Hanbit eBook

Realtime 87

처음 시작하는 임팔라

SQL로 하둡을 다루는 가장 쉬운 방법

Getting Started with Impala

존 러셀 지음 / 양원국 옮김

O'REILLY® 한빛미디어 Hanbit Media, Inc.

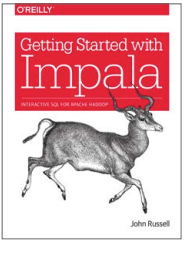
처음 시작하는 임팔라

SQL로 하둡을 다루는 가장 쉬운 방법

Getting Started with Impala

존 러셀 지음 / **양원국** 옮김

O'REILLY® 한빛미디어 Hanbit Media, Inc.



이 도서는
Getting Started with Impala(O'REILLY)의
번역서입니다

처음 시작하는 임팔라 SQL로 하둡을 다루는 가장 쉬운 방법

초판발행 2015년 4월 21일

지은이 존 러셀 / **옮긴이** 양원국 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-726-2 15000 / **정가** 11,000원

총괄 배용석 / **책임편집** 김창수 / **기획·편집** 김상민
디자인 표지/내지 여동일, 조판 최송실
마케팅 박상용 / **영업** 김형진, 김진불, 조유미

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자/역자 소개

지은이_ **존 러셀**

존 러셀John Russell은 소프트웨어 개발자이면서 테크니컬 라이터로, 현재 클라우데라 임팔라 프로젝트의 문서화를 이끌고 있다. 산업을 선도하는 팀의 구성원으로 데이터베이스와 SQL 분야를 경험했다. DB2의 초기 정보센터Information Center를 설계하고 제작했다. 오라클 데이터베이스에서 애플리케이션 개발 관련 주제를 문서화하고 프로젝트 타히티Project Tahiti 문서 검색 엔진을 설계하고 코딩했으며, MySQL에선 InnoDB 스토리지 엔진을 문서화했다. 캐나다의 뉴펀들랜드Newfoundland 출생으로 현재는 미국의 캘리포니아 주 버클리Berkeley에 산다.

옮긴이_ **양원국**

티맥스소프트에서 APM 솔루션을 개발했고, 그 후 빅데이터 전문회사 KT NexR에 재직하면서 빅데이터 처리에 오픈 소스 기술을 적용하고 운용하는 일을 했다. 현재는 프리랜서로 일하고 있다.

역서로는 『하이브 완벽 가이드』(한빛미디어, 2013, 공역), 『아파치 Kafka 따라잡기』(에이콘 출판사, 2014), 『Hadoop과 Solr를 이용한 기업용 검색 시스템 구축』(에이콘 출판사, 2014)이 있다.

임팔라는 하둡 위에 동작하는 질의 엔진으로, 빅데이터 하둡 플랫폼을 제공하는 양대 산맥 중 하나인 클라우데라에서 2012년 베타 테스트 배포판과 함께 처음 공개했다. 2013년 5월에 GA 버전이 나왔으며, 2013년 말에는 아마존 웹서비스, 2014년 초에는 MapR에서도 지원하기 시작했다.

임팔라를 적용하고 운영함에 있어 초기 진입 장벽은 낮은 편이다. 하둡 클러스터에 몇 개의 추가 데몬만 구동하면 바로 임팔라를 사용할 수 있다. 임팔라의 또 다른 장점은 사용자층이 두꺼운 SQL을 사용해 대용량 데이터를 대상으로 대화형 질의를 수행할 수 있다는 점이다. 하둡에 있는 데이터를 별도의 변형 없이 바로 질의할 수 있어서 다른 시스템으로 데이터를 처리하는 마이그레이션 작업을 요구하지 않는다. 따라서 추가 공간 낭비를 없애 줄 뿐만 아니라 이전한 데이터가 최신 데이터와 같은지 보장하기 위해 시스템을 복잡하게 만들고 추가 제약사항을 만드는 문제도 발생하지 않는다.

또한, 임팔라는 SQL로 하둡 데이터를 대화형으로 다루는 다른 오픈소스 프로젝트보다 엔터프라이즈 환경에 부합하는 보안 및 권한 관리 기능이 강력해 별도의 추가 개발 없이 엔터프라이즈 환경에 쉽게 적용할 수 있다. 빅데이터 플랫폼을 다루는 대표적인 회사인 클라우데라에서 프로젝트를 이끌어 가기 때문에 프로젝트가 안정적으로 발전할 기반이 있다는 점은 임팔라의 신뢰성을 보장한다. 특히 클라우데라나 MapR 하둡 배포판을 사용할 때 임팔라를 이용하면 다른 오픈소스 프로젝트보다 별도의 통합 검증 과정 없어 매우 편리하다.

하둡 환경을 테스트할 수 있고 SQL에 대한 기본 지식이 있다면 이 책에 있는 내용을 따라 하는 데 어려움이 없을 것이다.

이 책은 임팔라 설치부터 데이터를 다루는 깊이 있는 내용까지 실제 임팔라를 사용하면서 마주치는 문제를 쉽게 해결할 수 있는 좋은 가이드가 될 것이다.

클라우데라 임팔라는 오픈소스 프로젝트로 아파치 하둡 소프트웨어 스택을 데이터베이스 분석가를 포함한 개발자와 일반 사용자가 보다 편하게 사용할 수 있도록 돕는다. 임팔라 대량병렬처리massively parallel processing, MPP 엔진은 SQL 질의를 통해 하둡 데이터를 SQL에 친숙한 데이터베이스 분석가와 비즈니스 인텔리전스business intelligence 도구 사용자가 쉽게 사용할 수 있도록 단순화하고 사용자가 쌍방향 탐사와 실험을 할 수 있을 정도로 빠르게 해준다.

다시 말하면 임팔라 소프트웨어는 연결된 머신 클러스터에 분산되는 고성능 SQL 질의를 목적으로 만들어졌다.

누구를 위한 책인가

이 책은 데이터베이스, 데이터 웨어하우스, 빅데이터에 관한 지식이 있는 독자를 대상으로 한다. 따라서 `CREATE TABLE`, `SELECT`, `INSERT` 같은 구문과 주요 절에 관한 설명이 따로 필요 없을 정도로 독자는 SQL에 충분히 경험이 있다고 가정한다. 또한, 리눅스 경험이 있으면 더욱 좋다. 아파치 하둡 소프트웨어 스택 경험이 있으면 유용하나 필수는 아니다.

데이터베이스 경험은 있지만 아파치 하둡 소프트웨어 스택 경험은 없는 사람을 대상으로 임팔라 아키텍처와 사용법 측면에서 사례를 알려준다.

SQL 예제는 이해를 돕는 단순한 예제를 시작으로 고성능과 확장성을 보여주는 모범 사례를 다루는 예제로 확장해 나간다.

이 책의 표기법

이 아이콘은 일반적인 주석을 의미한다.

이 아이콘은 팁이나 제안을 의미한다.

이 아이콘은 경고나 주의를 의미한다.

본문 및 코드 내 굵은 글씨

코드 명령어 또는 여러분이 문자 그대로 입력해야 하는 텍스트를 의미하거나 SQL 문을 강조하기 위해 사용하였습니다.

예제 코드 내려받기

보조 자료(예제 코드, 연습 문제 등)는 https://github.com/oreillymedia/get-started-impala에서 내려받을 수 있습니다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾기도 쉽지 않습니다. 또한, 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1 eBook First
빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다

500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 좀 더 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한, 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2 무료로 업데이트되는 전자책 전용 서비스입니다

종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3 독자의 편의를 위해 DRM-Free로 제공합니다

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 어려운 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

chapter 1

왜 임팔라인가

아파치 하둡 에코시스템은 데이터에 중점을 두고 있어 SQL 경험이 있는 데이터베이스 개발자에 잘 맞는다. 하둡 애플리케이션 개발 작업의 많은 부분이 데이터 파일을 복사하고 변환하고 재조직화하여 분석하는 프로그램을 작성하는 일로 이루어져 있다. 이러한 일들을 대규모 병렬 방식의 네트워크 장비로 묶인 클러스터에서 신뢰성 있게 수행하는 데 엄청난 노력이 필요하다. 임팔라는 이러한 활동을 쉽고 빠르게 해주기 때문에 분산 컴퓨팅에 관한 전문적인 지식이나 새로운 API를 배울 필요도 없다. 또한, 하려는 작업을 단일 SQL 문장으로 표현할 수 있다면 프로그램조차 작성할 필요가 없다.

1.1 빅데이터 에코시스템에서 임팔라의 지위

클라우데라 임팔라 프로젝트는 빅데이터 세계에 적기에 나타났다. 데이터양 증가 속도가 빨라져 현실적으로 단일 서버에 저장하고 처리할 수 있는 양을 넘어서고 있다. 또한, 더 많은 사용자와 개발자가 하둡 소프트웨어 스택을 실제로 적용하기 위해 노력하고 있다.

임팔라는 유사 데이터베이스 ETL[Extract-Transform-Load] 처리에 높은 유연성을 가지고 있다. 다양한 표준 하둡 파일 포맷으로 된 데이터를 질의할 수 있다("3.7 파일 포맷"을 참조하자). 데이터를 변환하거나 복제할 필요 없이 아파치 하이브[Apache Hive], 아파치 피그[Apache Pig], 클라우데라 서치[Cloudera Search] 같은 하둡 구성요소와 임팔라 조합

으로 같은 데이터에 접근할 수 있다. 질의 속도가 중요하다면 컬럼 지향의 파케이 Parquet 파일 포맷을 사용해 데이터 재조직화를 단순하게 만들어 데이터 웨어하우스 스타일 질의 성능을 최대화할 수 있다.

전통적으로 빅데이터 처리는 메인 프레임 시대의 배치 작업과 닮아서 예상치 못한 어려운 작업을 처리하는 데 밤샘 작업이나 주말 작업을 해야만 했다. 이에 유사 SQL 문법으로 직접 만든 임팔라의 목적은 아무리 복잡한 질의라도 빠르게 돌아수 초 또는 수 분내에 답을 얻는 것이라고 할 수 있다. 응답성을 흔히 "인터랙티브"하다고 말하기도 한다.

임팔라는 SQL과 SQL 기반의 비즈니스 인텔리전스BI, Business Intelligence 도구 사용자를 고려해 새로운 분석 기법이 나올 때마다 자바 프로그램을 만들 필요 없는 효과적인 개발 모델을 가지고 있다. SQL 언어는 컴퓨터 산업에서 역사가 오래되었지만, 여기서는 빅데이터와 암팔라의 조합으로 새롭게 태어날 것이다.

펄 사용자가 텍스트 처리 스크립트를 하는 것처럼 복잡한 분석 질의를 자연스러운 표현 표기법으로 작성할 수 있다. 대형 데이터 집합과 데이터 구조를 파이썬 셸 내부의 파이서니스타Pythonista01와 같이 쌍방으로 답파할 수 있다. 또한, 장황한 특화 API를 암기하지 않아도 된다. SQL은 강력한 표준 명령어 집합인 RISC 명령어 집합과 유사하다. 시각화나 그래핑 등을 수행하는 API 라이브러리 사용이 필요할 때 C++, 자바, 파이썬 등으로 만들어진 프로그램에서 JDBC나 ODBC 프로토콜을 통해 임팔라 데이터에 접근할 수 있다.

직접 SQL 코드를 만들 필요 없이 내부에서 SQL을 사용하는 비즈니스 도구를 사용할 수 있는 것도 장점이다. 예를 들어 IBM 코그너스IBM Cognos, SAP 비즈니스 오브젝트SAP Business Objects, 마이크로스트레티지MicroStrategy 같은 전통적인 비즈니스

01 http://omz-software.com/pythonista/

인텔리전스 도구나 타블로Tableau 같은 새로운 세대의 데이터 발견 도구data discovery tools도 사용할 수 있다.

1.2 빅데이터 워크플로우 유연성

임팔라는 기존 하둡 컴포넌트인 보안, 메타데이터, 스토리지 관리, 파일 포맷과 연동한다. 이런 하둡의 강점인 유연성에 더해 SQL 질의를 전보다 빠르고 쉽게 처리한다.

SQL로 복잡한 분석 프로그램을 단순하고 직관적인 질의로 만들 수 있다. 질문에 대한 답을 찾고 문제를 해결할 때, SQL을 알고 있거나 SQL 위에 구축한 표준 BI 도구를 다루는 분석가를 구하면 된다. 그들은 SQL이나 BI 도구로 대량의 데이터 집합을 분석하는 법을 알고 있으며, 다양한 종류의 비즈니스 질문과 "만약에" 같은 시나리오에 대한 정확한 답을 빠르게 얻는 방법도 알고 있다. 일반 사용 사례나 독특하고 예상하지 못한 시나리오를 분석을 하는 데 필요한 데이터 구조를 추상화하고 데이터 구조를 어떻게 설계하는지 알고 있다.

SQL의 필터링, 계산, 정렬, 포맷팅 기술로 최종 결과를 만들어내는 클라이언트 측 로직과 많은 양의 원본 데이터를 만드는 대신 이런 종류의 일을 임팔라 질의 엔진에 위임할 수 있다.

임팔라는 대량의 데이터 집합도 작은 데이터처럼 다루기 쉽고 경제적이어야 한다는 빅데이터 철학을 담고 있어서 대량의 데이터도 밑에 깔린 데이터 변경 없이 즉시 유입시킬 수 있다. 질의할 데이터를 원래 원본 형태로 두어 유연성을 가지게 하거나 자주 질의하는 데이터를 좀 더 치밀하고 최적화된 형태로 변환할 수 있다. 어느 방식으로든 어떤 데이터를 저장해야 할지 고민할 필요가 없어 데이터 콘덴싱condensing으로 요약된 형태의 정보를 가지고 있는 대신 원래 값을 그대로 보존할 수 있다. 전통적 데이터 웨어하우스 환경에서 볼 수 있는 데이터 재조직화나 유연성

없는 구조를 적용해야 할 필요가 없다.

임팔라가 사용하는 데이터 파일은 명세서가 모두 공개되어 있고 문서화되어 있으며 상호 정보 교환이 가능한 포맷이다(일부는 사람이 읽을 수도 있다). 임팔라를 다른 하둡 구성요소와 같이 사용할 때도 데이터를 변환하거나 복사할 필요가 없어 차세대 데이터 처리 소프트웨어와 함께 작업한다고 해도 원래 데이터 파일을 계속 사용할 수 있다.

1.3 고성능 분석

임팔라 아키텍처는 하둡 데이터에 SQL 질의 속도를 증가시켜 일하는 방식이 효율적으로 바뀔 수 있다. 맵리듀스(MapReduce)나, 하이브 같은 다른 SQL 온 하둡(SQL on Hadoop) 기술보다 임팔라 질의 응답의 빠른 회전이 더 많은 문제를 해결할 수 있다. 그러므로, 하둡 데이터 분석을 대규모 계획과 일정표가 필요한 배치 작업으로 다루는 대신 원할 때 언제든지 결과를 얻는 방식으로 사용할 수 있고, 각 질의가 끝나기를 기다리면서 정신이 왔다 갔다 하는 대신 질의를 실행하고 바로 결과를 검토해 고쳐 나갈 수 있다. 이런 빠른 반복을 통해 워크플로우에 방해 없이 최적의 해법을 얻을 수 있다. 데이터를 대표하는 부분집합으로 줄이는 대신 가지고 있는 모든 것을 분석해 가장 정확한 답을 얻고 새로운 추세나 연관을 발견할 수 있다.

명령어 처리나 연산을 하는 데 너무 오래 걸려 티타임을 갖거나 다른 작업을 해야 하는 느린 컴퓨터나 소프트웨어를 사용한 경험이 있을 것이다. 그러다 더 빠른 소프트웨어를 쓰거나 컴퓨터를 업그레이드하면 시스템 반응이 빨라 일의 활기를 찾고 집중하게 되며 새로운 생각을 쉽게 떠올릴 수 있게 된다. 이는 임팔라가 하둡 사용자에게 전달하고픈 긍정적인 영향의 목표다.

1.4 탐색적 비즈니스 인텔리전스

이미 비즈니스 인텔리전스 질의를 작성해 봤다면 일반적으로 다루었던 데이터는 높은 정보 가치를 가진, 다룰 수 있을 정도의 양으로 응축된 데이터였을 것이다. 그리고 데이터베이스 시스템이 적재하기 위해 복잡한 추출-변형-적재extract-transform-load, ETL과정을 거쳤을 것이다.

임팔라는 이러한 과정을 줄여주어, 하둡에 데이터가 들어오고 몇 단계 후에는 임팔라에서 즉시 질의할 수 있게 된다. 고용량, 고속 스토리지인 하둡 클러스터는 가치가 있다고 생각하는, 부분 집합이 아닌 전체 데이터를 가져다준다. 임팔라는 원본 데이터 파일을 질의할 수 있기 때문에 예전 데이터베이스에서 시간이 오래 걸리는 데이터를 적재하고 재조직화하는 단계를 생략할 수 있다.

빠른 종단간 처리end-to-end process는 분석 질의에 새로운 가능성을 열어준다. 탐색적 데이터 분석exploratory data analysis[02]과 데이터 탐사data discovery[03] 같은 기술에 사용할 수 있다. 소프트웨어 초기 세대처럼 모든 데이터를 데이터 웨어하우스에 저장하여 비용을 낭비하고 사용 가능한 형태로 데이터를 적재하고 변환하는 데 시간이 많이 드는 걸 원치 않을 것이다.

구분자가 있는 텍스트 파일 같이 단순한 포맷의 원본 데이터를 받을 수도 있다. 텍스트 파일은 부피가 크고 질의에 특별히 효과적이지 않지만, 탐색적 비즈니스 인텔리전스에 특별히 문제가 되지는 않는다. 방대한 데이터 집합을 분석해 얻을 수 있는 새로운 통찰이 무엇인지 판단하려는 의도로 그러한 데이터에 질의를 하는데, 이를 통해 추세를 식별하고 흥미로운 부분 집합을 발견할 수도 있고 밑에 깔린 데이터 구조와 맞는 스키마schema를 어떻게 설계할지 배울 수 있다. 탐색적 BI는 일반적으로 에드 혹ad hoc 질의와 관련이 있다. 어떤 것을 찾기 위해 질의하고 미세

02 http://en.wikipedia.org/wiki/Exploratory_data_analysis

03 http://en.wikipedia.org/wiki/Data_discovery

조정하는 일을 반복한다. "어떤 OO가 있는가, "무엇이 가장 OO한가?" 등과 같은 질문을 쫓는 데 사용하는 질의는 `MAX()`, `MIN()`, `COUNT()`, `AVG()` 같은 집계 함수와 보통 관련이 있다.

주기적으로 실행해야 할 질의가 무엇인지 안다면 데이터와 스키마를 가능한 최적화할 수 있다. 집중적으로 분석할 데이터를 텍스트 파일이나 다른 최적화되지 않은 파일 포맷에서 더 좋게 바꾸고 싶다면 데이터를 압축된 컬럼 형태의 파일 포맷으로 바꾼다. 이 파일 포맷을 파케이Parquet 포맷이라 한다(하둡 경험이 있다면 이미 파케이 포맷을 데이터 파이프라인에서 사용 중일 것이다. 파케이 포맷을 이미 사용 중이면 변환 단계를 생략할 수 있고 탐색적 BI 단계에서 질의 속도의 향상을 누릴 수 있다).

chapter 2

임팔라 준비와 구동

아파치 하둡 전문성 여부나 하둡 인프라를 얼마나 가지고 있느냐에 따라 임팔라를 사용하는 방식이 다를 수 있다.

이 책의 일부 예제에서 사용하는 문법, 함수, 그 밖의 다른 기능들은 클라우데라 CDH 5.1과 CDH 4 하둡 배포판에서 사용할 수 있는 임팔라 1.4부터 도입되었다.

2.1 설치

클라우데라 라이브 데모Cloudera Live Demo

설치 없이 실행하는 가장 쉬운 방법은 클라우데라 라이브 데모Cloudera Live demo[01] (사용자 등록 선택사항이 있다)를 사용하는 법이다. 휴Hue 웹 인터페이스를 통한 임팔라 질의 편집기를 사용해 TPC-DS 벤치마크 스위트에서 온 예제 테이블 몇 개를 살펴보고 SQL 코드로 질의를 실행시키며 테이블을 만들어 데이터를 적재할 수 있다.

클라우데라 퀵스타트 브이엠Cloudera QuickStart VM

데이터베이스를 다룰 줄 아는 하둡 중급자라면 클라우데라 퀵스타트 브이엠 Cloudera Quick Start VM[02]을 통해 기본 임팔라 기능을 바로 사용해 볼 수 있다. 클라우데

01 http://go.cloudera.com/cloudera-live.html
02 http://bit.ly/quickstart-VM

라 쿽스타트 가상머신은 단일노드 가상머신 환경으로 임팔라 주요 기능에 익숙해지는 용도로 적합하다(성능이나 확장 테스트를 하려면 단일 사용자 단일 머신을 우선 마스터해야 할 것이고 클라우데라 매니저를 통해 전체 CDH 배포판을 실제 머신이나 고성능 가상머신에 설치해야 한다). 쿽스타트 가상머신을 VMWare, KVM, VirtualBox에 실행하고 클라우데라 매니저Cloudera Manager 웹 인터페이스를 통해 임팔라 서비스를 실행하면 `impala-shell` 인터프리터나 ODBC, JDBC 인터페이스를 통해 상호작용할 수 있다.

클라우데라 매니저Cloudera Manager와 CDH5

좀 더 깊이 있게 테스트하거나 대규모로 적용하려면 CDH5[03] 배포판 일부인 클라우데라 임팔라 소프트웨어를 설치하고 실제 클러스터 환경에서 사용한다. 단일 패키지나 업그레이드를 쉽게 해주는 클라우데라 매니저 파셀parcel 기능을 사용해 자유롭게 소프트웨어를 설치할 수 있다. 임팔라 서버를 각 데이터 노드와 하나의 지정 노드(일반적인 하둡 데이터 노드와 같다)에 설치해 임팔라 스테이트스토어StateStore 데몬을 구동할 수 있다. 가장 간단한 준비와 구동 방법은 클라우데라 매니저 애플리케이션을 이용하는 것이다. 하둡 클러스터를 임팔라와 함께 설치하는 전체 과정을 클러스터의 호스트 이름 목록으로 지정하기만 하면 시작할 수 있다.

수동 설치

수동 설치[04] 방식은 잘 사용하지 않는 방식이다. 클러스터의 모든 데이터 노드에 설치 과정을 적용해야 하므로 퍼펫Puppet이나 셰프Chef 같은 분산 소프트웨어 관리에 익숙한 사용자에게만 적합하다.

소스 빌딩하기

03 http://bit.ly/cdh5-dist
04 http://bit.ly/impala-manual

임팔라가 어떻게 동작하는지 깊이 있게 이해하기 원한다면 GitHub에서 임팔라 소스코드[05]를 얻어 직접 빌드해보면 된다. C++이나 자바 소스코드로 임팔라를 들여다보는 일은 재미있고 분산 컴퓨팅에 대해 많은 것을 배울 수 있지만 SQL 개발자나 업무 분석가를 대상으로 하는 이 책의 범위에는 벗어난다.

임팔라를 어떻게 시작하든지 사용자모임[06] 또는 메일링리스트[07]로 오픈소스 프로젝트 사용자 모임에 참여할 수 있다. 임팔라 웹사이트 전용 커뮤니티 자원 전체 목록[08]을 참고하자.

2.2 임팔라 접속

어떤 방식으로 임팔라를 설치하든지 마지막 과정에는 클러스터에서 임팔라 관련 데몬(impalad, catalogd, statestored) 몇 개를 시작해야 한다. 클라우데라 매니저 방식으로 설치하면 이러한 데몬을 묶어 하나의 임팔라 서비스로 만들어 한번에 시작하고 중지할 수 있다.

임팔라에 접근하는 가장 편리하고 유연한 방법은(이 책에서 주로 다루는) 바로 인터랙티브한 impala-shell 인터프리터를 사용하는 것이다. 개발 환경에서 산업 등급의 보안 기능을 활성화하하기 전에는 임팔라를 구동시킬 클러스터 내 호스트 이름만 알면 된다. 임팔라 질의 엔진은 완전히 분산화되어 있어서 아무 데이터 노드에 접속해 질의를 던지면 자동으로 클러스터에 분산해 작업한다.

단일 노드 가상머신에 사용할 때 편리하게 impala-shell은 기본설정으로 로컬 호스트에 접속하여(단일 노드 가상머신을 사용할 때 편리하다) 클러스트 내 임의의 데이

05 https://github.com/cloudera/impala
06 http://community.cloudera.com/t5/Interactive-Short-cycle-SQL/bd-p/Impala
07 http://bit.ly/google-impala-list
08 http://impala.io/community.html

터 노드에 로그인한다. 원격 서버에 접속하려면 -i 옵션을 사용한다. 기본 포트인 21000외의 다른 포트를 사용한다면 -i 옵션의 인자에 포트와 호스트를 적으면 된다.

다음은 로그인한 같은 머신에 구동 중인 임팔라 서버에 접속하는 방법을 보여주는 예다.

```
$ ssh impala_coder@host07.dev_test_cluster.example.com
$ impala-shell
[localhost:21000] > show databases;
```

다음은 원격 시스템의 임팔라 서버에 접속하는 방법이다.

```
$ ssh my_login@personal_linux_server.example.com
$ impala-shell -i host12.dev_test_cluster.example.com
[host12.dev_test_cluster.example.com:21000] > show tables;
```

다음은 기본 포트가 아닌 포트를 리스닝하는 원격지의 임팔라 서버에 접속하는 방법이다.

```
$ impala-shell -i host33.dev_test_cluster.example.com:27000
[host33.dev_test_cluster.example.com:27000] > create table foo (x int);
```

앞에서 다룬, 어디서나 접속할 수 있는 방식은 개발 단계에서만 사용할 수 있다. SSL[Secure Socket Layer] 인증과 권한부여 같은 보안 기능을 구성하기 전에 아무나 임팔라 인스턴스에 접속할 수 있다면 아마도 작업에 문제가 생기거나 성능이 저하될 수도 있다. 시스템 관리자나 네트워크 관리 담당자의 도움이 필요한 보안 설정 부분은 이 책에서 다루지 않으니 임팔라 보안 문서[09]를 참고하라.

09 http://bit.ly/impala-security

2.3 첫 임팔라 질의

좀 더 쉽게 배울 수 있도록 바로 사용해 볼 수 있는 질의를 몇 개 준비했다. 이들 질의는 임팔라 표준 SQL 문법을 설명하는 데에도 사용되지만 설치와 설정을 바로 했는지를 확인하는 데에도 사용된다.

데이터 타입이나 연산식 같은 기본 임팔라 질의 문법 요소를 활용할 수 있게 테이블이나 **WHERE** 문이 전혀 없는 다음 질의를 수행해 보자.

```
SELECT 1;
SELECT 2+2;
SELECT SUBSTR('Hello world',1,5);
SELECT CAST(99.5 AS INT);
SELECT CONCAT('aaa',"bbb",'ccc');
SELECT 2 > 1;
SELECT NOW() + INTERVAL 3 WEEKS;
```

이런 종류의 질의는 수학 연산이나 데이터 타입 변환, 내장 함수를 실험하는 데 유용하다. 유사 기법에 대한 더 많은 예제는 "5.2 튜토리얼: 테이블 없는 질의"에서 볼 수 있다.

임팔라는 미리 만들어진 테이블을 가지고 있지 않기 때문에 실제 데이터에 질의하려면 준비가 좀 필요하다. 다음 예에서 **INSERT ... VALUES** 문을 사용해 "장난감" 테이블 몇 개를 만들 것이다(확장성scalability 이유로 VALUES 문은 대량의 데이터를 다룰 때 적합하지 않아 실제 운영환경에서는 **INSERT ... SELECT** 문을 대신 사용한다).

```
-- 약어에 바탕을 둔 이름을 찾는 테이블을 만듦.
CREATE TABLE canada_regions (name STRING, abbr STRING);

-- 데이터값을 가진 크기가 커질 수 있는 테이블로 만듦. 질문에 답하는 데 사용할 것이다.
CREATE TABLE canada_facts
  (id STRING, sq_mi INT, population INT);
```

```
-- INSERT 문은 INSERT INTO를 통해 기존 테이블의 데이터에 추가하거나 INSERT
OVERWRITE로 완전히 대치한다.
INSERT INTO canada_regions VALUES
  ("Newfoundland and Labrador" ,"NL"),
  ("Prince Edward Island","PE"),
  ("New Brunswick","NB"), ("Nova Scotia","NS"),
  ("Quebec","PQ"), ("Ontario","ON"),
  ("Manitoba","MB"), ("Saskatchewan","SK"), ("Alberta","AB"),
  ("British Columbia","BC"), ("Yukon","YT"),
  ("Northwest Territories","NT"), ("Nunavut","NU");

INSERT OVERWRITE canada_facts VALUES ("NL",156453,514536),
  ("PE",2190,140204), ("NB",28150,751171), ("NS",21345,921727),
  ("PQ",595391,8054756), ("ON",415598,13505900),
  ("MB",250950,1208268), ("SK",251700,1033381),
  ("AB",255541,3645257), ("BC",364764,4400057),
  ("YT",186272,33897), ("NT",519734,41462), ("NU",78715,31906);

-- 단일 테이블에 질의하거나 조인을 통해 여러 테이블에 질의할 수 있고
-- 뷰 위에 새로운 질의를 만들 수 있다.
SELECT name AS "Region Name" FROM canada_regions
  WHERE abbr LIKE 'N%';
+-----------------------+
¦ region name           ¦
+-----------------------+
¦ Newfoundland and Labrador ¦
¦ New Brunswick         ¦
¦ Nova Scotia           ¦
¦ Northwest Territories ¦
¦ Nunavut               ¦
+-----------------------+

-- 조인 질의로 인구 수치를 하나의 테이블에서 가져오고 다른 테이블에서 전체 이름을 가져온
다.
SELECT canada_regions.name, canada_facts.population
  FROM canada_facts JOIN canada_regions
  ON (canada_regions.abbr = canada_facts.id)
  ORDER BY population DESC;
+---------------------------------+
¦ name                    ¦ population ¦
+---------------------------------+
```

```
| Ontario                   | 13505900   |
| Quebec                    | 8054756    |
| British Columbia          | 4400057    |
| Alberta                   | 3645257    |
| Manitoba                  | 1208268    |
| Saskatchewan              | 1033381    |
| Nova Scotia               | 921727     |
| New Brunswick             | 751171     |
| Newfoundland and Labrador | 514536     |
| Prince Edward Island      | 140204     |
| Northwest Territories     | 41462      |
| Yukon                     | 33897      |
| Nunavut                   | 31906      |
+---------------------------------+
```

```
-- 뷰는 긴 질의의 엘리어스(alias)이고 스토리지를 설정하는 데 시간이 들지 않는다.
-- 뷰를 질의에 사용함으로써 반복 구문을 피할 수 있고
-- 복잡한 질의를 쉽게 읽을 수 있게 해준다.
CREATE VIEW atlantic_provinces AS SELECT * FROM canada_facts
  WHERE id IN ('NL','PE','NB','NS');
CREATE VIEW maritime_provinces AS SELECT * FROM canada_facts
  WHERE id IN ('PE','NB','NS');
CREATE VIEW prairie_provinces AS SELECT * FROM canada_facts
  WHERE id IN ('MB','SK','AB');

-- 선택문에서 뷰를 이용해 일련의 필터와 함수를 조합해 사용하게 해준다.
SELECT SUM(population) AS "Total Population"
  FROM atlantic_provinces;
+----------------+
| total population  |
+----------------+
| 2327638           |
+----------------+
SELECT AVG(sq_mi) AS "Area (Square Miles)"
  FROM prairie_provinces;
+-----------------+
| area (square miles) |
+-----------------+
| 252730.3333333333  |
+-----------------+
```

테이블, 질의, 간단한 시나리오에서 시작해 파티션된 수 기가바이트의 테이블을 다루는 추가 예제는 5장을 참고하자.

앞에서 언급한 대로 **INSERT ... VALUES** 문은 임팔라로 대량의 데이터를 확장성 없는 방식으로 가져오고 만들어진 데이터도 효율적으로 질의하게 조직화하지 않는다. 사실 작업할 대부분의 대량 데이터는 임팔라 외부에 있을 것이다. **LOAD DATA** 문을 통해 이들 데이터를 가져와 수백에서 수천의 행을 한번에 연산하는 **INSERT ... SELECT** 문에 의해 복사되고 변형될 것이다. 이런 기법의 예제는 "4.1 임팔라 테이블에 데이터 입력"에 있다.