침을 사용하십시오.

- 단순한 열을 기준으로 그룹화합니다.
- 정렬된 입력을 사용합니다.
- 충분한 집계를 사용합니다.
- 데이터를 집계하기 전에 필터링합니다.
- 포트 연결을 제한합니다.

단순한 열을 기준으로 그룹화

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

정렬된 입력 사용

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

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

정렬된 입력 옵션은 세션 중에 캐싱되는 데이터의 양을 줄여 성능을 향상시킵니다. 이 옵션을 정렬된 포트의 소스 한정자 수 옵션과 함께 사용하거나 분류기 변환을 사용하여 정렬된 데이터를 집계 변환에 전달합니다.

다중 파티션이 있는 세션에서 정렬된 입력 옵션을 사용하면 성능을 향상시킬 수 있습니다.

증분 집계 사용

스스에서 대상의 절반 미만에 영향을 주는 변경 내용을 캡처할 수 있다면 증분 집계를 사용하여 집계 변환의 성능을 최적화할 수 있습니다.

증분 집계를 사용할 때 소스에서 캡처된 변경 내용을 세션의 집계 계산에 적용하십시오. 세션을 실행할 때마다 통합 서비스가 전체 소스를 처리하고 동일한 계산을 다시 수행하는 대신 대상을 증분 방식으로 업데이트합니다.

인덱스 및 데이터 캐시 크기를 늘려 디스크 페이징 없이 모든 데이터를 메모리에 유지할 수 있습니다.

관련 항목:

- [“캐시 크기 늘리기” 페이지 43](#)

집계 전 데이터 필터링

데이터를 집계하기 전에 필터링합니다. 매핑에서 필터 변환을 사용하는 경우 집계 변환 앞에 변환을 배치하여 불필요한 집계를 줄이십시오.

포트 연결 제한

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

사용자 지정 변환 최적화

통합 서비스는 단일 행을 사용자 지정 변환 프로시저 또는 배열의 행 블록에 전달할 수 있습니다. 프로시저 코드를 써서 프로시저에서 행 하나를 받을지 아니면 행 블록 하나를 받을지 지정할 수 있습니다.

프로시저가 행 블록을 받도록 하면 성능을 향상시킬 수 있습니다.

- 통합 서비스 및 프로시저가 수행하는 함수 호출의 수를 줄일 수 있습니다. 그러면 통합 서비스가 입력 행 알림 함수를 더 적게 호출하게 되고, 프로시저가 출력 알림 함수를 더 적게 호출하게 됩니다.
- 데이터에 대한 메모리 액세스 공간의 로컬리티를 높일 수 있습니다.
- 데이터의 각 행이 아니라 데이터 블록에 대해 알고리즘을 수행하도록 프로시저 코드를 쓸 수 있습니다.

조이너 변환 최적화

조이너 변환은 런타임 시 중간 결과를 유지하기 위해 추가 공간을 필요로 하므로 성능을 저하시킬 수 있습니다. 조이너 성능 카운터 정보를 확인하여 조이너 변환에 최적화가 필요한지 여부를 결정할 수 있습니다.

조이너 변환으로 세션 성능을 향상시키려면 다음 팁을 사용하십시오.

- **마스터 소스를 중복 키 값이 더 적은 소스로 지정합니다.** 통합 서비스는 정렬된 조이너 변환을 처리할 때 한 번에 백 개의 고유 키에 대한 행을 캐싱합니다. 마스터 소스에 키 값이 같은 행이 많이 포함된 경우 통합 서비스가 더 많은 행을 캐싱해야 하므로 성능이 저하될 수 있습니다.
- **마스터 소스를 행이 더 적은 소스로 지정합니다.** 세션 중 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 세션에서 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.

데이터베이스에서 조인을 수행하려면 다음 옵션을 사용하십시오.

- 데이터베이스에서 테이블을 조인하기 위한 세션 이전 저장 프로시저를 작성합니다.
- 조인 수행을 위해 소스 한정자 변환을 사용합니다.
- **가능한 경우 정렬된 데이터를 조인하십시오.** 세션 성능을 향상시키려면 정렬된 입력을 사용하도록 조이너 변환을 구성합니다. 정렬된 데이터를 사용하도록 조이너 변환을 구성하면 통합 서비스가 디스크 입력과 출력을 최소화하여 성능을 개선합니다. 대규모 데이터 집합으로 작업할 때 가장 큰 성능 향상을 얻을 수 있습니다. 비정렬 조이너 변환의 경우 행 수가 더 적은 소스를 마스터 소스로 지정하십시오.

조희 변환 최적화

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

조희 변환을 사용하는 경우 성능을 향상시키려면 다음 태스크를 수행하십시오.

- 최적의 데이터베이스 드라이버를 사용합니다.
- 조희 테이블을 캐싱합니다.
- 조희 조건을 최적화합니다.
- 조희 열을 필터링합니다.
- 조희 테이블을 인덱싱합니다.
- 다중 조희를 최적화합니다.
- 파이프라인 조희 변환을 작성하고 조희 소스를 구축하는 파이프라인에서 파티션을 구성합니다.

최적의 데이터베이스 드라이버 사용

통합 서비스는 원시 데이터베이스 드라이버나 ODBC 드라이버를 사용하여 조희 테이블에 연결할 수 있습니다. 원시 데이터베이스 드라이버가 ODBC 드라이버보다 나은 세션 성능을 제공합니다.

조회 테이블 캐싱

매핑에 조회 변환이 포함된 경우 조회 캐싱을 활성화할 수 있습니다. 캐싱을 활성화하면 통합 서비스가 조회 테이블을 캐싱하고 세션 중에 조회 캐시를 쿼리합니다. 이 옵션을 활성화하지 않으면 통합 서비스가 행별로 조회 테이블을 쿼리합니다.

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

조회 변환의 성능을 추가로 개선하려면 다음 태스크를 완료하십시오.

- 적절한 캐시 유형을 사용합니다.
- 동시 캐시를 활성화합니다.
- 조회 조건 일치를 최적화합니다.
- 캐싱되는 행 수를 줄입니다.
- ORDER BY 문의 재정렬합니다.
- 메모리가 더 많은 시스템을 사용합니다.

관련 항목:

- [“캐시” 페이지 42](#)

캐시의 유형

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. 세션 실행 간에 조회 테이블이 바뀌지 않는 경우 이 기능을 사용합니다. 지속형 캐시를 사용하면 통합 서비스가 데이터베이스 대신 캐시 파일에서 메모리 캐시를 구축하므로 성능이 향상될 수 있습니다.

동시 캐시 활성화

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

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

조회 조건 일치 최적화

조회 변환은 조회 캐시 데이터를 조회 조건과 일치시킬 때 데이터를 정렬하고 처음 일치하는 값과 마지막 일치하는 값을 결정합니다. 이 변환이 조회 조건과 일치하는 임의 값을 반환하도록 구성할 수 있습니다. 일치하는 임의 값을 반환하도록 조회 변환을 구성하면 변환이 조회 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.

캐싱된 행 수 줄이기

캐시에 포함되는 행 수를 줄여 성능을 향상시킬 수 있습니다. 조회 SQL 재정의 옵션을 사용하여 기본 SQL 문에 WHERE 절을 추가합니다.

ORDER BY 문 재정의

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

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

예를 들어 조회 변환에서 다음과 같은 조회 조건을 사용한다고 가정합니다.

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

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

조회 SQL 재정의에 다음과 같은 조회 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 시스템 사용

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

조회 조건 최적화

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 행 필터링

필터 조건을 작성하여 조회 캐시가 구축될 때 소스에서 검색되는 조회 행 수를 줄입니다.

조회 테이블 인덱싱

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. 세션 로그에 ORDER BY 문이 포함됩니다.
- **캐싱되지 않은 조회.** 성능을 향상시키려면 조회 조건에서 열을 인덱싱합니다. 통합 서비스는 조회 변환으로 전달되는 각 행에 대해 SELECT 문을 실행합니다.

다중 조회 최적화

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.

어떤 조회 변환에서 대부분의 데이터를 처리하는지 확인하려면 각 조회 변환에 대해 `Lookup_rowsinlookupcache` 카운터를 검사합니다. 이 카운터에서 그 수가 많은 조회 변환은 조회 식 조정을 통해 이점을 얻을 수 있습니다. 그러한 식을 최적화할 수 있다면 세션 성능이 향상됩니다.

관련 항목:

- [“식 최적화” 페이지 27](#)

파이프라인 조회 변환 작성

파이프라인 조회 변환을 포함하는 매핑에는 조회 소스 및 소스 한정자를 포함하는 부분 파이프라인이 포함됩니다. 통합 서비스는 이 파이프라인에서 조회 소스 데이터를 처리합니다. 조회 변환을 포함하는 파이프라인에 조회 소스 데이터를 전달하고 캐시를 작성합니다.

부분 파이프라인은 세션 속성에 있는 별도의 개별 대상 로드 순서 그룹입니다. 성능 향상을 위해 이 파이프라인에서 다중 파티션을 구성할 수 있습니다.

노멀라이저 변환 최적화

노멀라이저 변환은 행을 생성합니다. 성능을 최적화하려면 노멀라이저 변환을 대상에 가급적 가깝게 배치합니다.

시퀀스 생성기 변환 최적화

시퀀스 생성기 변환을 최적화하려면 재사용 가능한 시퀀스 생성기를 작성하고 다중 매핑에서 이를 동시에 사용합니다. 또한 총 캐시된 값 특성을 구성합니다.

총 캐시된 값 특성은 통합 서비스가 한 번에 캐시하는 값의 수를 결정합니다. 총 캐시된 값이 너무 작지는 않은지 확인하십시오. 총 캐시된 값을 1,000보다 큰 값으로 구성합니다.

값을 캐싱할 필요가 없는 경우 총 캐시된 값을 0으로 설정합니다. 캐시를 사용하지 않는 시퀀스 생성기 변환은 캐시가 필요한 시퀀스 생성기 변환보다 빠릅니다.

시퀀스 생성기 변환에서 `CURRVAL` 포트를 연결하면 통합 서비스가 각 블록에서 1개의 행을 전달합니다. 매핑에서 `NEXTVAL` 포트만 연결하면 성능을 최적화할 수 있습니다.

관련 항목:

- [“시퀀스 생성기 변환 최적화” 페이지 51](#)

분류기 변환 최적화

분류기 변환을 최적화하려면 다음 태스크를 완료하십시오.

- 데이터 정렬을 위한 충분한 메모리를 할당합니다.
- 분류기 변환의 각 파티션에 대해 서로 다른 작업 디렉터리를 지정합니다.
- PowerCenter 통합 서비스를 ASCII 모드에서 실행합니다.

메모리 할당

최적의 성능을 얻으려면 분류기 캐시 크기를 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. 통합 서비스가 데이터를 정렬할 수 있는 충분한 메모리를 할당하지 못하는 경우 세션이 실패합니다.

수신 데이터의 양이 분류기 캐시 크기의 양보다 큰 경우 통합 서비스는 데이터를 분류기 변환 작업 디렉터리에 임시로 저장합니다. 작업 디렉터리에 데이터를 저장할 경우 통합 서비스에 적어도 수신 데이터 양의 두 배에 이르는 디스크 공간이 필요합니다. 수신 데이터의 양이 분류기 캐시 크기보다 상당히 큰 경우에는 통합 서비스에 작업 디렉터리에서 사용할 수 있는 디스크 공간 양의 두 배보다 훨씬 많은 디스크 공간이 필요할 수 있습니다.

참고: 세션 로그에는 분류기 변환에 대한 수신 데이터 크기와 입력 행 수가 포함되어 있습니다.

예를 들어 통합 서비스가 분류기 변환을 처리할 때 다음 메시지가 나타납니다.

`SORT_40422` 분류기 변환 [`srt_1_Billion_FF`](으)로부터 출력의 끝입니다. 999999999개의 행(866325637228 입력 바이트, 868929593344 임시 I/O 바이트)이 처리되었습니다.

메시지에서 입력 행의 수는 999999999이고 입력 행의 크기는 866325637228입니다.

파티션의 작업 디렉터리

통합 서비스는 데이터를 정렬할 때 임시 파일을 작성하고 이를 작업 디렉터리에 저장합니다. 통합 서비스 노드에서 작업 디렉터리로 사용할 디렉터리를 지정할 수 있습니다. 기본적으로, 통합 서비스는 `$PMTempDir` 서비스 프로세스 변수에 지정된 값을 사용합니다.

분류기 변환을 사용하여 세션을 분할할 때 파이프라인의 각 파티션에 대해 서로 다른 작업 디렉터리를 지정할 수 있습니다. 세션 성능을 높이려면 통합 서비스 노드의 물리적으로 분리된 디스크에 작업 디렉터리를 지정합니다.

유니코드 모드

분류기 변환을 최적화하려면 PowerCenter 통합 서비스를 ASCII 모드에서 실행합니다. PowerCenter 통합 서비스를 유니코드 모드에서 실행하면 분류기 변환이 추가 데이터를 디스크로 보냅니다.

소스 한정자 변환 최적화

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

SQL 변환 최적화

SQL 변환을 작성할 때 외부 SQL 쿼리 또는 변환에서 정의하는 쿼리를 사용하도록 변환을 구성합니다. 스크립트 모드에서 실행되도록 SQL 변환을 구성하는 경우 통합 서비스는 각 입력 행에 대해 외부 SQL 스크립트를 처리합니다. 쿼리 모드에서 변환을 실행하는 경우 통합 서비스는 변환에서 정의하는 SQL 쿼리를 처리합니다.

통합 서비스는 세션에서 새로운 쿼리를 처리할 때마다 `SQLPrepare`라는 함수를 호출하여 SQL 프로시저를 생성한 후 데이터베이스에 전달합니다. 입력 행마다 쿼리가 변경된다면 이것이 성능에 영향을 줄 수 있습니다.

쿼리 모드에서 변환을 실행하는 경우 성능을 향상시키려면 변환에서 정적 쿼리를 구성합니다. 쿼리 절의 데이터는 변경되지만 정적 쿼리 문은 변경되지 않습니다. 정적 쿼리를 작성하려면 SQL 편집기에서 문자열 대체 대신 매개 변수 바인딩을 사용합니다. 매개 변수 바인딩을 사용할 때 쿼리 절의 매개 변수를 변환 입력 포트의 값으로 설정합니다.

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

SQL 변환을 작성할 때 변환이 데이터베이스에 연결되는 방식을 구성합니다. 정적 연결을 선택하거나 런타임 시에 연결 정보를 변환에 전달할 수 있습니다.

정적 연결을 사용하도록 변환을 구성하는 경우 워크플로우 관리자 연결에서 연결을 선택합니다. SQL 변환은 세션 중 데이터베이스에 한 번 연결됩니다. 동적 연결 정보를 전달할 때 SQL 변환은 변환이 입력 행을 처리할 때마다 데이터베이스에 연결됩니다.

XML 변환 최적화

XML 파서 변환에서 그리고 XML 소스 정의에서 불필요한 그룹을 제거할 수 있습니다. 이러한 불필요한 그룹에 대해 메모리를 할당할 필요는 없지만 기본 키-외래 키 제약 조건은 유지해야 합니다.

변환 오류 제거

변환 오류가 통합 서비스의 성능을 저하시키는 경우가 많습니다. 변환 오류가 발생할 때마다 오류의 원인을 확인하고 오류를 발생시킨 행을 데이터 흐름에서 제거하기 위해 통합 서비스가 일시 중지됩니다. 그런 다음 통합 서비스는 보통 세션 로그 파일에 행을 씁니다.

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

오류 임계값을 설정하지 않으면 통합 서비스가 오류 행을 계속 처리하기 때문에 세션 런타임이 증가합니다. 성능을 최적화하려면 설정된 행 오류 수 이후 세션이 중지되도록 오류 임계값을 설정합니다.

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

관련 항목:

- [“오류 추적” 페이지 45](#)

제 7 장

세션 최적화

이 장에 포함된 항목:

- [그리드, 40](#)
- [푸시다운 최적화, 41](#)
- [동시 세션 및 워크플로우, 41](#)
- [버퍼 메모리, 41](#)
- [캐시, 42](#)
- [대상 기반 커밋, 44](#)
- [실시간 처리, 44](#)
- [준비 영역, 44](#)
- [로그 파일, 44](#)
- [오류 추적, 45](#)
- [세션 이후 전자 메일, 45](#)

그리드

그리드를 사용하여 세션 및 워크플로우 성능을 높일 수 있습니다. 그리드는 워크플로우 및 세션을 노드에 자동으로 배포할 수 있도록 노드 그룹에 할당된 별칭입니다.

로드 균형 조정기는 노드에 대한 오버로드 없이 노드에 태스크를 배포합니다.

그리드를 사용할 경우 통합 서비스는 워크플로우 태스크 및 세션 스레드를 여러 노드에 배포합니다. 로드 균형 조정기는 노드에 대한 오버로드 없이 노드에 태스크를 배포합니다. 그리드 노드에서 워크플로우와 세션을 실행하면 다음과 같은 성능 향상이 제공됩니다.

- 통합 서비스 작업 부하의 균형을 조정합니다.
- 동시 세션을 더 빠르게 처리합니다.
- 파티션을 더 빠르게 처리합니다.

통합 서비스에는 입력 데이터 구문 분석과 출력 데이터 형식 지정을 위한 **CPU** 리소스가 필요합니다. 그리드는 세션의 추출 및 로드 단계에서 성능 병목 현상이 있을 때 성능을 향상시킬 수 있습니다.

그리드는 메모리 또는 임시 저장소가 성능 병목 현상을 유발할 때 성능을 향상시킬 수 있습니다. **PowerCenter** 매핑에 캐시 메모리가 있는 변환이 포함되어 있을 때 각 캐시 인스턴스에 대해 적절한 메모리와 별도의 디스크 저장소를 배포하면 성능이 향상됩니다.

그리드는 세션 실행을 위한 더 많은 리소스를 제공하므로 그리드에서 세션을 실행하면 처리량이 향상될 수 있습니다. 그리드에서 한 번에 적은 수의 세션을 실행하면 성능이 향상됩니다. 동시 세션 파티션의 수가 노드의 수보다 적은 경우 그리드에서 세션을 실행하는 것은 그리드를 통해 워크플로우를 실행하는 것보다 더 효율적입니다.

그리드에서 여러 세션을 실행할 때 세션 하위 태스크는 다른 동시 세션의 하위 태스크와 노드 리소스를 공유합니다. 그리드에서 세션을 실행하려면 서로 다른 노드에서 실행 중인 프로세스 간에 조정이 필요합니다. 일부 매핑의 경우 그리드에서 세션을 실행하려면 데이터를 한 노드에서 다른 노드로 이동하기 위한 추가적인 오버헤드가 필요합니다. 각 노드에서 메모리와 CPU 리소스를 로드하는 것 외에 그리드에서 여러 세션을 실행하면 네트워크 트래픽이 추가됩니다.

그리드에서 워크플로우를 실행하면 통합 서비스가 노드에서 메모리와 CPU 리소스를 로드하며, 이때 노드 간 조정은 필요하지 않습니다.

관련 항목:

- [“그리드 배포 최적화” 페이지 46](#)

푸시다운 최적화

세션 성능을 높이려면 소스 또는 대상 데이터베이스에 변환 논리를 푸시합니다. 매핑 및 세션 구성을 기반으로, 통합 서비스는 통합 서비스 내의 변환 논리를 처리하는 대신 소스 또는 대상 데이터베이스에 대해 SQL을 실행합니다.

동시 세션 및 워크플로우

가능한 경우 성능을 향상시키려면 세션 및 워크플로우를 동시 실행합니다. 예를 들어 차원 및 팩트 테이블이 있는 분석 스키마로 데이터를 로드하는 경우 차원을 동시 로드합니다.

버퍼 메모리

통합 서비스는 세션을 초기화할 때 소스 및 대상 데이터를 유지하기 위해 메모리 블록을 할당합니다. 통합 서비스는 각 소스 및 대상 파티션에 대해 최소 두 개의 블록을 할당합니다. 많은 수의 소스와 대상을 사용하는 세션에는 추가적인 메모리 블록이 필요할 수 있습니다. 통합 서비스가 데이터를 유지할 수 있는 충분한 메모리 블록을 할당하지 못하는 경우 세션이 실패합니다.

버퍼 메모리의 양을 구성하거나 런타임 시에 버퍼 설정을 계산하도록 통합 서비스를 구성할 수 있습니다.

사용 가능한 메모리 블록의 수를 늘리려면 다음 세션 속성을 조정하십시오.

- **DTM 버퍼 크기.** 세션 속성의 속성 탭에서 DTM 버퍼 크기를 늘립니다.
- **기본 버퍼 블록 크기.** 세션 속성의 구성 개체 탭에서 버퍼 블록 크기를 줄입니다.

참고: 데이터 분할이 활성화된 경우 DTM 버퍼 크기는 전체 파티션에 할당된 모든 메모리 버퍼 풀의 총 크기입니다. n 개의 파티션이 포함되어 있는 세션의 경우, 파티션 하나가 포함된 세션 값의 n 배 이상으로 DTM 버퍼 크기를 설정합니다.

DTM 버퍼 크기 늘리기

DTM 버퍼 크기 설정은 통합 서비스가 DTM 버퍼 메모리로 사용하는 메모리의 양을 지정합니다. DTM 버퍼 메모리를 늘리면 통합 서비스가 더 많은 버퍼 블록을 작성하여 순간적인 성능 저하가 발생할 때 성능이 향상됩니다.

DTM 버퍼 메모리 할당을 늘리면 보통 처음에는 성능이 향상되고 이후에 안정화됩니다. 현저한 성능 개선이 확인되지 않는다면 DTM 버퍼 메모리 할당이 세션 성능의 요인이 아닌 경우입니다.

DTM 버퍼 크기를 늘리려면 세션 속성을 열고 속성 탭을 클릭합니다. 성능 설정에서 DTM 버퍼 크기 속성을 편집합니다. 버퍼 블록 크기의 배수로 DTM 버퍼 크기 속성을 늘립니다.

버퍼 블록 크기 최적화

시스템에 제한된 실제 메모리가 있고 세션의 매핑에 많은 수의 소스, 대상 또는 파티션이 포함되어 있는 경우 버퍼 블록 크기를 줄여야 할 수 있습니다.

매우 큰 데이터 행을 조작 중인 경우에 성능을 향상시키려면 버퍼 블록 크기를 늘려야 합니다. 대략적인 행 크기를 모르는 경우에는 다음 단계를 완료하여 행 크기를 판별하십시오.

필요한 버퍼 블록 크기를 구하려면 다음과 같이 하십시오.

1. 매핑 디자이너에서 세션의 매핑을 엽니다.
2. 대상 인스턴스를 엽니다.
3. 포트 탭을 클릭합니다.
4. 대상의 모든 열에 대한 정밀도를 추가합니다.
5. 매핑에 둘 이상의 대상이 있는 경우 각 대상의 정밀도를 계산하려면 각 추가 대상에 대해 2 - 4 단계를 반복합니다.
6. 매핑의 각 소스 정의에 대해 2 - 5 단계를 반복합니다.
7. 모든 소스 및 대상 정밀도 중 가장 큰 정밀도를 버퍼 블록 크기 계산의 총 정밀도로 선택합니다.

총 정밀도는 가장 큰 데이터 행을 이동하는 데 필요한 총 바이트 수를 나타냅니다. 예를 들어 총 정밀도가 33,000인 경우 통합 서비스는 해당 행을 이동하기 위해 버퍼 블록에서 33,000바이트가 필요합니다. 버퍼 블록 크기가 단 64,000바이트라면 통합 서비스는 한 번에 두 개 이상의 행을 이동할 수 없습니다.

버퍼 블록 크기를 설정하려면 세션 속성을 열고 구성 개체 탭을 클릭합니다. 고급 설정에서 기본 버퍼 블록 크기 속성을 편집합니다.

DTM 버퍼 메모리 할당과 마찬가지로 버퍼 블록 크기를 늘리면 성능이 향상됩니다. 성능 향상이 확인되지 않는다면 버퍼 블록 크기가 세션 성능의 요인이 아닌 경우입니다.

캐시

통합 서비스는 XML 대상과 집계, 순위, 조회 및 조이너 변환에 대해 인덱스 캐시와 데이터 캐시를 사용합니다. 통합 서비스는 변환된 데이터를 파이프라인에 반환하기 전에 데이터 캐시에 저장하고 인덱스 캐시에 그룹 정보를 저장합니다. 또한 통합 서비스는 캐시를 사용하여 분류기 변환에 대한 데이터를 저장합니다.

캐시 메모리의 양을 구성하려면 캐시 계산기를 사용하거나 캐시 크기를 지정합니다. 런타임 시에 캐시 메모리 설정을 계산하도록 통합 서비스를 구성할 수도 있습니다.

할당된 캐시가 부족하여 데이터를 저장할 수 없는 경우 통합 서비스는 해당 데이터를 세션 데이터 처리 시 임시 디스크 파일인 캐시 파일에 저장합니다. 통합 서비스가 임시 파일로 페이지징할 때마다 성능은 저하됩니다. 통합 서비스가 파일로 페이지징하는 빈도를 확인하려면 성능 카운터를 검사합니다.

캐시를 최적화하려면 다음 태스크를 수행하십시오.

- 연결된 입력/출력과 출력 전용 포트의 수를 제한합니다.
- 최적의 캐시 디렉터리 위치를 선택합니다.
- 캐시 크기를 늘립니다.
- 64비트 버전의 PowerCenter를 사용하여 큰 캐시 세션을 실행합니다.

연결된 포트 수 제한

데이터 캐시를 사용하는 변환의 경우 연결된 입력/출력 및 출력 전용 포트의 수를 제한합니다. 연결된 입력/출력 또는 출력 포트 수를 제한하면 변환이 데이터 캐시에 저장하는 데이터의 양이 줄어듭니다.

캐시 디렉터리 위치

그리드에서 통합 서비스를 실행하고 일부 통합 서비스 노드만 공유 캐시 파일 디렉터리에 빠르게 액세스할 수 있는 경우 큰 캐시가 포함된 각 세션이 디렉터리에 빠르게 액세스할 수 있는 노드에서 실행되도록 구성합니다. 세션이 디렉터리에 빠르게 액세스할 수 있는 노드에서 실행되도록 구성하려면 다음 단계를 완료하십시오.

1. PowerCenter 리소스를 작성합니다.
2. 디렉터리에 빠르게 액세스할 수 있는 노드에서 리소스를 사용할 수 있게 합니다.
3. 리소스를 세션에 할당합니다.

그리드의 모든 통합 서비스 프로세스가 캐시 파일에 느리게 액세스하는 경우 각 통합 서비스 프로세스에 대한 별도의 로컬 캐시 파일 디렉터리를 설정합니다. 캐시 디렉터리가 있는 동일한 시스템에서 통합 서비스 프로세스를 실행하는 경우 통합 서비스 프로세스가 캐시 파일에 더 빠르게 액세스할 수 있습니다.

참고: 매핑된 또는 마운트된 드라이브에서 많은 양의 데이터를 캐싱하면 성능 저하가 발생할 수 있습니다.

캐시 크기 늘리기

변환 처리에 할당되는 메모리의 양을 지정하도록 캐시 크기를 구성합니다. 구성하는 메모리의 양은 사용하려는 메모리 캐시 및 디스크 캐시의 양에 따라 다릅니다. 캐시 크기를 구성했지만 그것이 메모리에서 변환을 처리하는 데 충분하지 않은 경우 통합 서비스는 메모리에서 변환을 일부 처리하고 정보를 캐시 파일에 페이지징하여 나머지 변환을 처리합니다. 통합 서비스가 캐시 파일로 페이지징할 때마다 성능은 저하됩니다.

세션의 성능 세부 정보를 검토하여 통합 서비스가 캐시 파일로 페이지징하는 시기를 결정할 수 있습니다. 집계, 순위 또는 조이너 변환의 *Transformation_readfromdisk* 또는 *Transformation_writetodisk* 카운터는 변환 처리를 위해 통합 서비스가 디스크로 페이지징하는 횟수를 나타냅니다.

세션에 캐시를 사용하는 변환이 포함되어 있고 메모리가 충분한 시스템에서 세션을 실행하는 경우 메모리에서 변환을 처리하려면 캐시 크기를 늘립니다.

64비트 버전의 PowerCenter 사용

큰 데이터 볼륨을 처리하거나 메모리 집약적인 변환을 수행하는 경우 64비트 PowerCenter 버전을 사용하여 세션 성능을 높일 수 있습니다. 64비트 버전은 디스크 입력/출력을 크게 줄이거나 제거할 수 있는 더 많은 메모리 공간을 제공합니다.

이것은 다음과 같은 영역에서 세션 성능을 향상시킬 수 있습니다.

- **캐싱.** 32비트 플랫폼과 달리 64비트 플랫폼을 사용하면 통합 서비스가 2GB 캐시 제한에 영향을 받지 않습니다.
- **데이터 처리량.** 더 많은 메모리 공간을 사용할 수 있기 때문에 판독기, 기록기 및 DTM 스레드가 더 큰 데이터 볼륨을 처리할 수 있습니다.

대상 기반 커밋

커밋 간격 설정은 통합 서비스가 대상에 데이터를 커밋하는 지점을 결정합니다. 통합 서비스가 커밋할 때마다 성능은 저하됩니다. 따라서 커밋 간격이 짧을수록 통합 서비스가 대상 데이터베이스에 쓰는 빈도가 높아지고 전반적인 성능은 더 저하됩니다.

커밋 간격을 늘리면 통합 서비스 커밋 횟수가 감소하고 성능이 향상됩니다.

커밋 간격을 늘릴 때에는 대상 데이터베이스의 로그 파일 제한을 고려하십시오. 커밋 간격이 너무 길면 통합 서비스가 데이터베이스 로그 파일을 채워 세션이 실패할 수 있습니다.

따라서 커밋 간격을 늘릴 때의 이점과 실패한 세션을 복구하는 데 소모되는 추가 시간을 비교하여 검토하십시오.

커밋 간격을 검토 및 조정하려면 세션 속성에서 일반 옵션 설정을 클릭합니다.

실시간 처리

플러시 대기 시간

플러시 대기 시간은 통합 서비스가 소스에서 실시간 데이터를 플러시하는 빈도를 결정합니다. 플러시 대기 시간 간격을 낮게 설정할수록 통합 서비스가 대상에 메시지를 커밋하는 빈도는 높아집니다. 통합 서비스가 대상에 메시지를 커밋할 때마다 세션은 더 많은 리소스를 사용하고 처리량은 떨어집니다.

처리량을 높이려면 플러시 대기 시간을 늘립니다. 처리량은 하드웨어 및 사용 가능한 리소스에 따라 플러시 대기 시간을 특정 임계값까지 늘릴 때 증가합니다.

소스 기반 커밋

소스 기반 커밋 간격은 통합 서비스가 실시간 데이터를 대상에 커밋하는 빈도를 결정합니다. 가장 빠른 대기 시간을 얻으려면 소스 기반 커밋을 1로 설정합니다.

준비 영역

준비 영역을 사용할 때 통합 서비스는 데이터에 대한 여러 패스를 수행합니다. 가능한 경우 성능을 향상시키려면 준비 영역을 제거합니다. 통합 서비스는 단일 패스를 사용하여 여러 소스를 읽을 수 있기 때문에 준비 영역의 필요에 대한 부담을 줄일 수 있습니다.

관련 항목:

- [“단일 패스 읽기 구성” 페이지 26](#)

로그 파일

세션 및 워크플로우 로그 파일을 쓰도록 구성하지 않으면 워크플로우를 더 빨리 실행할 수 있습니다. 워크플로우 및 세션은 항상 이진 로그를 작성합니다. 로그 파일을 쓰도록 세션 또는 워크플로우를 구성하면 통합 서비스가

로그 이벤트를 두 번 씁니다. **Administrator** 도구에서 이진 로그 세션 및 워크플로우 로그에 액세스할 수 있습니다.

오류 추적

성능을 향상시키려면 통합 서비스가 세션을 실행하면서 생성하는 로그 이벤트의 수를 줄입니다. 세션에 많은 수의 변환 오류가 포함되어 있고 이러한 오류를 해결할 필요가 없다면 세션 추적 수준을 간단으로 설정합니다. 이 추적 수준에서는 데이터 서비스가 데이터 거부에 대한 오류 메시지나 행 수준 정보를 쓰지 않습니다.

매핑을 디버깅해야 하는 경우 추적 수준을 자세한 정보 표시로 설정하면 세션을 실행할 때 상당한 성능 저하가 발생할 수 있습니다. 성능을 조정할 때에는 자세한 정보 표시 추적을 사용하지 마십시오.

이 세션 추적 수준은 매핑 내의 모든 변환 관련 추적 수준을 재정의합니다. 높은 변환 오류 수준에 대한 장기간의 응답에는 사용하지 않는 것이 좋습니다.

세션 이후 전자 메일

세션 로그를 세션 이후 전자 메일에 첨부할 때 플랫폼 파일 로깅을 활성화합니다. 플랫폼 파일 로깅을 활성화하면 통합 서비스가 디스크에서 세션 로그 파일을 가져옵니다. 플랫폼 파일 로깅을 활성화하지 않으면 통합 서비스가 로그 관리자에서 로그 이벤트를 가져오고 전자 메일에 첨부할 세션 로그 파일을 생성합니다. 통합 서비스가 로그 서비스에서 세션 로그를 검색하면 워크플로우 성능이 저하되는데, 특히 세션 로그 파일이 크고 로그 서비스가 마스터 DTM이 아닌 다른 노드에서 실행될 때 성능 저하가 두드러집니다. 성능을 최적화하려면 세션 로그를 첨부할 세션 이후 전자 메일을 구성할 때 로그 파일에 쓰도록 세션을 구성합니다.

제 8 장

그리드 배포 최적화

이 장에 포함된 항목:

- [그리드 배포 최적화 개요, 46](#)
- [파일 저장, 46](#)
- [공유 파일 시스템 사용, 47](#)
- [파일 시스템 전체에 파일 배포, 49](#)
- [시퀀스 생성기 변환 최적화, 51](#)

그리드 배포 최적화 개요

그리드에서 **PowerCenter**를 실행할 때 리소스를 효율적으로 사용하고 확장성을 극대화하기 위해 그리드, 세션 및 워크플로우를 구성할 수 있습니다.

그리드에서 **PowerCenter** 성능을 개선하려면 다음 태스크를 완료하십시오.

- 그리드에 노드를 추가합니다.
- 저장소 용량과 대역폭을 늘립니다.
- 공유 파일 시스템을 사용합니다.
- 다음 태스크를 완료할 때 처리량이 높은 네트워크를 사용합니다.
 - 네트워크를 통해 소스 및 대상 액세스.
 - 그리드의 세션 옵션 사용 시 그리드의 노드 간에 데이터 전송.

파일 저장

그리드에서 실행되도록 **PowerCenter**를 구성할 때 소스 파일, 로그 파일, 캐시 파일과 같은 다양한 유형의 세션 파일에 대한 저장소 위치를 지정합니다. 성능을 향상시키려면 파일을 최적의 위치에 저장하십시오. 예를 들어 지속형 캐시 파일을 높은 대역폭 공유 파일 시스템에 저장할 수 있습니다. 저장소 요구 사항은 파일 유형에 따라 다릅니다.

파일을 다음 유형의 위치에 저장할 수 있습니다.

- **공유 파일 시스템.** 모든 통합 서비스 프로세스가 같은 파일에 액세스하도록 하려면 파일을 공유 파일 시스템에 저장합니다. 파일을 낮은 대역폭 및 높은 대역폭 공유 파일 시스템에 저장할 수 있습니다.

- **로컬.** 다른 통합 서비스 프로세스가 파일에 액세스하지 않을 때 통합 서비스 프로세스를 실행하는 로컬 시스템에 파일을 저장합니다.

높은 대역폭 공유 파일 시스템 파일

다음 파일은 세션 중에도 자주 액세스될 수 있으므로 높은 대역폭 공유 파일 시스템에 저장합니다.

- 소스 파일(조회를 위한 플랫폼 파일 포함).
- 대상 파일(분할된 세션을 위한 병합 파일 포함).
- 조회 또는 증분 집계를 위한 지속형 캐시 파일.
- 그리드의 그리드 활성화 세션 전용 비지속형 캐시 파일.

이렇게 하면 통합 서비스가 캐시를 한 번만 구축할 수 있습니다. 이러한 캐시 파일이 로컬 파일 시스템에 저장되면 통합 서비스가 각 파티션 그룹에 대해 캐시를 구축합니다.

낮은 대역폭 공유 파일 시스템 파일

다음 파일은 세션 중에 비교적 자주 액세스되지 않으므로 낮은 대역폭 공유 파일 시스템에 저장합니다.

- 매개 변수 파일 또는 기타 구성 관련 파일.
- 간접 소스 또는 대상 파일.
- 로그 파일.

로컬 저장소 파일

다음 파일은 공유 파일 시스템을 사용할 때 불필요한 파일 공유를 피할 수 있도록 로컬로 저장합니다.

- 분류기 변환 임시 파일을 포함하여 그리드에 대해 활성화되지 않는 세션용 비지속형 캐시 파일.
- 분할된 세션에 대해 순차 병합을 수행할 때 서로 다른 파티션에 대한 개별 대상 파일.
- 세션 실행의 마지막에 삭제되는 기타 임시 파일. 일반적으로 이를 설정하려면 로컬 파일 시스템에 대해 `$PmTempFileDir`을 구성합니다.

이러한 파일은 대역폭이 높은 경우에도 공유 파일 시스템에 저장하지 마십시오.

공유 파일 시스템 사용

파일 공유를 위해 다음 공유 파일 시스템을 사용할 수 있습니다.

- Windows의 CIFS(SMB) 또는 UNIX의 NFS(Network File System)와 같은 네트워크 파일 시스템. 네트워크 파일 시스템이 고성능 컴퓨팅용으로 설계된 것은 아니지만 순차적 파일 액세스에 잘 작동할 수 있습니다.
- 클러스터된 파일 시스템. 클러스터된 파일 시스템은 높은 대역폭 파일 액세스가 포함된 노드 그룹과 파일 및 디렉터리에 대한 통합 네임스페이스를 제공합니다. 클러스터된 파일 시스템 성능은 직접 연결된 로컬 파일 시스템과 유사합니다.

참고: 고가용성 옵션이 있다면 클러스터된 파일 시스템을 사용하십시오.

작은 그리드 성능에는 올바른 구성과 조정이 중요할 수 있습니다. 또한 공유 파일 시스템의 본질적인 한계를 피할 수 있도록 매핑과 세션을 구성할 수도 있습니다.

공유 파일 시스템 구성

공유 파일 시스템을 구성하려면 다음 일반 지침을 사용하십시오.

- 네트워크에 대역폭이 충분한지 확인합니다.
- 기본 저장소에 충분한 I/O 대역폭이 있는지 확인합니다.
- 파일에 신속하게 액세스할 수 있는 충분한 스레드를 갖도록 공유 파일 시스템 데몬을 구성합니다(특히 클라이언트). 예를 들어 IBM에서는 동시 액세스가 필요한 파일의 수를 예측하고 각 파일에 최소 두 개의 biod 스레드를 제공하도록 권장합니다.
플랫 파일 소스 또는 대상을 사용하는 그리드에서 동시 세션을 실행할 때 충분한 스레드를 제공하여 각 파티션이 필요한 소스 또는 대상 파일에 동시에 액세스할 수 있도록 합니다.
- 액세스 요구 사항을 기반으로 공유 파일 시스템의 마운트 지점을 구성합니다. 플랫 파일 소스 또는 대상을 사용하는 그리드에서 순차적 세션을 실행할 때 기본적인 미리 읽기(read-ahead) 또는 나중에 쓰기(write-behind) 프로세스의 효율성을 저하시킬 수 있는 구성을 피합니다. 파일 시스템은 미리 읽기 및 나중에 쓰기를 통해 순차적 파일 액세스를 최적화합니다.
- 필요한 경우 공유 파일 시스템 미리 읽기 및 나중에 쓰기 설정을 조정합니다.
- 클라이언트 및 서버 모두에 대해 공유 파일 시스템의 캐시 설정을 검토합니다. 기본 설정을 높이면 성능이 향상될 수 있습니다.
- 데이터 액세스 후 메모리 페이지 해제를 위해 파일 시스템의 나중에 해제(release-behind) 설정을 구성합니다. 그러지 않으면 큰 파일을 읽거나 쓸 때 성능이 저하될 수 있습니다.
- 액세스 패턴의 차이로 인해 소스 및 대상 그리고 지속형 캐시에 대해 서로 다른 마운트 지점을 사용할 수 있습니다.

자세한 내용은 공유 파일 시스템 설명서를 참조하십시오.

CPU 및 메모리 사용량 균형 조정

로컬 파일 시스템과 달리, 공유 파일 시스템 서버는 추가 CPU 주기를 사용하여 파일에 액세스할 수 있습니다. 계산 노드 중 하나를 나머지 노드의 공유 파일 시스템 서버로 사용하는 경우 서버가 오버로드되고 전체 그리드에 병목 현상이 발생할 수 있습니다. 공유 파일 시스템 서버가 오버로드되면 반복된 전송 및 제한 시간 요청과 함께 CPU 주기가 증가할 수 있습니다.

이 문제를 방지하려면 하나 이상의 시스템을 PowerCenter 그리드 노드에 대한 전용 공유 파일 시스템 서버로 사용합니다. 각 시스템에는 필수 태스크를 위한 충분한 저장소, CPU 및 네트워크 대역폭이 있어야 합니다.

또는 공유 파일 시스템 서버를 교차 마운트하여 파일 서버 로드를 그리드 노드에 배포할 수 있습니다. PowerCenter 매핑 및 세션이 I/O와 CPU 사용량의 고른 균형을 활용하도록 구성되어 있는 경우 공유 파일 시스템 서버를 교차 마운트하면 성능을 최적화할 수 있습니다. 그리드의 노드 수가 적고 I/O와 CPU 사용량이 균형 있게 혼합되어 있는 경우 전용 공유 파일 시스템 서버가 필요하지 않을 수 있습니다.

둘 이상의 전용 또는 교차 마운트된 공유 파일 시스템 서버를 사용하는 경우에는 공유 파일을 서버에 배포하십시오.

관련 항목:

- [“파일 시스템 전체에 파일 배포” 페이지 49](#)

PowerCenter 매핑 및 세션 구성

성능을 향상시킬 수 있는 가장 중요한 방법 중 하나는 불필요한 파일 공유를 피하는 것입니다. 올바르게 구성된 공유 파일 시스템은 소스 및 대상 파일의 순차적 액세스를 위한 적절한 성능을 제공할 수 있습니다. 하지만 지속형 캐시 파일(특히 파일 크기가 큰 경우)에 필요한 무작위 액세스는 더 문제가 될 수 있습니다.

공유 파일 시스템이 포함된 그리드에 대한 지속형 캐시 파일 구성(예: 지속형 동적 조회)에 대해서는 다음 지침을 사용하십시오.

- 가능한 경우 더 작은 크기의 지속형 캐시 파일이 메모리에 유지되도록 세션 캐시 크기를 구성합니다.
- 지속형 조회 전에 입력 행을 정렬하려면 매핑에 분류기 변환을 추가합니다. 분류기 변환에서는 로컬 파일 시스템을 사용할 수 있기 때문에 지속형 조회에서 분류기 변환으로 작업을 전환하면 성능을 향상시킬 수 있습니다.
- 통합 서비스가 캐시의 각 페이지를 읽는 횟수를 최소화하려면 동일한 조회 캐시의 페이지에 액세스해야 하는 행을 그룹화합니다.
- 입력 데이터의 크기가 큰 경우 정렬이 메모리에서 수행되도록 입력 데이터를 관리하려면 소스 기반 커밋을 사용합니다.

예를 들어 매핑 논리의 변경 없이 줄일 수 없는 **4GB**의 지속형 동적 조회와 **10GB**의 소스 데이터가 있다고 가정합니다. 우선 입력 데이터 정렬을 위해 분류기 변환을 추가하여 조회 캐시의 무작위 액세스를 줄인 다음 다음 작업을 완료합니다.

- **1GB**의 커밋 간격으로 소스 기반 커밋을 수행하도록 세션을 구성합니다.

- 분류기 변환 트랜잭션 범위를 트랜잭션으로 설정합니다.

- 소스 입력에 충분한 **1GB** 캐시 크기에 대한 분류기 변환을 구성합니다.

이 구성을 사용하면 통합 서비스가 한 번에 **1GB**의 입력 데이터를 정렬하고 캐시의 유사한 데이터에 액세스해야 하는 지속형 조회에 행을 전달합니다.

- 둘 이상의 파일 시스템을 사용할 수 있는 경우 서로 다른 파일 시스템을 사용하도록 각 파티션의 캐시 파일을 구성합니다.
- 둘 이상의 파일 시스템을 사용할 수 있는 경우 서로 다른 파일 시스템에 파일을 배포하도록 세션을 구성합니다.

파일 시스템 전체에 파일 배포

각 파일 시스템에서 독립적인 실제 디스크 하위 시스템을 사용한다고 가정하고 파일 시스템의 결합된 대역폭을 사용하도록 파일을 서로 다른 파일 시스템에 배포합니다. 파일을 서로 다른 파일 시스템에 배포하면 공유 파일 시스템 또는 다중 처리(SMP)를 사용하는 그리드에서 성능을 향상시킬 수 있습니다.

최적의 I/O 대역폭을 위해, 파일을 여러 저장소 장치에 배포하는 파일 시스템을 선택합니다. 클러스터된 파일 시스템을 사용하는 경우에는 파일을 서버 간에 배포합니다. 가능하다면 소스, 대상 및 캐시 파일을 서로 다른 저장소 장치에 배포합니다.

파일 시스템에 파일을 배포할 때에는 다음 지침을 사용하십시오.

- **소스 파일.** 통합 서비스가 다수의 파일에서 데이터를 읽을 수 있는 파일 시스템에 소스 파일을 배포하는 경우에는 큰 파일을 캐싱하기 전에 파일 시스템 미리 읽기(read-ahead) 설정을 조정합니다.
- **임시 파일.** 통합 서비스가 큰 파일에서 데이터를 읽고 임시 파일에 쓸 수 있는 파일 시스템에 임시 파일을 배포하는 경우에는 큰 파일에 대한 파일 시스템 읽기 및 쓰기 설정을 조정합니다.
- **대상 파일.** 통합 서비스가 큰 파일을 디스크에 쓸 수 있는 파일 시스템에 대상 파일을 배포하는 경우에는 큰 블록 동시 쓰기를 위해 파일 시스템을 조정합니다. 대상 파일에는 분할된 세션을 위한 병합 파일이 포함될 수 있습니다. 그리드에서 분할된 세션은 파일을 디스크에 써야 하기 때문에 최적의 잠금 성능을 위해서는 파일 시스템을 조정합니다.

파일 배포 세션 구성

세션을 수동으로 구성하여 파일 로드를 배포할 수 있습니다. 로드 변화가 심할 때 또는 새 세션이나 파일 시스템을 추가할 때(교차 마운트된 공유 파일 시스템이 포함된 그리드에 새 노드를 추가하는 것 포함)에는 세션을 편집할 필요가 있습니다.

세션을 수동으로 편집하는 대신 세션 변수를 사용하여 파일을 서로 다른 디렉터리에 배포할 수 있습니다. 이렇게 하면 필요할 때 세션 파일을 서로 다른 파일 서버로 리디렉션할 수 있습니다.

세션 변수를 사용하려면 다음 지침을 사용하십시오.

- 비즈니스 논리를 반영할 세션 파일 이름 및 디렉터리에 대한 변수 이름을 지정합니다.
- 매개 변수 파일에서 파일 로드가 사용 가능한 모든 파일 시스템에 균일하게 배포되도록 각 변수를 정의합니다. 노드 특정 변수도 정의할 수 있습니다.
- 선택적으로 매개 변수 파일을 처리하는 스크립트를 사용하여 재구성을 자동화합니다.

참고: 스크립트를 사용할 때 스크립트가 필요에 따라 세션 변수를 재정의할 수 있도록 자리 표시자를 사용하십시오.

매개 변수 파일 및 스크립트에 대한 지침

매개 변수 파일 및 스크립트를 작성할 때 다음 지침을 사용하십시오.

- 세션 파일 위치의 제어 및 유연성을 쉽게 유지하려면 스크립트를 사용하여 매개 변수 파일의 자리 표시자를 바꿉니다.
- 파일 위치를 정의할 때 예상 파일 크기 및 파일 시스템 용량을 고려합니다.
- 세션 및 워크플로우에서 비즈니스 관련 파일에 동시 액세스해야 하는 경우 비즈니스 논리에 따라 파일을 구성하지 않습니다. 예를 들어 캘리포니아 파일을 한 파일 시스템에 저장하고 뉴욕 파일을 다른 파일 시스템에 저장하는 경우 세션에서 두 파일에 동시 액세스해야 하면 병목 현상이 발생할 수 있습니다.
- 가능하다면 소스, 대상 또는 조치가 동일한 다른 파티션의 파일을 서로 다른 파일 시스템에 두십시오.

예

원시 매개 변수 파일의 다음 발췌에서, 자리 표시자 "{fs}"는 디렉터리가 위치한 파일 시스템을 나타내며 사용에 앞서 스크립트에 의해 할당되어야 합니다.

```
[SessionFFSrc_FFTgt_CA]
  $InputFile_driverInfo_CA={fs}/driverinfo_ca.dat
  $SubDir_processed_CA={fs}
# Session has Output file directory set to:
# $PmTargetFileDir/$SubDir_processed_CA
# This file is the input of SessionFFSrc_DBTgt_CA.
  $SubDir_RecordLkup_Cache_CA={fs}
# This session builds this persistent lookup cache to be used
#   by SessionFFSrc_DBTgt_CA.
# The Lookup cache directory name in the session is set to:
# $PmCacheDir/$SubDir_RecordLkup_Cache_CA
[SessionFFSrc_FFTgt_NY]
  $InputFile_driverInfo_NY={fs}/driverinfo_ny.dat
  $SubDir_processed_NY={fs}
[SessionFFSrc_DBTgt_CA]
  $SubDir_processed_CA={fs}
# session has Source file directory set to:
# $PmTargetFileDir/$SubDir_processed_CA
  $SubDir_RecordLkup_Cache_CA={fs}
# Use the persistent lookup cache built in SessionFFSrc_FFTgt_CA.
```

다음 매개 변수 파일 발췌에서, 스크립트가 자리 표시자를 `file_system_1`, `file_system_2`와 같이 적절한 파일 시스템 이름으로 바꿨습니다.

```
[SessionFFSrc_FFTgt_CA]
  $InputFile_driverInfo_CA=file_system_1/driverinfo_ca.dat
  $SubDir_processed_CA=file_system_2
# Session has Output file directory set to:
# $PmTargetFileDir/$SubDir_processed_CA
# This file is the input of SessionFFSrc_DBTgt_CA.
  $SubDir_RecordLkup_Cache_CA=file_system_1
# This session builds this persistent lookup cache to be used
# by SessionFFSrc_DBTgt_CA.
# The Lookup cache directory name in the session is set to:
# $PmCacheDir/$SubDir_RecordLkup_Cache_CA
[SessionFFSrc_FFTgt_NY]
  $InputFile_driverInfo_NY=file_system_2/driverinfo_ny.dat
  $SubDir_processed_NY=file_system_1
[SessionFFSrc_DBTgt_CA]
  $SubDir_processed_CA=file_system_1
# session has Source file directory set to:
# $PmTargetFileDir/$SubDir_processed_CA
  $SubDir_RecordLkup_Cache_CA=file_system_2
# Use the persistent lookup cache built in SessionFFSrc_FFTgt_CA.
```

시퀀스 생성기 변환 최적화

시퀀스 생성기 변환이 포함된 그리드에서 세션을 실행할 때 성능을 향상시키려면 캐시된 값의 수를 각 데이터 행의 수에 근접하게 늘립니다. 이렇게 하면 마스터 및 작업자 DTM 프로세스와 리포지토리 간의 통신이 줄어들어 빠릅니다. 마스터 및 작업자 DTM은 캐시된 각 값에 대해 한 번 통신합니다.

예를 들어 150,000개의 데이터 행과 7개의 시퀀스 생성기 변환이 있다고 가정합니다. 캐시된 값의 수는 10입니다. 마스터 및 작업자 DTM은 15,000회 통신합니다. 캐시된 값의 수를 15,000으로 늘리면 마스터 및 작업자 DTM은 10회 통신합니다.

제 9 장

PowerCenter 구성 요소 최적화

이 장에 포함된 항목:

- [PowerCenter 구성 요소 최적화 개요, 52](#)
- [PowerCenter 리포지토리 성능 최적화, 52](#)
- [통합 서비스 성능 최적화, 54](#)

PowerCenter 구성 요소 최적화 개요

다음 PowerCenter 구성 요소의 성능을 최적화할 수 있습니다.

- PowerCenter 리포지토리
- 통합 서비스

여러 시스템에서 PowerCenter를 실행하는 경우 리포지토리 서비스와 통합 서비스를 서로 다른 시스템에서 실행합니다. 대량의 데이터를 로드하려면 처리 능력이 더 높은 시스템에서 통합 서비스를 실행합니다. 또한 PowerCenter 리포지토리를 호스팅하는 시스템에서 리포지토리 서비스를 실행합니다.

PowerCenter 리포지토리 성능 최적화

PowerCenter 리포지토리 성능을 향상시키려면 다음 태스크를 완료하십시오.

- PowerCenter 리포지토리가 리포지토리 서비스 프로세스와 같은 시스템에 있는지 확인합니다.
- 개체 쿼리에서 조건의 순서를 지정합니다.
- PowerCenter 리포지토리를 DB2 데이터베이스에 설치하는 경우 PowerCenter 리포지토리에 대해 단일 노드 테이블스페이스를 사용합니다.
- PowerCenter 리포지토리를 DB2 또는 Microsoft SQL Server 데이터베이스에 설치하는 경우 PowerCenter 리포지토리에 대해 데이터베이스 스키마를 최적화합니다.

리포지토리 서비스 프로세스 및 리포지토리의 위치

고가용성 옵션 없이 구성된 리포지토리 서비스의 성능을 최적화할 수 있습니다. 성능을 최적화하려면 리포지토리 데이터베이스가 있는 시스템과 같은 시스템에서 리포지토리 서비스 프로세스가 실행되고 있는지 확인합니다.

개체 쿼리의 조건 순서 지정

리포지토리 서비스가 여러 조건이 포함된 매개 변수를 처리하는 경우 입력한 순서대로 처리됩니다. 예상하는 결과를 얻고 성능을 향상시키려면 실행하려는 순서대로 매개 변수를 입력합니다.

단일 노드 DB2 데이터베이스 테이블스페이스 사용

단일 노드 테이블스페이스에 **PowerCenter** 리포지토리를 저장할 경우 **IBM DB2 EEE** 데이터베이스에서 리포지토리 성능을 최적화할 수 있습니다. **IBM DB2 EEE** 데이터베이스를 설정할 때 데이터베이스 관리자가 단일 노드에 데이터베이스를 정의할 수 있습니다.

테이블스페이스가 하나의 노드를 포함한 경우 **PowerCenter** 클라이언트 및 통합 서비스는 리포지토리 테이블이 다른 데이터베이스 노드에 있는 것보다 더 빨리 리포지토리에 액세스합니다.

리포지토리를 작성, 복사 또는 복원할 때 테이블스페이스 이름을 지정하지 않는 경우 **DB2** 시스템은 각 리포지토리 테이블에 대해 기본 테이블스페이스를 지정합니다. **DB2** 시스템이 단일 노드 테이블스페이스를 지정할 수도 지정하지 않을 수도 있습니다.

데이터베이스 스키마 최적화

Administration Console에서 리포지토리 서비스에 대해 데이터베이스 스키마 최적화 옵션을 활성화하면 **IBM DB2** 및 **Microsoft SQL Server** 데이터베이스의 리포지토리 성능을 향상시킬 수 있습니다. 데이터베이스 스키마 최적화 옵션을 사용하면 리포지토리 서비스가 가변 길이 문자 데이터를 **CLOB** 열이 아닌 **Varchar(2000)** 열에 저장합니다. **Varchar(2000)** 열을 사용하면 다음과 같은 방식으로 리포지토리 성능이 향상됩니다.

- **디스크 액세스 감소.** **PowerCenter** 리포지토리는 **Varchar** 데이터를 데이터베이스 테이블 내의 열에 직접 저장하고, **CLOB** 데이터를 다른 테이블에 대한 참조로 저장합니다. 리포지토리에서 **CLOB** 데이터를 검색하려면 리포지토리 서비스가 하나의 데이터베이스 테이블에 액세스하여 참조를 얻은 다음 참조된 테이블에 액세스하여 데이터를 읽어야 합니다. **Varchar** 데이터를 검색하기 위해 리포지토리 서비스는 하나의 데이터베이스 테이블에 액세스합니다.
- **캐싱 향상.** 리포지토리 데이터베이스 버퍼 관리자는 **Varchar** 열을 캐싱할 수 있지만 **CLOB** 열은 캐싱할 수 없습니다.

데이터베이스 스키마 최적화는 다음 작업에 대해 리포지토리 성능을 향상시킬 수 있습니다.

- 리포지토리 백업
- 리포지토리 복원
- 리포지토리 개체 내보내기
- 개체 간 종속성 나열
- 폴더 배포

일반적으로 성능은 리포지토리 데이터베이스와 페이지 크기가 증가하면 비례적으로 향상됩니다. 따라서 데이터베이스 스키마를 최적화하면 대규모 **PowerCenter** 리포지토리에서 더 큰 성능 향상을 얻을 수 있습니다.

리포지토리 콘텐츠를 작성하거나 기존 리포지토리를 백업 및 복원할 때 데이터베이스 스키마를 최적화할 수 있습니다. 데이터베이스 스키마를 최적화하려면 리포지토리 데이터베이스가 다음 페이지 크기 요구 사항을 충족해야 합니다.

- **IBM DB2.** 데이터베이스 페이지 크기가 **4KB** 이상입니다. 최소 한 개의 임시 테이블스페이스로 페이지 크기가 **16KB** 이상입니다.
- **Microsoft SQL Server.** 데이터베이스 페이지 크기가 **8 KB** 이상입니다.

리포지토리 서비스에 대한 개체 캐싱

리포지토리 개체 캐싱은 워크플로우를 실행할 때 성능을 높입니다. 리포지토리 서비스가 메모리에서 개체를 캐싱할 때 통합 서비스는 워크플로우 실행을 완료하는 데 필요한 캐싱된 개체에 더 쉽게 액세스할 수 있습니다.

리포지토리 서비스에 대한 개체 캐싱을 관리하려면 다음 속성을 구성하십시오.

속성	설명
리포지토리 에이전트 캐싱	선택 사항입니다. 리포지토리 에이전트 캐싱을 활성화합니다. 이것은 통합 서비스가 여러 개의 세션을 반복적으로 실행할 때 성능을 향상시킵니다. 이 속성을 활성화하는 경우 리포지토리 서비스는 통합 서비스에서 요청하는 메타데이터와 리포지토리 개체를 설명하는 메타데이터를 캐싱합니다. 기본값은 Yes입니다.
에이전트 캐시 용량	선택 사항입니다. 리포지토리 에이전트 캐싱이 활성화된 경우 캐시에 포함할 수 있는 개체 수입니다. 리포지토리 서비스 프로세스가 실행되는 시스템에 사용 가능한 메모리가 있는 경우 개체 수를 늘릴 수 있습니다. 기본값은 10,000입니다.
에이전트 캐싱으로 쓰기 허용	선택 사항입니다. 리포지토리 에이전트 캐싱이 활성화된 경우 PowerCenter 클라이언트 도구를 사용하여 리포지토리에서 메타데이터를 변경할 수 있습니다. 쓰기를 활성화하는 경우 PowerCenter 클라이언트를 통해 메타데이터를 저장할 때마다 리포지토리 서비스 프로세스가 캐시를 플러시합니다. 쓰기를 비활성화하면 PowerCenter 클라이언트를 통해 메타데이터를 리포지토리에 저장할 수 없으며 캐시도 플러시되지 않습니다. 쓰기를 비활성화하는 경우 통합 서비스는 세션 및 워크플로우 메타데이터를 리포지토리에 계속 쓸 수 있습니다. 리포지토리 서비스는 통합 서비스에서 메타데이터를 쓸 때 캐시를 플러시하지 않습니다. 기본값은 Yes입니다.

복원력 최적화

리포지토리 성능을 향상시키려면 복원력과 관련된 다음 응용 프로그램 서비스 속성을 구성하십시오.

속성	설명
복원력 제한 시간 한도	선택 사항입니다. 복구를 위해 서비스에서 리소스를 유지하는 최대 시간입니다. 이 속성은 서비스에 연결하는 클라이언트에 대해 제한을 지정합니다. 한계를 초과하는 모든 복원력 제한 시간은 한도에서 잘립니다. 이 속성 값이 비어 있는 경우 도메인 수준 설정에서 값이 파생됩니다. 기본값은 비어 있음입니다.
복원력 제한 시간	선택 사항입니다. 서비스가 다른 서비스에 대한 연결을 설정하거나 다시 설정하려고 시도하는 기간입니다. 이 속성 값이 비어 있는 경우 도메인 수준 설정에서 값이 파생됩니다. 기본값은 비어 있음입니다.

통합 서비스 성능 최적화

통합 서비스 성능을 향상시키려면 다음 태스크를 완료하십시오.

- 통합 서비스에 대해 ODBC 드라이버 대신 원시 드라이버를 사용합니다.
- 문자 데이터가 7비트 ASCII 또는 EBCDIC인 경우 통합 서비스를 ASCII 데이터 이동 모드에서 실행합니다.
- 리포지토리 서비스에 대해 PowerCenter 메타데이터를 캐싱합니다.
- 고가용성과 함께 통합 서비스를 실행합니다.

참고: 고가용성과 함께 통합 서비스를 구성하면 통합 서비스가 임시 네트워크 또는 시스템 오류로 실패할 수 있는 워크플로우와 세션을 복구합니다. 워크플로우 또는 세션을 복구하기 위해 통합 서비스는 각 워크플로우 및 세션의 상태를 공유 디렉터리의 임시 파일에 기록합니다. 이것은 성능을 저하시킬 수 있습니다.

원시 및 ODBC 드라이버 사용

통합 서비스는 **ODBC** 드라이버 또는 원시 드라이버를 사용하여 데이터베이스에 연결합니다. 성능을 향상시키려면 원시 드라이버를 사용하십시오.

ASCII 데이터 이동 모드에서 통합 서비스 실행

통합 서비스에서 처리하는 모든 문자 데이터가 7비트 **ASCII** 또는 **EBCDIC**인 경우 통합 서비스가 **ASCII** 데이터 이동 모드에서 실행되도록 구성합니다. **ASCII** 모드에서 통합 서비스는 각 문자를 저장하는 데 **1**바이트를 사용합니다. 통합 서비스를 유니코드 모드에서 실행하면 각 문자에 대해 **2**바이트를 사용하므로 세션 성능이 저하될 수 있습니다.

리포지토리 서비스에 대해 PowerCenter 메타데이터 캐싱

DTM 프로세스 성능 향상을 위해 리포지토리 에이전트 캐싱을 사용할 수 있습니다. 리포지토리 에이전트 캐싱을 활성화한 경우 리포지토리 서비스는 통합 서비스에서 요청한 메타데이터를 캐싱합니다. 메타데이터를 캐싱할 때 통합 서비스는 리포지토리에서 메타데이터를 가져오는 대신 태스크의 후속 실행을 위해 캐시를 읽습니다. 통합 서비스에서 요청한 메타데이터만 캐싱됩니다.

예를 들어 **1,000**개의 세션이 포함된 워크플로우를 실행한다고 가정합니다. 캐싱을 활성화한 상태에서 처음 워크플로우를 실행할 때 통합 서비스가 리포지토리에서 세션 메타데이터를 가져옵니다. 워크플로우의 후속 실행 동안 리포지토리 서비스가 캐시에서 세션 메타데이터를 가져옵니다. 이렇게 하면 **DTM** 프로세스 성능을 높일 수 있습니다.

제 10 장

시스템 최적화

이 장에 포함된 항목:

- [시스템 최적화 개요, 56](#)
- [네트워크 속도 향상, 57](#)
- [여러 개의 CPU 사용, 57](#)
- [페이징 감소, 57](#)
- [프로세서 바인딩 사용, 57](#)

시스템 최적화 개요

세션이 비효율적인 연결이나 오버로드된 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 세션 성능을 저하시킬 수 있습니다. 시스템 관리자에게 시스템의 하드 디스크를 평가하도록 요청하십시오.

시스템 모니터링 도구를 통해 시스템에 병목 현상이 있음을 확인했다면 다음과 같은 전역 변경을 통해 모든 세션의 성능을 개선하십시오.

- **네트워크 속도를 향상시킵니다.** 네트워크 연결이 느리면 세션 성능이 저하될 수 있습니다. 시스템 관리자에게 네트워크가 최적의 속도로 실행되는지 확인하도록 요청하십시오. 통합 서비스 프로세스와 데이터베이스 사이의 네트워크 홉 수를 줄입니다.
- **여러 개의 CPU를 사용합니다.** 여러 개의 CPU를 사용하여 다중 세션과 다중 파이프라인 파티션을 병렬로 실행합니다.
- **페이징을 줄입니다.** 운영 체제에서 실제 메모리가 부족해지면 실제 메모리를 확보하기 위해 디스크로 페이징하기 시작합니다. 통합 서비스 프로세스 시스템의 디스크 페이징을 최소화할 수 있도록 실제 메모리를 구성합니다.
- **프로세서 바인딩을 사용합니다.** 다중 프로세서 UNIX 환경에서 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.

네트워크 속도 향상

통합 서비스의 성능은 네트워크 연결과 관련되어 있습니다. 로컬 디스크는 네트워크보다 5~20배 빨리 데이터를 이동할 수 있습니다. 네트워크 활동을 최소화하고 통합 서비스 성능을 향상시키려면 다음 옵션을 고려합니다.

세션에서 플랫 파일을 소스 또는 대상으로 사용하고 통합 서비스가 단일 노드에서 실행되는 경우 성능을 향상시키려면 파일을 통합 서비스와 동일한 시스템에 저장합니다. 통합 서비스와 다른 시스템에 플랫 파일을 저장하면 세션 성능이 네트워크 연결의 성능에 종속됩니다. 파일을 통합 서비스 프로세스 시스템으로 이동하고 디스크 공간을 추가하면 성능을 향상시킬 수 있습니다.

관계형 소스 또는 대상 데이터베이스를 사용하는 경우 소스/대상 데이터베이스와 통합 서비스 프로세스 사이의 네트워크 홉 수를 최소화합니다. 대상 데이터베이스를 서버 시스템으로 이동하면 통합 서비스 성능을 향상시킬 수 있습니다.

다중 파티션이 포함된 세션을 실행하는 경우 네트워크 관리자는 네트워크를 분석하고 네트워크에 모든 파티션에서 네트워크 간을 이동하는 데이터를 처리할 수 있는 충분한 대역폭이 있는지 확인해야 합니다.

여러 개의 CPU 사용

성능을 향상시키려면 더 많은 CPU를 사용하도록 시스템을 구성합니다. 여러 개의 CPU를 사용하면 시스템에서 다중 세션과 다중 파이프라인 파티션을 병렬로 실행할 수 있습니다.

하지만 추가 CPU는 디스크 병목 현상을 유발할 수 있습니다. 디스크 병목 현상을 방지하려면 디스크에 액세스하는 프로세스의 수를 최소화하십시오. 디스크에 액세스하는 프로세스에는 데이터베이스 기능 및 운영 체제 기능이 포함됩니다. 병렬 세션 또는 병렬 파이프라인 파티션도 디스크 액세스가 필요합니다.

페이징 감소

통합 서비스 프로세스 운영 체제에 특정 작업을 위한 메모리가 부족하고 메모리를 위해 로컬 디스크를 사용하는 경우 페이징이 발생합니다. 더 많은 메모리를 확보하거나 실제 메모리를 늘리면 페이징을 줄이고 페이징의 결과로 나타나는 성능 저하를 줄일 수 있습니다. 시스템 도구를 사용하여 페이징 활동을 모니터링합니다.

다음과 같은 상황이라면 시스템 메모리를 늘려야 할 수 있습니다.

- 대규모의 캐싱된 조회를 사용하는 세션을 실행합니다.
- 다수의 파티션이 포함된 세션을 실행합니다.

메모리를 확보할 수 없다면 시스템에 메모리를 추가해야 합니다.

프로세서 바인딩 사용

다중 프로세서 UNIX 환경에서 많은 수의 세션을 실행하는 경우 통합 서비스가 대량의 시스템 리소스를 사용할 수 있습니다. 그 결과 시스템의 다른 응용 프로그램에서는 충분한 시스템 리소스를 사용하지 못할 수 있습니다. 프로세서 바인딩을 사용하면 통합 서비스 프로세스 노드의 프로세서 사용량을 제어할 수 있습니다. 소스 및 대상 데이터베이스가 같은 시스템에 있는 경우에도 프로세서 바인딩을 사용하여 데이터베이스가 사용하는 리소스를 제한하십시오.

Sun Solaris 환경에서 시스템 관리자는 **psrset** 명령을 사용하여 프로세서 집합을 작성하고 관리할 수 있습니다. 그런 다음 프로세서 집합이 통합 서비스만 실행하도록 **pbind** 명령을 사용하여 통합 서비스를 프로세서 집합에 바인딩할 수 있습니다. 또한 Sun Solaris 환경에서는 **psrinfo** 명령을 통해 구성된 각 프로세서에 대한 세부 정보를 표시할 수 있고 **psradm** 명령으로 프로세서의 작동 상태를 변경할 수도 있습니다. 자세한 내용은 시스템 관리자 문서 및 Sun Solaris 문서를 참조하십시오.

AIX 환경에서 시스템 관리자는 AIX 5L의 **Workload Manager**를 사용하여 최대 리소스 수요 시 시스템 리소스를 관리합니다. **Workload Manager**는 리소스를 할당하고 CPU, 메모리 및 디스크 I/O 대역폭을 관리할 수 있습니다. 자세한 내용은 시스템 관리자 문서 및 AIX 문서를 참조하십시오.

제 11 장

파이프라인 파티션 사용

이 장에 포함된 항목:

- [파이프라인 파티션 사용 개요, 59](#)
- [분할을 위해 소스 데이터베이스 최적화, 61](#)
- [분할을 위해 대상 데이터베이스 최적화, 62](#)

파이프라인 파티션 사용 개요

단일 파티션 성능을 극대화하기 위해 응용 프로그램, 데이터베이스 및 시스템을 조정하면 시스템의 활용성이 떨어질 수 있습니다. 이때, 둘 이상의 파티션을 갖도록 세션을 구성할 수 있습니다.

파이프라인 분할을 사용하여 세션 성능을 향상시킬 수 있습니다. 파티션 또는 파티션 지점의 수를 늘리면 스레드의 수가 증가합니다. 따라서 파티션 또는 파티션 지점의 수를 늘리면 통합 서비스 노드의 로드도 증가합니다. 통합 서비스 노드의 CPU 대역폭이 충분하면 세션에서 데이터 행을 동시에 처리하여 세션 성능을 향상시킬 수 있습니다.

참고: 단일 노드 통합 서비스를 사용하고 대량의 데이터를 처리하는 세션에서 많은 수의 파티션 또는 파티션 지점을 작성하는 경우 시스템의 오버로드를 유발할 수 있습니다.

분할 옵션이 있는 경우 파티션을 수동으로 설정하려면 다음 작업을 수행하십시오.

- 파티션의 수를 늘립니다.
- 파이프라인의 특정 지점에서 성능이 가장 뛰어난 파티션 유형을 선택합니다.
- 여러 개의 CPU를 사용합니다.

파티션 수 늘리기

파이프라인에서 파티션의 수를 늘려 세션 성능을 향상시킬 수 있습니다. 파티션의 수를 늘리면 통합 서비스에서 소스에 대한 다중 연결을 작성하고 소스 데이터의 파티션을 동시에 처리할 수 있습니다.

세션에서 파일 소스를 사용할 때 하나의 스레드 또는 다중 스레드를 사용하여 소스를 읽도록 세션을 구성할 수 있습니다. 세션 성능을 높이려면 다중 스레드를 사용하여 파일 소스를 읽도록 세션을 구성합니다. 통합 서비스는 파일 소스에 대한 다중 동시 연결을 작성합니다.

세션을 작성할 때 워크플로우 관리자는 분할을 위해 매핑에서 각 파이프라인의 유효성을 검증합니다. 통합 서비스에서 분할된 데이터를 처리할 때 데이터 일관성을 유지할 수 있는 경우 파이프라인에서 다중 파티션을 지정할 수 있습니다.

세션에 파티션을 추가할 때 다음 팁을 사용하십시오.

- **한 번에 하나의 파티션을 추가합니다.** 성능을 가장 효과적으로 모니터링하려면 한 번에 하나의 파티션을 추가하고 각 파티션을 추가하기 전에 세션 설정을 참고합니다.
- **DTM 버퍼 메모리를 설정합니다.** 파티션의 수를 늘릴 때 DTM 버퍼 크기를 늘립니다. 세션에 n 개의 파티션이 포함되어 있다면 파티션 하나가 포함된 세션 값의 n 배 이상으로 DTM 버퍼 크기를 늘립니다.
- **시퀀스 생성기에 대해 캐시된 값을 설정합니다.** 세션에 n 개의 파티션이 포함되어 있다면 시퀀스 생성기 변환에 대해 "총 캐시된 값" 특성을 사용해서는 안 됩니다. 이 값을 0보다 큰 값으로 설정하는 경우 파티션 하나가 포함된 세션에 대한 원래 값의 n 배 이상으로 값을 설정합니다.
- **소스 데이터를 균일하게 분할합니다.** 동일한 행 수가 추출되도록 각 파티션을 구성합니다.
- **세션을 실행하는 동안 시스템을 모니터링합니다.** CPU 주기를 사용할 수 있는 경우 파티션을 추가하여 성능을 향상시킬 수 있습니다. 예를 들어 시스템에 20%의 유휴 시간이 있다면 CPU 주기를 사용할 수 있습니다.
- **파티션 추가 후에 시스템을 모니터링합니다.** CPU 사용률이 상승하지 않거나, I/O 대기 시간이 상승하거나, 총 데이터 변환율이 하락한다면 하드웨어 또는 소프트웨어 병목 현상이 있는 경우일 수 있습니다. I/O 대기 시간이 큰 폭으로 상승한다면 시스템에 하드웨어 병목 현상이 있는지 확인합니다. 병목 현상이 없다면 데이터 베이스 구성을 확인합니다.

관련 항목:

- [“버퍼 메모리” 페이지 41](#)

성능이 가장 뛰어난 파티션 유형 선택

파이프라인의 서로 다른 지점에서 다른 유형의 파티션을 지정하여 세션 성능을 향상시킬 수 있습니다. 세션 성능을 최적화하려면 소스 및 대상 데이터베이스에 대해 데이터베이스 분할 파티션 유형을 사용하십시오. Oracle 및 IBM DB2 소스 그리고 IBM DB2 대상에 대해 데이터베이스 분할을 사용할 수 있습니다. 소스 데이터베이스 분할을 사용할 때 통합 서비스는 데이터베이스 시스템에 테이블 파티션 정보를 쿼리하고 데이터를 세션 파티션으로 가져옵니다. 대상 데이터베이스 분할을 사용할 때 통합 서비스는 데이터를 해당 데이터베이스 파티션 노드로 로드합니다.

다중 파이프라인 파티션 및 데이터베이스 파티션을 사용할 수 있습니다. 성능을 향상시키려면 파이프라인 파티션의 수를 데이터베이스 파티션의 수와 같게 하십시오. 하위 분할된 Oracle 소스의 성능을 향상시키려면 파이프라인 파티션의 수를 데이터베이스 하위 파티션의 수와 같게 하십시오.

성능을 높이려면 파이프라인의 다음 파티션 지점에서 파티션 유형을 지정하십시오.

- **소스 한정자 변환.** 다중 플랫폼 파일에서 데이터를 동시에 읽으려면 소스 한정자 변환에서 각 플랫폼 파일에 대해 하나의 파티션을 지정합니다. 기본 파티션 유형인 통과 파티션을 사용합니다.
- **필터 변환.** 소스 파일은 크기가 다양하기 때문에 각 파티션은 서로 다른 데이터 양을 처리합니다. 필터 변환에서 파티션 지점을 설정하고 라운드 로빈 분할을 선택하여 필터 변환으로 들어가는 로드의 균형을 조정합니다.
- **분류기 변환.** 분류기 변환 및 집계 변환에서 겹치는 그룹을 제거하려면 분류기 변환에서 해시 자동 키 분할을 사용하십시오. 이렇게 하면 분류기 변환 및 집계 변환에서 행을 처리하기 전에 통합 서비스가 동일한 설명을 가진 모든 항목을 동일한 파티션으로 그룹화합니다. 집계 변환에서 기본 파티션 지점을 삭제할 수 있습니다.
- **대상.** 대상 테이블은 키 범위를 기준으로 분할되므로 대상에서 키 범위 분할을 지정하면 대상에 대한 데이터 쓰기를 최적화할 수 있습니다.

여러 개의 CPU 사용

통합 서비스는 파이프라인에 대한 읽기, 변환 및 쓰기 처리를 병렬로 수행합니다. 이 서비스 프로세스는 단일 세션 내에서 파이프라인의 여러 파티션을 처리할 수 있고, 여러 세션을 병렬로 처리할 수 있습니다.

SMP(Symmetric Multi-Processing) 플랫폼을 보유하고 있으면 여러 CPU를 사용하여 프로세스 세션 데이터 또는 데이터의 파티션을 동시에 처리할 수 있습니다. 이를 통해 진정한 병렬 처리를 달성하여 성능이 향상됩니다. 단일 프로세서 플랫폼에서는 이러한 태스크가 CPU를 공유하므로 병렬 처리는 없습니다.

통합 서비스는 여러 CPU를 사용하여 다중 파티션을 포함하는 세션을 처리할 수 있습니다. 사용되는 CPU의 수는 파티션 수, 스레드 수, 사용 가능한 CPU 수 및 매핑 처리에 필요한 리소스의 양 같은 여러 요인에 따라 달라집니다.

분할을 위해 소스 데이터베이스 최적화

파티션을 추가하여 쿼리의 속도를 높일 수 있습니다. 일반적으로, 판독기 측의 각 파티션은 처리될 데이터의 하위 집합을 나타냅니다.

분할을 위해 소스 데이터베이스를 최적화하려면 다음 태스크를 완료하십시오.

- **데이터베이스를 조정합니다.** 데이터베이스가 제대로 조정되지 않으면 파티션 작성이 세션을 더 빠르게 만들지 못할 수 있습니다.
- **병렬 쿼리를 활성화합니다.** 일부 데이터베이스에는 병렬 쿼리를 활성화하도록 설정되어야 하는 옵션이 있을 수 있습니다. 이러한 옵션에 대해서는 데이터베이스 설명서를 확인하십시오. 이러한 옵션이 꺼져 있으면 통합 서비스가 다중 파티션 SELECT 문을 순차적으로 실행합니다.
- **데이터를 여러 테이블스페이스로 분리합니다.** 각 데이터베이스는 데이터를 여러 테이블스페이스로 분리하는 옵션을 제공합니다. 데이터베이스에서 허용하는 경우 단일 파티션에서 데이터를 추출하는 쿼리를 제공하려면 PowerCenter SQL 재정의 기능을 사용합니다.
- **정렬된 데이터를 그룹화합니다.** 정렬된 조이너 변환에 대한 성능을 높이기 위해 소스 데이터를 분할하고 그룹화할 수 있습니다.
- **단일 정렬된 쿼리를 최대화합니다.**

데이터베이스 조정

데이터베이스가 제대로 조정되지 않으면 세션을 더 빠르게 만들지 못할 수 있습니다. 데이터베이스가 제대로 조정되었는지 확인하기 위해 데이터베이스를 테스트할 수 있습니다.

데이터베이스가 제대로 조정되었는지 확인하려면 다음과 같이 하십시오.

1. 하나의 파티션을 가진 파이프라인을 작성합니다.
2. 워크플로우 모니터에서 판독기 처리량을 측정합니다.
3. 파티션을 추가합니다.
4. 처리량의 배율이 선형으로 조정되는지 확인합니다.

예를 들어 세션에 두 개의 파티션이 있으면 판독기 처리량은 두 배 빨라야 합니다. 처리량의 배율이 선형으로 조정되지 않는다면 데이터베이스를 조정해야 할 수 있습니다.

정렬된 데이터 그룹화

정렬된 조이너 변환에 대한 성능을 높이기 위해 소스 데이터를 분할하고 그룹화할 수도 있습니다. 각 그룹 내에서 그룹화를 유지하고 데이터를 정렬하려면 분류기 변환 전에 파티션 지점을 배치합니다.

데이터를 그룹화하려면 동일한 키 값을 가진 행이 동일한 파티션으로 라우팅되는지 확인합니다. 데이터가 그룹화되어 파티션에 고르게 배포되는지 확인하는 가장 좋은 방법은 정렬 기점 앞에 해시 자동 키 또는 키 범위 파티션을 추가하는 것입니다.

단일 정렬된 쿼리 최적화

데이터베이스에서 단일 정렬된 쿼리를 최적화하려면 병렬화를 활성화하는 다음 조정 옵션을 고려하십시오.

- 자동 조정을 수행하는 구성 매개 변수를 확인합니다. 예를 들어, Oracle에는 `parallel_automatic_tuning`이라는 매개 변수가 있습니다.
- 내부 병렬도가 활성화되어 있는지 확인합니다. 내부 병렬도는 단일 쿼리에서 다중 스레드를 실행하는 기능입니다. 예를 들어 Oracle에서 `parallel_adaptive_multi_user`를 확인합니다. DB2에서는 `intra_parallel`을 확인합니다.
- 병렬 실행에 사용할 수 있는 병렬 프로세스의 최대 수를 확인합니다. 예를 들어 Oracle에서 `parallel_max_servers`를 확인합니다. DB2에서는 `max_agents`를 확인합니다.
- 병렬화에서 사용되는 다양한 리소스의 크기를 확인합니다. 예를 들어 Oracle에는 `large_pool_size`, `shared_pool_size`, `hash_area_size`, `parallel_execution_message_size`, `optimizer_percent_parallel`과 같은 매개 변수가 있습니다. DB2에는 `dft_fetch_size`, `fcm_num_buffers`, `sort_heap`과 같은 구성 매개 변수가 있습니다.
- 병렬도의 정도를 확인합니다. 데이터베이스 구성 매개 변수를 사용하거나 테이블 또는 쿼리의 옵션을 사용하여 이 옵션을 설정할 수 있습니다. 예를 들어 Oracle에는 `parallel_threads_per_cpu` 및 `optimizer_percent_parallel` 매개 변수가 있습니다. DB2에는 `dft_prefetch_size`, `dft_degree`, `max_query_degree`와 같은 구성 매개 변수가 있습니다.
- 데이터베이스 확장성에 영향을 줄 수 있는 옵션을 끕니다. 예를 들어 Oracle에서 보관 기록 및 기간별 통계를 비활성화합니다.

포괄적인 데이터베이스 조정 옵션 목록은 데이터베이스 설명서를 참조하십시오.

분할을 위해 대상 데이터베이스 최적화

세션에 다중 파티션이 포함되어 있는 경우 각 파티션의 처리량은 단일 파티션 세션의 처리량과 동일해야 합니다. 이러한 상관 관계가 확인되지 않는다면 데이터베이스가 행을 순차적으로 데이터베이스에 삽입하는 중일 수 있습니다.

데이터베이스가 행을 병렬로 삽입하는지 확인하려면 대상 데이터베이스에서 다음 구성 옵션을 확인하십시오.

- 데이터베이스에서 병렬 삽입을 활성화하기 위한 옵션을 설정합니다. 예를 들어 병렬 삽입을 활성화하려면 Oracle 데이터베이스의 경우 `db_writer_processes` 옵션을, DB2의 경우 `max_agents` 옵션을 설정합니다. 일부 데이터베이스의 경우 이러한 옵션이 기본적으로 활성화되어 있을 수도 있습니다.
- 대상 테이블을 분할합니다. 이를 수행할 때에는 가능한 경우 라우터 변환을 통해 각 파티션이 단일 데이터베이스 파티션에 쓰도록 합니다. 또한 파이프라인 파티션에서 I/O 경합이 발생하지 않도록 데이터베이스 파티션을 개별 디스크에 둡니다.
- 데이터베이스 확장성을 개선하기 위한 데이터베이스 옵션을 설정합니다. 예를 들어 Oracle 데이터베이스에서 보관 기록 및 기간별 통계를 비활성화하면 확장성이 개선됩니다.

부록 A

성능 카운터

이 부록에 포함된 항목:

- [성능 카운터 개요, 63](#)
- [Errorrows 카운터, 63](#)
- [Readfromcache 및 Writetocache 카운터, 64](#)
- [Readfromdisk 및 Writetodisk 카운터, 64](#)
- [Rowsinlookupcache 카운터, 65](#)

성능 카운터 개요

모든 변환에는 카운터가 있습니다. 통합 서비스는 각 변환에 대해 입력 행, 출력 행 및 오류 행의 수를 추적합니다. 일부 변환에는 성능 카운터가 있습니다. 다음 성능 카운터를 사용하여 세션 성능을 높일 수 있습니다.

- Errorrows
- Readfromcache 및 Writetocache
- Readfromdisk 및 Writetodisk
- Rowsinlookupcache

Errorrows 카운터

변환 오류는 세션 성능에 영향을 줍니다. 변환의 *Transformation_errorrows* 카운터에 많은 수의 오류 행이 있는 경우 오류를 제거하여 성능을 향상시킬 수 있습니다.

관련 항목:

- [“변환 오류 제거” 페이지 38](#)

Readfromcache 및 Writetocache 카운터

세션에 집계, 순위 또는 조이너 변환이 포함된 경우 *Transformation_readfromcache* 및 *Transformation_writetocache* 카운터를 *Transformation_readfromdisk* 및 *Transformation_writetodisk* 카운터와 함께 검사하여 통합 서비스가 디스크에 쓰거나 디스크에서 읽는 방법을 분석합니다. 세션이 실행되는 동안 세션 성능 세부 정보를 보려면 워크플로우 모니터에서 세션을 마우스 오른쪽 단추로 클릭한 다음 속성을 선택합니다. 세부 정보 대화 상자에서 속성 탭을 클릭합니다.

디스크 액세스를 분석하려면 우선 적중률 또는 실패율을 계산합니다. 적중률은 캐시에서 통합 서비스가 수행하는 읽기 또는 쓰기 작업의 수를 나타냅니다.

실패율은 디스크에서 통합 서비스가 수행하는 읽기 또는 쓰기 작업의 수를 나타냅니다.

캐시 실패율을 계산하려면 다음 공식을 사용합니다.

$$\frac{[(\# \text{ of reads from disk}) + (\# \text{ of writes to disk})]}{[(\# \text{ of reads from memory cache}) + (\# \text{ of writes to memory cache})]}$$

캐시 적중률을 계산하려면 다음 공식을 사용합니다.

$$[1 - \text{Cache Miss ratio}]$$

디스크에 쓰기 및 읽기 작업의 수를 최소화하려면 캐시 크기를 늘립니다. 최적의 캐시 적중률은 1입니다.

Readfromdisk 및 Writetodisk 카운터

세션에 집계, 순위 또는 조이너 변환이 포함된 경우 각 *Transformation_readfromdisk* 및 *Transformation_writetodisk* 카운터를 검사합니다. 세션이 실행되는 동안 세션 성능 세부 정보를 보려면 워크플로우 모니터에서 세션을 마우스 오른쪽 단추로 클릭한 다음 속성을 선택합니다. 세부 정보 대화 상자에서 속성 탭을 클릭합니다.

이러한 카운터가 0이 아닌 숫자를 표시하는 경우 캐시 크기를 늘려 세션 성능을 향상시킬 수 있습니다. 통합 서비스에서는 인덱스 캐시를 사용하여 그룹 정보를 저장하고 데이터 캐시를 사용하여 변환된 데이터를 저장하는데, 일반적으로 데이터 캐시 크기가 인덱스 캐시 크기보다 더 큼니다. 따라서 인덱스 캐시 크기와 데이터 캐시 크기가 모두 성능에 영향을 미치는 것은 하지만 데이터 캐시 크기를 인덱스 캐시 크기 이상으로 늘릴 필요가 있습니다. 하지만 처리되는 데이터의 볼륨이 사용 가능한 메모리보다 큰 경우에는 인덱스 캐시 크기를 늘려 성능을 높일 수 있습니다.

예를 들어 통합 서비스에서 인덱스 캐시를 저장하는 데 100MB를 사용하고 데이터 캐시를 저장하는 데 500MB를 사용한다고 가정합니다. 인덱스 캐시와 데이터 캐시 각각에 임의로 배포되는 200개의 액세스가 있는 경우 다음과 같은 방법으로 캐시를 구성할 수 있습니다.

- 성능을 최적화하려면 100MB를 인덱스 캐시에 할당하고 200MB를 데이터 캐시에 할당합니다. 통합 서비스가 인덱스 캐시에서 데이터의 100%를 액세스하고 데이터 캐시에서 데이터의 40%를 액세스합니다. 통합 서비스는 항상 인덱스 캐시에 액세스하고 데이터 캐시에 120회 액세스하지 않습니다. 따라서 액세스되는 데이터의 비율은 70%입니다.
- 인덱스 캐시에 50MB를 할당하고 데이터 캐시에 250MB를 할당합니다. 통합 서비스가 인덱스 캐시에서 데이터의 50%를 액세스하고 데이터 캐시에서 데이터의 50%를 액세스합니다. 통합 서비스가 인덱스 캐시와 데이터 캐시 모두에 각각 100회 액세스하지 않습니다. 따라서 액세스되는 데이터의 비율은 50%입니다.

세션에서 증분 집계를 수행하는 경우 통합 서비스는 세션 중에 로컬 디스크에서 기록 집계 데이터를 읽고 기록 데이터를 저장할 때 디스크에 씁니다. 그 결과, `Aggregator_readtodisk` 및 `Aggregator_writetodisk` 카운터는 0 이외의 숫자도 표시합니다.

하지만 통합 서비스는 세션이 종료될 때 기록 데이터를 파일에 쓰기 때문에 계속 세션 중에 카운터를 평가할 수 있습니다. 세션 실행 중에 카운터가 0이 아닌 숫자를 표시하면 캐시 크기를 조정하여 성능을 높일 수 있습니다. 하지만 메모리 할당 또는 할당 취소에는 비용이 발생하므로 통합 서비스에서 처리할 데이터 볼륨을 알고 있는 경우에는 더 많은 데이터 볼륨을 수용하기 위해 캐시 크기를 증가시키지는 마십시오.

Rowsinlookupcache 카운터

다중 조회는 세션 성능을 떨어뜨릴 수 있습니다. 세션 성능을 향상시키려면 더 큰 조회 테이블에 대한 조회 식을 조정합니다.

관련 항목:

- [“다중 조회 최적화” 페이지 36](#)

인덱스

A

ASCII 모드
성능 [55](#)

B

페이지 크기
리포지토리 데이터베이스 스키마 최적화를 위한 최소값 [53](#)
페이징
감소 [57](#)
평가
식 [29](#)
포트
연결됨, 제한 [43](#)
푸시다운 최적화
성능 [41](#)
프로세서
바인딩 [57](#)
플랫 파일
XML 파일과 비교 [26](#)
구분자로 분리된 소스 파일 [26](#)
버퍼 길이 [25](#)
소스 최적화 [25](#)
최적의 저장소 위치 [57](#)
플랫 파일 로깅
세션 이후 전자 메일 [45](#)
플러시 대기 시간
성능, 높이기 [44](#)
필터
소스 [23](#)
필터 변환
소스 병목 현상, 식별 [15](#)
필터링
데이터 [27](#)
소스 데이터 [38](#)
함수
DECODE와 LOOKUP [28](#)
연산자와 비교 [28](#)
함수 호출
사용자 지정 변환을 위한 최소화 [32](#)

C

Char 데이터 유형
후행 공백 제거 [28](#)
CPU
동시 워크플로우, 다중 [57](#)
파이프라인 분할, 다중 [60](#)

D

DB2
PowerCenter 리포지토리 성능 [53](#)
DECODE 함수
Lookup 함수에 비교 [28](#)
최적화를 위해 사용 [28](#)
DTM 버퍼
최적의 풀 크기 [42](#)

F

FastExport
Teradata 소스 [23](#)

I

IBM DB2
리포지토리 데이터베이스 스키마, 최적화 [53](#)
IIF 식
조정 [29](#)
IPC 프로토콜
Oracle 소스 [23](#)

L

LOOKUP 함수
DECODE 함수에 비교 [28](#)
최적화를 위해 최소화 [28](#)

M

Microsoft SQL Server
리포지토리 데이터베이스 스키마, 최적화 [53](#)
인메모리 데이터베이스 [24](#)

N

네트워크
속도 개선 [57](#)
조정 [57](#)
네트워크 파일 시스템
공유 파일 시스템[네트워크 파일 시스템
aaa] 참조 [47](#)
네트워크 패킷
늘리기 [21](#), [23](#)
노멀라이저 변환
조정 [36](#)
단일 패스 읽기
정의 [26](#)

- 대량 로드
 - 관계형 대상 조정 [20](#)
- 대상
 - 버퍼 메모리 할당 [41](#)
 - 병목 현상 식별 [14](#)
 - 병목 현상, 원인 [14](#)
 - 병목 현상, 제거 [14](#)
- 대상 파일
 - 최적의 저장소 [47](#)
- 데이터 유형
 - Char [28](#)
 - Varchar [28](#)
 - 변환 최적화 [27](#)
- 데이터 이동 모드
 - 최적 [55](#)
- 데이터 캐시
 - 연결된 포트 [43](#)
 - 최적의 위치 [43](#)
 - 최적의 크기 [43](#)
- 데이터 흐름
 - 모니터링 [63](#)
 - 최적화 [63](#)
- 데이터베이스
 - Oracle 대상 조정 [21](#)
 - 검사점 간격 [20](#)
 - 교착 상태 최소화 [21](#)
 - 네트워크 패킷 크기 [21](#), [23](#)
 - 단일 정렬된 쿼리 조정 [62](#)
 - 분할을 위한 대상 최적화 [62](#)
 - 분할을 위한 소스 최적화 [61](#)
 - 소스 조정 [22](#)
 - 조인 [33](#)
 - 데이터베이스 드라이버
 - 통합 서비스에 최적 [55](#)
 - 데이터베이스 쿼리
 - 소스 병목 현상, 식별 [15](#)
 - 디렉터리
 - 공유 캐시 [43](#)
 - 디스크
 - 액세스, 최소화 [57](#)
 - 런타임
 - 스레드 통계 [13](#)
 - 로그 파일
 - 최적의 저장소 [47](#)
 - 로컬 변수
 - 식 바꾸기 [28](#)
 - 리포지토리
 - 데이터베이스 스키마, 최적화 [53](#)
 - 리포지토리 서비스
 - 리포지토리 메타데이터 캐싱 [55](#)
 - 리포지토리 서비스 프로세스
 - 최적의 위치 [52](#)
 - 매개 변수 파일
 - 성능 지침 [50](#)
 - 최적의 저장소 [47](#)
 - 매핑
 - 공통 논리 추출 [28](#)
 - 단일 패스 읽기 [26](#)
 - 병목 현상, 식별 [16](#)
 - 병목 현상, 제거 [16](#)
 - 조정 [25](#)
 - 통과 매핑, 조정 [26](#)
- 메모리
 - 64비트 PowerCenter [43](#)
 - Microsoft SQL Server 데이터베이스 [24](#)
 - Sybase ASE 데이터베이스 [24](#)
 - 늘리기 [57](#)
- 메모리 (계속)
 - 버퍼 [41](#)
- 메서드
 - 데이터 필터링 [27](#)
- 문자열 연산
 - 숫자 연산과 비교 [28](#)
 - 최소화 [28](#)
- 바인딩
 - 프로세서 [57](#)
- 버퍼 길이
 - 최적의 설정 [25](#)
- 버퍼 메모리
 - 할당 [41](#)
- 버퍼 블록 크기
 - 최적 [42](#)
- 버퍼링
 - 데이터 [30](#)
- 변환
 - 데이터 유형 [27](#)
 - 오류 제거 [38](#)
 - 조정 [31](#)
 - 최적화 [63](#)
- 변환 스레드
 - 스레드 작업 시간 [13](#)
- 병목 현상
 - UNIX의 경우 [18](#)
 - Windows의 경우 [17](#)
 - 대상 [14](#)
 - 매핑 [16](#)
 - 세션 [16](#)
 - 소스 [14](#)
 - 스레드 통계 [13](#)
 - 시스템 [17](#)
 - 식별 [12](#)
 - 제거 [12](#)
- 분류기 변환
 - 메모리 할당을 사용하여 최적화 [37](#)
 - 조정 [37](#)
 - 파티션 디렉터리 최적화 [37](#)
- 비지속형 캐시
 - 파일을 위한 최적의 저장소 [47](#)
- 사용 시간
 - 스레드 통계 [13](#)
- 사용자 지정 변환
 - 데이터 블록 처리 [32](#)
 - 조정 [32](#)
 - 함수 호출 최소화 [32](#)
- 삭제
 - 인덱스 및 키 제약 조건 [19](#)
- 성능
 - 리포지토리 데이터베이스 스키마, 최적화 [53](#)
 - 실시간 세션 [44](#)
 - 조정, 개요 [11](#)
 - 플러시 대기 시간 [44](#)
- 성능 카운터
 - Rowsinlookupcache [65](#)
 - Transformation_errorrows [63](#)
 - Transformation_readfromcache [64](#)
 - Transformation_readfromdisk [64](#)
 - Transformation_writetocache [64](#)
 - Transformation_writetodisk [64](#)
 - 유형 [63](#)
- 세션
 - 그리드 [40](#)
 - 동시 [41](#)
 - 병목 현상, 식별 [16](#)
 - 병목 현상, 원인 [16](#)

세션 (계속)

- 병목 현상, 제거 [16](#)
- 조정 [40](#)
- 푸시다운 최적화 [41](#)

세션 로그 파일

- 비활성화 [44](#)

세션 이후 전자 메일

- 성능 [45](#)

소스

- 관계형, 조정 [22](#)
- 병목 현상 식별 [14](#)
- 병목 현상, 원인 [14](#)
- 병목 현상, 제거 [15](#)
- 쿼리 조정 [22](#)
- 필터 [23](#)

소스 파일

- 최적의 저장소 [47](#)
- 플랫폼 XML [26](#)

소스 한정자 변환

- 조정 [38](#)

순위 변환

- 성능 세부 정보 [64](#)

순차 병합

- 최적의 파일 저장소 [47](#)

숫자 연산

- 문자열 연산과 비교 [28](#)

스레드

- 런타임 [13](#)
- 병목 현상, 식별 [13](#)
- 사용 시간 [13](#)
- 스레드 작업 시간 [13](#)
- 유휴 시간 [13](#)

스레드 통계

- 병목 현상, 식별 [13](#)
- 병목 현상, 제거 [13](#)

시스템

- UNIX의 병목 현상, 식별 [18](#)
- Windows의 병목 현상, 식별 [17](#)
- 병목 현상, 워크플로우 모니터로 식별 [17](#)
- 병목 현상, 원인 [17](#)
- 병목 현상, 제거 [18](#)
- 조정 [56](#)

시퀀스 생성기 변환

- 그리드 성능 [51](#)
- 재사용 가능 [36](#)
- 조정 [36](#)

식

- 로컬 변수 바꾸기 [28](#)
- 조정 [27](#)
- 평가 [29](#)

실시간 세션

- 성능, 높이기 [44](#)

연산자

- 함수와 비교 [28](#)

오류

- 추적 수준 최소화 [45](#)

오류 추적

- 추적 수준[오류 추적
aaa] 참조 [45](#)

외부 로더

- 성능 [20](#)

외부 프로시저 변환

- 데이터 차단 [30](#)

워크플로우

- 동시 [41](#)

워크플로우 로그 파일

- 비활성화 [44](#)

유휴 시간

- 스레드 통계 [13](#)

인덱스

- 삭제 [19](#)

- 조회 테이블 [35](#)

인덱스 캐시

- 최적의 위치 [43](#)

- 최적의 크기 [43](#)

읽기 테스트 매핑

- 소스 병목 현상, 식별 [15](#)

임시 파일

- 최적의 저장소 [47](#)

작업

- 숫자와 문자열 [28](#)

정렬된 입력

- 집계 변환 최적화 [32](#)

제거

- 후행 공백 [28](#)

조이너 변환

- 마스터 소스 지정 [33](#)

- 성능 세부 정보 [64](#)

- 정렬된 데이터 [33](#)

- 조정 [33](#)

조인

- 데이터베이스 내부 [33](#)

조정

- PowerCenter 리포지토리 [52](#)

- SQL 변환 [38](#)

- XML 변환 [38](#)

- 관계형 소스 [22](#)

- 네트워크 [57](#)

- 노멀라이저 변환 [36](#)

- 매핑 [25](#)

- 변환 [31](#)

- 분류기 변환 [37](#)

- 사용자 지정 변환 [32](#)

- 세션 [40](#)

- 소스 한정자 변환 [38](#)

- 시스템 [56](#)

- 시퀀스 생성기 변환 [36](#)

- 식 [27](#)

- 조이너 변환 [33](#)

- 조회 변환 [33](#)

- 집계 변환 [31](#)

- 캐시 [42](#)

- 통합 서비스 [54](#)

조회 SQL 재정의 옵션

- 캐시 크기 줄이기 [35](#)

조회 변환

- ORDER BY 문을 사용하여 최적화 [35](#)

- 다중 조회 식 최적화 [36](#)

- 데이터베이스 드라이버를 사용하여 최적화 [33](#)

- 동시 캐시를 사용하여 최적화 [34](#)

- 많은 양의 메모리가 있는 시스템을 사용하여 최적화 [35](#)

- 인덱싱을 사용하여 최적화 [35](#)

- 조정 [33](#)

- 조회 조건 일치 최적화 [34](#)

- 조회 조건 최적화 [35](#)

- 최적화 [65](#)

- 캐시 감소를 사용하여 최적화 [35](#)

- 캐시를 사용하여 최적화 [34](#)

조회 조건

- 일치 [34](#)

- 최적화 [35](#)

준비 영역

- 제거 [44](#)

- 증분 집계
 - 집계 변환 최적화 [32](#)
- 지속형 캐시
 - 조회 [34](#)
- 지속형 캐시 파일
 - 구성 지침 [48](#)
 - 최적의 저장소 [47](#)
- 집계 변환
 - 성능 세부 정보 [64](#)
 - 정렬된 입력을 사용하여 최적화 [32](#)
 - 제한된 포트 연결을 사용하여 최적화 [32](#)
 - 조정 [31](#)
 - 증분 집계 [32](#)
 - 포트별 그룹으로 최적화 [31](#)
 - 필터를 사용하여 최적화 [32](#)
- 집계 함수
 - 호출 최소화 [28](#)
- 최소화
 - 집계 함수 호출 [28](#)
- 최적의 파일 저장소
 - 대상 파일 [47](#)
 - 로그 파일 [47](#)
 - 매개 변수 파일 [47](#)
 - 비지속형 캐시 파일 [47](#)
 - 소스 파일 [47](#)
 - 임시 파일 [47](#)
- 추적 수준
 - 최소화 [45](#)
- 추출
 - 매핑의 공통 논리 [28](#)
- 캐시
 - 리포지토리 메타데이터 [55](#)
 - 시퀀스 값 [36](#)
 - 조정 [42](#)
 - 최적의 위치 [43](#)
 - 최적의 크기 [43](#)
 - 캐싱된 행 줄이기 [35](#)
- 캐시 디렉터리
 - 공유 [43](#)
- 캐시 파일
 - 최적의 저장소 [47](#)
- 커밋 간격
 - 세션 성능 [44](#)
- 쿼리
 - 관계형 소스 조정 [22](#)
- 클러스터된 파일 시스템
 - 고가용성 [47](#)
 - 공유 파일 시스템[클러스터된 파일 시스템
aaa] 참조 [47](#)
- 키 제약 조건
 - 삭제 [19](#)
- 테이블스페이스
 - DB2에 대한 최적의 유형 [53](#)
- 통과 매핑
 - 조정 [26](#)
- 통합 서비스
 - 그리드 [40](#)
 - 조정 [54](#)
 - 최적의 데이터베이스 드라이버 [55](#)
 - 커밋 간격 [44](#)
- 파이프라인
 - 데이터 흐름 모니터링 [63](#)
- 파이프라인 분할
 - 다중 CPU [60](#)
 - 대상 데이터베이스 최적화 [62](#)
 - 성능 최적화 [59](#)
 - 소스 데이터베이스 조정 [61](#)

- 파이프라인 분할 (계속)
 - 소스 데이터베이스 최적화 [61](#)
 - 최적의 파티션 유형 [60](#)
 - 파티션 추가 [59](#)
- 파일 공유
 - 네트워크 파일 시스템 [47](#)
 - 클러스터 파일 시스템 [47](#)
- 파일 시스템
 - 공유, 구성 [48](#)
 - 네트워크 [47](#)
 - 클러스터 [47](#)
- 파일 저장소
 - 공유 파일 시스템 [46](#)
 - 로컬 [46](#)
 - 유형 [46](#)
- 파티션
 - 추가 [59](#)
- 파티션 유형
 - 최적 [60](#)

O

- Oracle
 - IPC 프로토콜 [23](#)
 - 대상 조정 [21](#)
 - 연결 최적화 [23](#)
 - 외부 로더 [20](#)
- ORDER BY
 - 조회 변환을 위한 최적화 [35](#)

P

- PowerCenter 리포지토리
 - DB2의 성능 [53](#)
 - 조정 [52](#)
 - 최적의 위치 [52](#)

S

- SQL 변환
 - 조정 [38](#)
- Sybase ASE
 - 인메모리 데이터베이스 [24](#)
- Sybase IQ
 - 외부 로더 [20](#)

T

- Teradata FastExport
 - 소스 성능 [23](#)

U

- UNIX
 - 병목 현상, 제거 [18](#)
 - 시스템 병목 현상 [18](#)
 - 프로세서 바인딩 [57](#)

V

- Varchar 데이터 유형
 - 후행 공백 제거 [28](#)

W

Windows

병목 현상 [17](#)

병목 현상, 제거 [18](#)

X

XML 변환

조정 [38](#)

XML 소스

버퍼 메모리 할당 [41](#)

XML 파일

플랫 파일과 비교 [26](#)

┐

개체 쿼리

조건 순서 지정 [53](#)

검사점 간격

늘리기 [20](#)

고가용성

클러스터된 파일 시스템 [47](#)

고유 항목 선택

소스 데이터 필터링 [38](#)

공백

후행, 제거 [28](#)

공유 캐시

조화 [34](#)

공유 파일 시스템

CPU, 균형 조정 [48](#)

개요 [47](#)

구성 [48](#)

낮은 대역폭 [47](#)

높은 대역폭 [47](#)

서버 로드, 배포 [48](#)

교착 상태

최소화 [21](#)

구분자로 분리된 플랫 파일

소스 [26](#)

그룹 기준 포트

집계 변환 최적화 [31](#)

그리드

노드 병목 현상 [48](#)

성능 [40, 46](#)

시퀀스 생성기 성능, 높이기 [51](#)

최적의 저장소 위치 [46](#)

그리드의 세션

시퀀스 생성기 성능, 높이기 [51](#)