Tech Notes (vol.9)

Informix Web DataBlade 아키텍처

소개

Informixa Web DataBladea는 매우 복잡한 동적 멀티미디어 기반의 웹 사이트를 유지하고, 관리하고, 전달할 수 있게 해 줍니다. 이 글에서는 이와 관련한 아키텍처 구성 요소에 대한 명확한 실례를 살펴보고 이의 작동 원리를 설명하고자 합니다. 여기서 다루고자 하는 주제가 Web DataBlade이므로 DataBlade 및 DataBlade 함수들을 직접 활용할 수 있도록 상세한 설명이 포함된 단원을 추가하였습니다. 그러나 이처럼 상세한 설명이 많이 나온다고 해도 이 중에서 핵심 요점은 빠뜨리지 않으면서 DataBlade의 작동 원리에 대해 설명할 것입니다. Web DataBlade의 작동 원리를 이해함으로써 웹 프로젝트 수행 시 중요한 디자인 결정에 도움을 받을 수 있을 것입니다.

이 글에서는 다음과 같은 주제를 다룹니다.

- Informix Web DataBlade 아키텍처
- 동적 페이지 생성의 작동 원리
- webexplode() DataBlade 기능의 이해

Informix Web DataBlade 아키텍처

그림 1은 Informix Web DataBlade의 구성 요소를 보여줍니다.

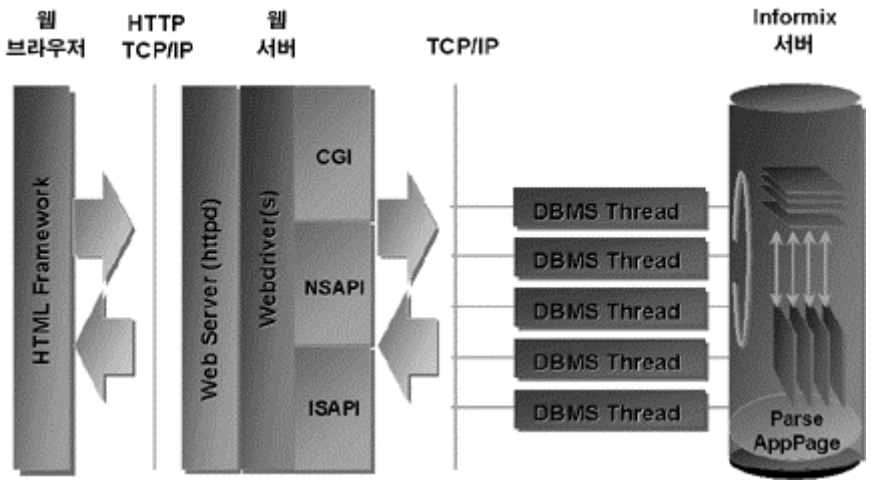

그림 1: Informix Web DataBlade 아키텍처

위 그림에서 볼 수 있듯이 웹 브라우저, 웹 서버, Informix 데이터베이스 서버, 이렇게 세 가지의 필수 플랫폼이 있습니다. 이 플랫폼들은 각기 자신만의 특화된 작업을 수행하는 논리적으로 독립된 호스트 또는 컴퓨터임을 알 수 있습니다. 표 1은 이러한 플랫폼들과 이들의 상호 작용 방식을 보여줍니다.

플랫폼	구성 요소

	미들웨어와 기타 필요한 공유 객체를 호스팅합니다.
TCP/IP	TCP/IP는 webdriver 프로그램과 Informix 서버 사이의 네트워크 통신을 지원합니다. webdriver와 Informix 서버가 같은 호스트에 있다면 공유 메모리, 스트림 파이프(stream pipes) 또는 명명된 파이프(named pipes)를 사용할 수 있습니다.
Informix 데이터베이스 서버	Informix Dynamic Servera with Universal Data Optiona 또는 Informix Dynamic Server.2000에 대한 UNIX 또는 NT4 호스트

표 1: Informix Web DataBlade 아키텍처의 논리적 플랫폼 구성 요소

아키텍처 소프트웨어 구성 요소

표 2는 아래에서 볼 수 있듯이 각 플랫폼의 소프트웨어 구성 요소를 보다 상세히 열거하고 있습니다.

구성 요소	설명
HTML Framework	브라우저 소프트웨어에서 처리하는 웹 응용 프로그램. 웹 서버를 제외한 다른 플랫폼 구성 요소와는 직접적인 상호 작용이 없음.
HTTPD 웹 서버	웹 서버 데몬. 이것은 CGI 표준 웹 서버 중 하나일 수 있습니다. API 구현이 필요하다면 이것은 Netscape(Fasttrack/Enterprise), Microsoft(IIS) 또는 Apache 중 하나여야 합니다.
webdriver(s)	webdriver는 Informix 데이터베이스에 연결되어

	webexplode() DataBlade 함수에 의해 웹 페이지가 표출되도록 요구하는 프로그램입니다.
CGI	webdriver의 CGI 구현. 이것은 각각의 모든 CGI 요청이 있을 때마다 호출됩니다.
NSAPI	webdriver는 Netscape Fasttrack과 Enterprise 같은 웹 서버와 통합하기 위한 공유 객체로서 구현됩니다. 구성 가능한 webdriver 스레드의 수는 URL 요청 시 만들어집니다.
ISAPI	webdriver는 IIS와 통합하기 위한 공유 객체로서 구현됩니다. 구성 가능한 webdriver 스레드의 수는 URL 요청 시 만들어집니다.
DBMS Thread	webdriver가 Informix 서버에 연결되면 DBMS 스레드가 시작됩니다. NSAPI/ISAPI 연결 시에는 스레드가 유지됩니다. CGI 연결 시에는 페이지 구성이 일단 완료되면 스레드가 닫힙니다.
Parse AppPage	이것은 AppPage 템플리트에서 생기는 HTML 페이지를 구성하는 webexplode() DataBlade 함수입니다.

표 2: 아키텍처 소프트웨어 구성 요소

분산 구성 요소 아키텍처

하나의 물리적 플랫폼 상에 논리적 플랫폼 세 개를 모두 가질 수 있습니다. 예를 들면, NT4 Server의 경

우는 Netscape 브라우저, Netscape Web Server, Informix 데이터베이스 서버를 호스팅할 수 있습니다. 반면에 Netscape Web Server와 webdriver는 NT4에서 호스팅하고, Informix 데이터베이스는 다른 컴퓨터의 UNIX에서 호스팅할 수 있습니다. 표 3은 Web DataBlade 구성 요소의 분산 방식을 보여 주고 있습니다.

NT4 구성 요소	UNIX 구성 요소
웹 서버	Informix Dynamic Server.2000, Informix Internet Foundation.2000
NT4용 Web DataBlade(webdriver)	UNIX용 Web DataBlade
클라이언트/NT용 SDK. 이 분산 아키텍처는 기본적으로 SQLI 프로토콜을 구현하는 I-Connect 라이브러리를 필요로 합니다.	없음

표 3: Web DataBlade 구성 요소 분산

webdriver처럼 함께 운영되는 객체를 다른 컴퓨터에 재배치하는 결정은, 사용자 설치와 관련된 보다 큰 문제점들에 따라 몇 가지 요인을 바탕으로 이루어집니다. 예를 들면 다음과 같습니다.

- 현재의 아키텍처가 혼합형이고, 재정적 또는 기타 여러 이유로 인해 그러한 혼합형이 요구되는 경우.
- 성능상의 요구 사항 때문에 데이터베이스 서버 플랫폼이 Informix 데이터베이스와 클라이언트 응용 프로그램(4GL 보고서와 일괄 처리 작업 등) 전용으로 사용되며, 기타 컴퓨터들은 클라이언트 HTTP 웹 서버 요청(예를 들면 용량이 큰 사이트)을 처리하는 데에만 사용되는 경우.
- 보안상의 이유로 웹 서버는 방화벽 밖에 있으나 데이터베이스는 방화벽 안에 있는 경우.

그림 2는 구성 요소들의 초기 위치를 보여 줍니다. 웹 서버, webdriver, Informix 데이터베이스 서버(Web DataBlade가 설치된)는 인트라넷 맥락에서 동일한 호스트 컴퓨터(NT 또는 UNIX)에서 작동하고 있습니다.

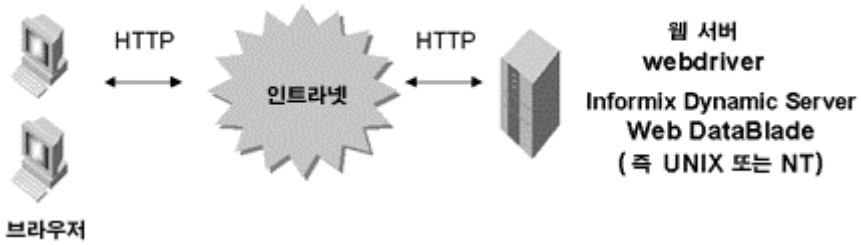

그림 2: 데이터베이스, 웹 서버, Web DataBlade 의 단일 서버 호스팅

그림 3은 서로 다른 서버들 사이에서 Web DataBlade의 진보된 구현을 보여 줍니다. webdriver 미들웨어가 웹 서버 소프트웨어와 함께 호스팅된다는 점에 유의하십시오. webdriver 미들웨어는 표 4에서 나타나 있는 종속성이 충족되는 한 네트워크를 통해 Informix 데이터베이스 서버와 통신할 수 있습니다.

NT 웹 서버 대 UNIX 데이터베이스	UNIX 웹 서버 대 NT/UNIX 데이터베이스
setnet을 사용하여 데이터베이스 호스트와 서버에 대해 초기화되는 레지스트리 항목	데이터베이스 호스트와 서버를 설명하기 위해 설정되는 Sqlhosts 항목

표 4: 연결 종속성

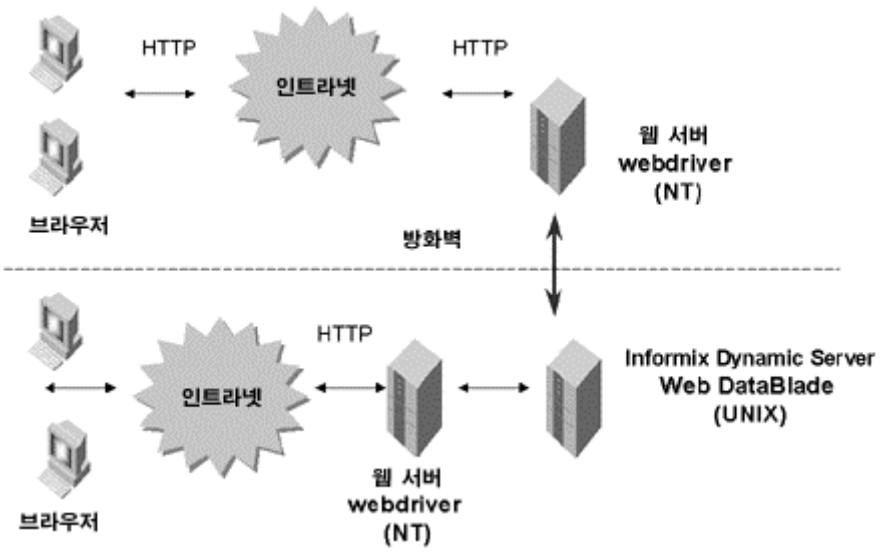

그림 3: 분산 웹 구성 요소 아키텍처

통신을 위한 프로토콜 사용

아키텍처의 각 구성 요소는 다른 구성 요소들과 메시지를 주고 받을 수 있어야 합니다. 프로토콜이란 컴퓨터와 소프트웨어 응용 프로그램이 상호 통신하는 방식을 정의하는 규칙의 집합입니다. 프로토콜은 서로 다른 컴퓨터에서 작동하는 다양한 응용 프로그램들이 공유된 "언어"를 이용하여 서로 통신할 수 있도록 해 줍니다. HTTP 프로토콜은 HTTP 클라이언트 (브라우저)와 HTTP 프로토콜 핸들러(웹 서버) 사이의 상호 작용을 정의합니다. 프로토콜 핸들러는 프로토콜의 명령을 해석하고 클라이언트와 서버 간에 통과된 데이터를 사용하여 프로토콜 명령에서 정의한 함수를 수행합니다.

- 프로토콜 형식

사용 목적에 따라 다양한 형식의 프로토콜이 있습니다. 웹 서버가 UNIX에 있건 NT4 Server에 있건 웹 브라우저는 개의치 않습니다. 그것은 웹 서버가 HTTP 프로토콜을 이해할 수 있기 때문

입니다. 브라우저 주소 입력란에 http://를 입력한다는 것은 기본 포트 번호 80의 대상 호스트에 있는 HTTP 서버와 프로토콜 교신을 시작함을 의미합니다. ftp://를 입력한다면 브라우저는 대상 지정된 FTP 서버와 포트 번호 21에서 FTP 프로토콜 교신을 시도할 것입니다.

- 프로토콜 레벨

HTTP에서 정의된 프로토콜 명령과 FTP에서 정의된 프로토콜 명령은 서로 다릅니다. TCP/IP (Transmission Control Protocol/Internet Protocol)는 컴퓨터 사이의 통신을 위한 전송/주소 지정 프로토콜입니다. HTTP나 FTP 프로토콜 명령은 TCP가 정의한 패킷에 의해 IP 주소(예를 들면 920.325.123.0)를 갖고 있는 컴퓨터로 전송됩니다. HTTP와 FTP는 응용 프로그램 레벨 프로토콜이고, TCP는 전송 레벨 프로토콜, IP는 인터넷-주소 지정 프로토콜입니다. 이더넷은 물리적 레벨에서 데이터를 전송하는 데에 사용됩니다. Informix 클라이언트 프로그램(예를 들면 webdriver)은 데이터베이스 서버에 연결되기 때문에 클라이언트 프로그램이라고 부릅니다. 그러나 Informix 데이터베이스 서버와의 통신은 클라이언트와 서버가 이해할 수 있는 SQLI라는 프로토콜을 통해서 이루어집니다. 이 경우 Informix 서버는 프로토콜 핸들러가 됩니다. 이는 응용 프로그램 개발자들이 쉽게 이해할 수 있는 내용입니다.

- 응용 프로그램 프로토콜

자신의 고유한 응용 프로그램 프로토콜을 정의할 수 있습니다. 단순히 일련의 명령어들을 정의한 후 이것들을 이해하고 해석할 수 있는 프로토콜 핸들러를 작성하기만 하면 됩니다. Java를 이용해서 네트워크를 통해 클라이언트와 통신이 가능한 프로토콜 핸들러를 매우 쉽게 구현할 수 있습니다. 다른 컴퓨터에서 실행되는 다른 일괄 처리 작업에 의존하여 각각의 컴퓨터에서 일괄 처리 작업이 실행되도록 하는 분산 일괄 처리 스케줄링 응용 프로그램을 예로 들 수 있습니다. 각 컴퓨터에 설치되어 있는 제어 프로그램이 다른 프로그램이 하는 작업을 이해할 수 있도록 프로토콜을 구현함으로써 클라이언트와 서버로서 상호 작용하는 구성 요소를 개발할 수 있습니다. 그것은 이들이 모두 언어와 독립된 프로토콜의 의미를 해석해야 하기 때문입니다. 이 경우 각 컴퓨터에 설치된 제어 프로그램은 프로토콜 클라이언트(명령을 보냄)임과 동시에 프로토콜 핸들

러(명령을 받아서 처리함)입니다.

동적 페이지 생성의 작동 원리

이 단원에서는 Informix Web DataBlade가 HTML 템플리트(AppPage), AppPage 스크립팅 태그, webdriver, Web DataBlade를 사용하여 동적 웹 페이지를 만드는 데 어떤 역할을 하는가를 설명합니다. 그림 4는 브라우저 클라이언트가 URL을 요청할 때 동적 웹 페이지를 생성하는 과정을 보여줍니다.

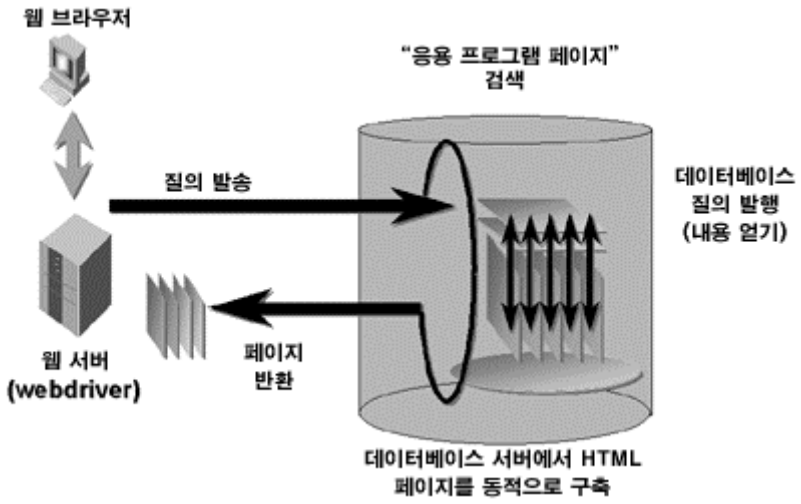

그림 4: 동적 웹 페이지 생성

이 그림에서 보이는 구성 요소를 표 5에서 설명합니다.

구성 요소	설명

	cgi/webdriver.exe?Mival=helloWorld
웹 서버	URL 요청을 처리함. 웹 서버는 웹 서버 구성 파일에 저장된 정보를 사용하여 webdriver.exe에 대한 CGI 호출을 URL로 매핑합니다.
webdriver	데이터베이스로부터 "HelloWorld"라는 페이지를 요청하고 웹 서버를 통해 이를 브라우저에게 반환하는 CGI 프로그램.
질의 발송	webdriver는 데이터베이스로 질의를 보내어 AppPage를 인출하고, webexplode() 함수를 사용하여 이를 처리합니다. "Mival=HelloWorld" 매개변수는 어떤 AppPage를 인출해야 하는지를 webdriver에게 알려 줍니다.
"응용 프로그램 페이지" 검색	Informix 서버는 데이터베이스에서 요청된 AppPage를 우선 선택함으로써 질의를 처리합니다.
내용 얻기	질의에서 사용되는 webexplode() 함수는 AppPage 스크립트 태그를 처리하는데, 여기에 SQL 문을 포함시켜 오디오와 비디오 파일 같은 데이터와 객체를 검색할 수 있습니다.
동적으로 구성된 HTML	데이터베이스에서 데이터가 검색되면 이것은 HTML 페이지로 포맷됩니다. 예를 들면, 고객의 목록은 HTML 표의 행으로 포맷되고 큰 객체 참조 내용은 적절한 HTML 태그로 처리될 수 있습니다.
페이지 반환	webdriver가 보낸 질의는 HTML 페이지로 결과를 반환합니다. 이 페이지는 웹 서버를 통해 브라우저로 되돌려집니다.

표 5: 동적 웹 페이지 생성 구성 요소

단계별 동적 페이지 생성

webexplode() DataBlade 함수 이해하기

webexplode()는 DataBlade 함수입니다. 이 함수는 Informix DataBlade API를 사용해서 Informix Dynamic Server와 상호 작용을 합니다. webexplode()는 DataBlade 함수이기 때문에 Informix Dynamic Server.2000이나 Informix Internet Foundation.2000을 필요로 합니다. 이러한 서버에서는 DataBlade API를 사용하여 써드 파티 공급업체, 고객, 컨설턴트들이 만든 webexplode()와 같은 함수를 연결할 수 있습니다. Informix는 이러한 함수를 마치 핵심적인 SQL 구현의 일부인 것처럼 SQL 문에서 사용할 수 있습니다.

webexplode() 함수는 Informix 데이터베이스에 저장된 데이터를 기반으로 동적 HTML 페이지를 구축합니다. webexplode()는 다음과 같은 작업을 수행합니다.

- HTML 내의 AppPage 스크립트 태그를 포함하는 AppPages를 분석하고 그 태그 내에 포함된 SQL 문과 프로세싱 명령을 동적으로 구축하고 실행합니다.
- 이 SQL 문과 프로세싱 명령의 결과를 포맷하고 결과적으로 작성된 HTML 문서를 클라이언트 응용 프로그램(webdriver)에 반환합니다.
- 예컨대, 다음과 같은 질의는 Web DataBlade 변수, 특히 Web DataBlade webexplode() 함수에 값을 할당할 것입니다. 이러한 질의는 dbaccess 또는 AppPage에서 실행할 수 있습니다.

맺음말

이 글에서는 Informix Web DataBlade 아키텍처를 소개하였습니다.

또한 AppPage가 어떤 방식으로 구축되고 동적 페이지 생성이 어떤 식으로 여러 대의 컴퓨터에서 분산되는지에 대해서도 설명하였습니다. 특히 webdriver 구성 요소와 NSAPI의 CGI 구현과 ISAPI 및 Apache API 구현 간의 차이점에 대해 설명하였습니다. 이러한 차이점은 Web DataBlade 응용 프로그램을 관리하는 방법을 이해하는 데 매우 중요한 사항입니다. 이와 함께 관리 및 아키텍처 이슈들은 Informix 데이터베이스 기반의 웹사이트 관리 능력을 전에 비해 향상시켜 주고 Informix Web DataBlade 구성 요소를 여러 컴퓨터에 재배치하는 방식을 결정하는데 도움을 줄 것입니다.

Informix Internet Foundation의 객체 관계형 기능: UDT, UDR, 사용자 정의 액세스 메서드

소개

전통적인 관계형 데이터베이스 관리 시스템(Relational Database Management System; RDBMS)은 매우 제한된 종류의 단순한 데이터 형식(datatype)과 색인 액세스 메서드(Indexing access methods)만을 지원합니다. RDBMS로는 사용자가 자신의 고유한 데이터 형식을 정의하고 이러한 새로운 데이터 형식을 액세스하기 위한 자신의 색인 메서드를 구성할 수 없습니다. 이러한 한계 때문에, RDBMS 사용자는 복합적인 데이터 형식의 처리와 데이터 집약적인 처리를 데이터베이스 시스템 밖에서 해야만 합니다. 이런 이유로 데이터베이스 시스템은 단순 저장 장치로서의 역할 정도로만 인식될 수도 있으며, 관계형 데이터베이스에 저장된 실제 업무 데이터는 네트워크를 통해 클라이언트 측으로 옮겨져 처리되는 형식으로 사용되기 때문에 성능이 떨어질 수도 있습니다.

성능 측면에서 보면, 데이터가 저장된 장소와 가까운 곳에서 데이터 집약적인 처리 논리를 수행하는 것이 바람직합니다. 즉, 데이터베이스 내에서 직접 수행하는 것이 가장 바람직합니다. 객체 관계형 데이터베이스 관리 시스템(Object-Relational Database Management Systems; ORDBMS)의 출현으로 이 작업이 가능해졌습니다. RDBMS에서 제공하는 기능 외에도, ORDBMS는 사용자가 데이터 형식과 루틴(즉 데이터 형식을 처리하기 위한 저장 프로시저와 기능), 그리고 자신만의 액세스 메서드를 정의해서 데이터 형식을 효율적으로 저장하고 액세스할 수 있도록 해 줍니다. 이로써 눈에 띄는 성능 향상을 얻을 수 있습니다.

Informix Dynamic Server™(IDS)의 9.2 버전인 Informix® Internet Foundation™(Foundation)은 순수 ORDBMS 아키텍처를 가지고 있습니다. 사용자가 자신의 Informix DataBlade® 모듈을 만들 수 있게 해 줌으로써 RDBMS의 기능성과 ORDBMS의 유연성을 통합하였습니다. DataBlade 모듈은 ORDBMS에 플러그인되어 실행되는 소프트웨어 구성 요소처럼 작동함으로써 사용자 정의 데이터 형식(UDT), 사용자 정의 루틴(UDR), 데이터 형식을 색인하는 데 필요한 액세스 메서드를 포착합니다. 복합 데이터 형식을 관계형 테이블에 매핑해서 RDBMS로 하여금 데이터 형식에 관한 아무런 정보도 갖고 있지 않게 만드는 전통적인 클라이언트 기반의 솔루션과는 달리, Foundation은 UDT를 마치 내장 데이터 형식(INTEGER나 CHARACTER 등)인 것처럼 취급해서 데이터베이스 내에서 실행합니다. 복합 데이터를 처리하는 응용 프로그램에서는 이러한 기능을 이용하여 성능을 향상할 수 있습니다.

이 글은 Foundation에서 제공하는 객체 관계형 기능을 다음과 같이 세 개의 핵심 영역으로 나누어서 설명합니다.

- 사용자 정의 형식
- 사용자 정의 루틴
- 사용자 정의 액세스 메서드

사용자 정의 형식(User-Defined Types)

그림 1은 Foundation에서 지원하는 여러 데이터 형식을 도표로 나타낸 것입니다. 이 단원에서는 다양한 UDT의 특성과 중요성에 대해 자세히 설명합니다.

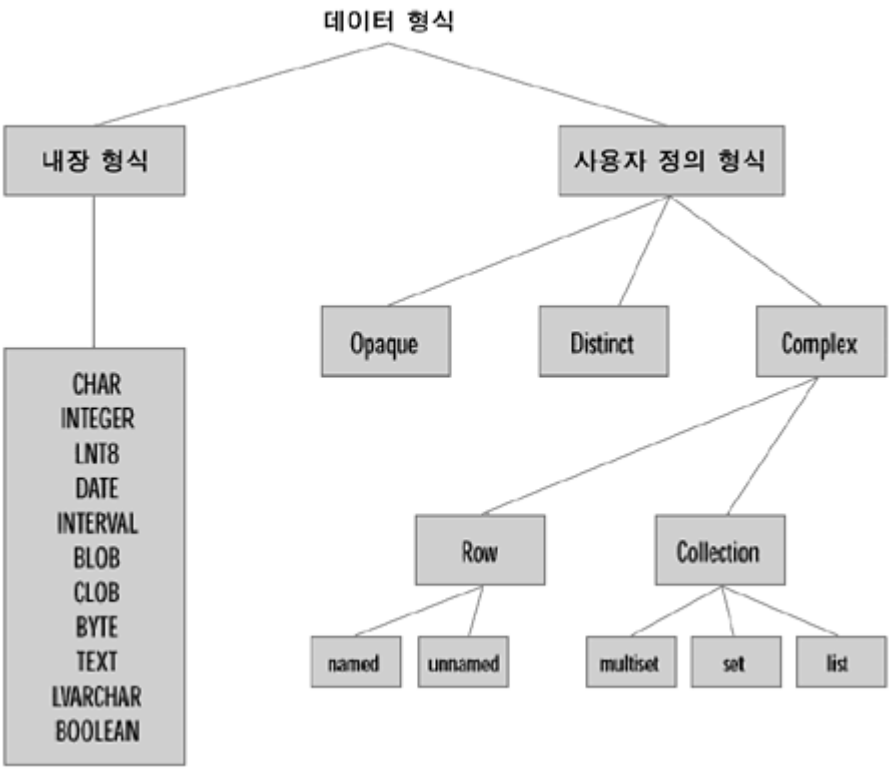

그림 1: Foundation 에서 지원하는 데이터 형식

복합 데이터 형식(Complex Datatypes)

복합 데이터 형식은 row와 collection의 두 범주로 나누어집니다. row 형식은 C 구조나 C++의 클래스와 유사한데, 이는 관련된 데이터 항목을 그룹화하고 이러한 그룹을 이름으로 참조합니다. 데이터 항목들이 논리적으로 연관되어 있으며 또한 자주 액세스될 경우, 점 표기법(dot notation)을 통해 개별적인 요소를 액세스할 수 있도록 이러한 관련 항목들로 구성된 row 형식을 만들면 매우 효율적이고 관리하기도 편리합니다. row 형식이 "create row type"이라는 문장으로 명백하게 생성될 때, 이를 명명된 row 형식(named row type)이라고 합니다. row 형식이 명백하게 생성되지는 않고 "row(int, char(50)" 같은 일반적인 row 형식 구성자를 사용해서 참조되면 이는 명명되지 않은 row 형식 (unnamed row type)입니다.

row 형식이 생성되면 데이터베이스는 그 row 형식 안에 있는 요소에 대해 완전한 지식을 갖게 됩니다. row 형식 안의 개별적인 요소들은 서로 다른 데이터 형식을 갖는 경우가 종종 있습니다. 이는 동일한 데이터 형식의 요소를 그룹화하는 collection 형식과 대조적입니다. collection 데이터 형식에는 세 종류가 있습니다. set, list, multiset이 그것입니다. Set collection 형식은 중복된 요소를 허용하지 않습니다. List collection 형식은 중복 가능한 요소를 순서대로 나열합니다. Multiset collection 형식은 중복된 요소를 허용한다는 점을 제외하면 set collection 형식과 동일합니다.

다음은 row와 collection 형식을 사용해서 테이블과 함께 "dimension" row 형식을 생성하는 예입니다.

```
create row type dimension_t

( length decimal, width decimal, height decimal );
```

```
create table part

(part_id serial, part_name varchar(30),

part_dimension dimension_t,

part_name_alias set(char(30)));
```

형식과 테이블 상속

명명된 row 형식은 객체 지향적 프로그래밍에서 나온 개념인 상속(inheritance)을 지원합니다. 예를 들면, 사용자는 "person" row 형식, person row 형식을 상속하는 "employee" row 형식, employee row 형식을 상속하는 "manager" row 형식을 생성할 수 있습니다. 그런 다음 사용자는 이 row 형식 계층에 대하여 형식이 지정된 테

이블 계층을 생성할 수 있습니다.

```
create row type person_t (name name_t, sex char(1),
   birthdate date, residence address_t);



create row type employee_t
   (employer varchar(30), salary decimal)
   under person_t;



-- employee_t 는 person_t 의 하위 형식입니다.



create row type manager_t (departmentManaged char(10)) under
   employee_t;
```

-- manager_t 는 employee_t 의 하위 형식입니다.

create table person_tab of type person_t;

create table employee_tab of type employee_t under person_tab;

-- employee_tab 은 person_t 의 하위 형식입니다.

create table manager_tab of type manager_t under employee_tab;

-- manager_tab 은 employee_tab 의 하위 형식입니다.

이 기능을 이용하면 Foundation 내에서 클래스 상속 계층을 명명된 row 형식 계층으로 매핑할 수 있습니다. 그런 다음 이 클래스의 인스턴스를 형지정된 테이블에 저장할 수 있습니다. 더 나아가, 테이블 계층 내에서 하위 테이블은 상위 테이블로부터 모든 컬럼, 제한 조건, 색인, 트리거를 상속 받음으로써 테이블 관리 오버헤드를 효과적으로 감소시키고 스키마 구성 요소를 지속적으로 재사용할 수 있습니다.

은폐 데이터 형식(Opaque Datatypes)

복합 데이터 형식과는 대조적으로, 데이터베이스에 C 구조를 직접 저장할 수 있는 은폐(opaque) 데이터 형식을 생성할 수 있습니다. 은폐 형식을 사용하면 Foundation은 구조의 내용을 해석하지 않고 단지 데이터베이스에 구조의 내용을 저장하기만 합니다. 이 때문에 은폐 형식이란 이름이 붙여졌습니다. 은폐 형식을 사용함으로써 얻게 되는 이점은 사용자가 C 구조 각 구성원의 데이터 형식을 SQL 데이터 형식에 직접 매핑할 필요가 없다는 것입니다. 은폐 형식은 사용자가 작성한 루틴을 통해 액세스됩니다. 사용자는 은폐 형식의 저장 크기를 정의하고 데이터 형식을 처리하는 데 필요한 모든 지원 루틴을 작성합니다.

다음은 "circle" 은폐 형식을 생성하는 예로서, circle 데이터 형식을 처리하기 위한 모든 UDR을 정의합니다. circle은 Spatial DataBlade 모듈에서 사용하는 데이터 형식입니다. circle 데이터 형식은 C++ 클래스와 유사하며 circle 데이터 형식을 처리하는 UDR은 C++ 클래스에 대해 정의된 메서드와 유사하다고 이해하면 됩니다.

다음은 C로 circle 데이터 형식을 정의한 것입니다.

```
typedef struct
```

```
{
        double x;
        double y;
        double radius
} circle;
```

다음은 Foundation에서 circle 데이터 형식과 이를 지원하도록 하는 UDR을 등록하기 위한 SQL 코드입니다.

```
create opaque type circle (internallength = 24, alignment = 8);

create function circle_input(lvarchar) returning circle
   with (not variant, parallelizable)
   external name "$USERFUNCDIR/circle_datablade.udr(circle_input)"
   language c;
```

```
create implicit cast (lvarchar as circle with circle_input);
```

circle_input 함수를 이용하면 SQL문에서 input으로 circle 데이터 형식을 생성할 수 있으며, Foundation에서는 형변환 생성(cast creation)을 통해 SQL에서 circle 데이터 형식으로 암시적인 변환을 할 수 있습니다.

```
create function circle_output(circle) returning lvarchar with

(not variant, parallelizable)

external name "$USERFUNCDIR/circle_datablade.udr(circle_output)"

language c;

create explicit cast (circle as lvarchar with circle_output);
```

circle_output 함수는 Foundation으로 하여금 circle 데이터 형식을 SQL문으로 변환하도록 합니다.

```
create function equal(circle, circle) returning boolean with
```

(not variant, parallelizable)

external name "$USERFUNCDIR/circle_datablade.udr(circle_eq)"

language c;

일단 equal 함수가 정의되면, Foundation은 두 circle이 동일한지 비교하기 위해 "=" SQL 연산자를 사용해야 할 때 이 함수를 호출할 수 있습니다. 이와 마찬가지로, SQL문의 circle 데이터 형식에서도 !=, >=, >, <=, < SQL 연산자를 사용할 수 있도록 notequal, greaterthanorequal, greaterthan, lessthanorequal, lessthan 함수를 정의해야 합니다.

create function getArea(circle) returning double with

(not variant, parallelizable)

external name "$USERFUNCDIR/circle_datablade.udr(getArea)"

language c;

일단 getArea 함수가 정의되면, Foundation은 circle 컬럼에서 getArea()를 실행하는 SQL 표현식을 평가할 수 있게 됩니다.

다음은 이러한 함수를 모두 호출하는 SQL문입니다.

create table circle_demo(id SERIAL, cir circle);

```
select id, cir, getArea(cir)

from circle_demo

where cir > "3 4 6.78";
```

위의 질의에서 Foundation은 circle_input(lvarchar) 함수를 호출하여 SQL 문자열 "3 4 6.78" 을 circle 데이터 형식으로 변환합니다. 그런 다음 circle_demo 테이블 안의 각 행에 대한 ">" 연산자를 평가하기 위해 greaterthan(circle, circle) 함수를 호출합니다. Foundation은 ">" 조건을 만족시키는 각 행에 대해 getArea(circle) 함수를 호출하여 circle 영역을 계산한 다음, circle_output(circle) 함수를 호출하여 circle을 SQL 문자열로 변환합니다.

은폐 데이터 형식을 생성하는 것은 row 형식을 생성하는 것보다 시간이 더 오래 걸리는 작업이긴 하지만, C 구조의 각 필드를 row 형식의 각 필드에 매핑하지 않고도 데이터 형식의 C 구조를 데이터베이스에 직접 저장할 수 있다는 장점이 있습니다.

사용자 정의 은폐 데이터 형식은 Foundation이 도메인 특정(domain-specific) 데이터를 즉시 처리할 수 있도록, 도메인 특정 데이터를 처리하는 상속(legacy) C/C++ 라이브러리를 Informix DataBlade 모듈로 변환할 경우 유용합니다. 사실상 상당 수의 Informix DataBlade 모듈이 각자의 형식 시스템을 생성하기 위해 사용자 정의 은폐 데이터 형식을 사용합니다.

개방 데이터 형식(Distinct Datatypes)

개방(distinct) 데이터 형식은 사용자로 하여금 기존의 데이터 형식을 기반으로 새로운 데이터 형식을 생성할 수 있게 해 줍니다. 기존의 데이터 형식은 내장 형식, 사용자 정의 은폐 형식, 명명된 row 형식, 심지어는 또 다른 개방 형식이 될 수도 있습니다. 기존 형식에 대해 정의된 모든 UDR은 새로운 개방 데이터 형식에 대해서도 사용이 가능합니다.

다음은 "decimal" 내장 데이터 형식을 기반으로 한 "inches" 개방 데이터 형식을 정의하는 예입니다. inches라고 명명된 새로운 구별 형식은 기존의 decimal 형식과 동일한 표현을 이용해 생성됩니다. 그러나 inches 형식과 decimal 형식 중 어느 하나라도 명시적으로 형변환되지 않으면 더하거나, 빼거나, 비교할 수가 없습니다.

```
create distinct type inches as decimal;
```

```
create table items(item_id serial, length inches);
```

decimal 상수가 inches 데이터 형식으로 명시적 형변환이 되어야

한다는 점에 유의하십시오.

```
insert into items(length) values (3.3::inches);
```

```
insert into items(length) values)(3.4::inches);
```

```
select * from items where length = 3.4::inches;
```

사용자 정의 루틴(User-Defined Routines)

"사용자 정의 루틴"은 사용자 정의 프로시저 와 사용자 정의 함수를 일컫는 SQL3 용어입니다. 프로시저와 함수의 차이점은 함수가 값을 반환할 수 있는데 반해, 프로시저는 그럴 수 없다는 점입니다. Foundation은 사용자가 Informix SPL(Stored Procedure Language) 뿐만 아니라 C, C++, Java로 UDR을 작성할 수 있게 해 줍니다. 전통적인 RDBMS에서는 UDR을 "내장 프로시저"라고 합니다.

그림 2는 Foundation에서 지원하는 루틴을 도표로 보여 줍니다.

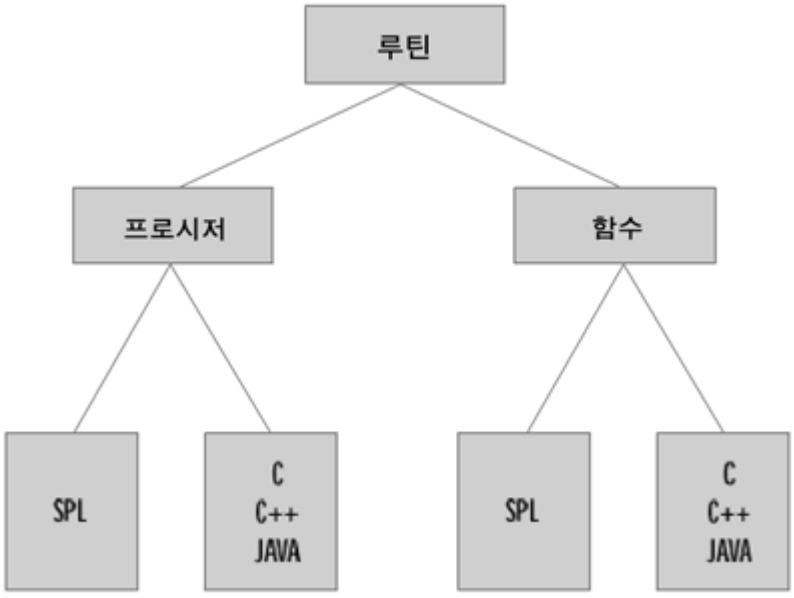

그림 2: Foundation 에서 지원하는 루틴

Foundation 에서는 SPL 루틴에서도 UDT 를 사용할 수 있습니다. 예를 들면, volume SPL 함수를 써서 dimension_t row 형식의 볼륨을 계산할 수 있습니다.

```
create function volume (dimension_t d) returning decimal

    return d.length * d.width * d.height

end function;
```

사용자 정의 루틴 사용

SQL 집약적인 사용자 정의 루틴

UDR의 장점은 질의의 프로시저 논리와 SQL 질의 액세스 논리를 결합할 수 있다는 점입니다. 이로써 클라이언트가 특정 질의를 제출하고 클라이언트 응용 프로그램 내에서 프로시저 논리를 실행할 때 전보다 훨씬 빠른 성능을 보여 줍니다. 그 이유는 객체 관계형 데이터베이스가 일단 서버에 생성된 UDR을 마치 시스템에 정의된 내장 함수인 것처럼 관리하기 때문입니다. Foundation은 데이터베이스 내에서 직접 UDR을 작동시키며 네트워크 프로토콜을 통하지 않고 SQL 질의를 처리합니다. 또한 서버는 UDR에 정의된 SQL 질의에 대한 질의 계획과 내부 데이터 구조를 내부적으로 캐시 처리합니다.

다음은 SPL로 작성된 UDR 내에서 질의를 실행하는 예입니다.

```
create function raise_salary_by_point(point integer)returning integer

    if (point < 100) then
```

```
update employee

set salary = salary + salary * point /100;

return 1;

    else

return 0;

    end if;

end function;
```

다음은 hr 클래스에 대해 정적 메서드인 raise_salary_by_point를 정의하는 실제 Java 코드입니다.

```
import java.sql.*;

import java.lang.*;

import java.util.*;

public class hr {

public static int raise_salary_by_point(int point) throws
```

```
SQLException {

if (point >= 100)

{

return 0;

}

   try {

Connection conn =

                    DriverManager.getConnection("jdbc:Informix direct:

//ConnectionClass=IfxDirectConnection;




    //ProtocolClass=

IfxDirectProtocol");

Statement stmt = conn.prepareStatement();

int rs =
```

```
stmt.executeUpdate("update employee set salary = salary +

salary * point /100;");

stmt.close();

conn.close();

return 1;

}



catch (Exception e) {

throw new SQLException(e.toString());

}

}

}
```

JDBC 연결에 사용된 URL에는 IfxDirectConnection과 IfxDirectProtocol이 포함되어 있다는 점에 유의하십시오. 이런 식으로 JDBC를 연결하는 것을 서버 측 JDBC 연결이라고 부르는데, 이는 전형적인 클라이언트 측

JDBC 연결과 비교하면 매우 가벼운 연결입니다. 서버 측 JDBC 연결은 데이터베이스 안에서 이루어지며 이러한 연결시 네트워크 오버헤드는 발생하지 않습니다.

UDT에 대한 지원 루틴

UDT에 대한 지원 루틴은 C++ 클래스에 대한 메서드와 유사합니다. 은폐 데이터 형식 단원에서 볼 수 있는 getArea(circle) 함수는 "circle" 은폐 데이터 형식에 대한 지원 루틴의 예입니다.

데이터 형식 간의 Cast 함수

은폐 데이터 형식 단원에서 설명한 circle_input(lvarchar) 함수는 lvarchar 데이터 형식을 circle 데이터 형식으로 변환시키는 cast 함수의 예입니다.

사용자 정의 액세스 메서드와 함수 색인에 대한 지원 루틴

이 내용은 사용자 정의 액세스 메서드 단원에서 자세하게 설명할 것입니다.

SQL 표현식에 사용된 사용자 정의 함수

사용자 정의 함수는 마치 시스템의 내장 함수인 것처럼 SQL 선택 목록, WHERE 절, GROUP BY, HAVING 절의 표현식으로 쓰일 수 있습니다. 예를 들면, Java로 만든 isActiveUrl 함수를 정의하여 URL 주소를 이용해 유효한 주소인지, 사이트가 활성 상태인지를 테스트할 수 있습니다.

```
create function isValidUrl(address lvarchar) returning boolean

with (class="jvp")

external name "sqlnet.isValidUrl"
```

Language java;

create table urlTrack(urlid serial, url lvarchar);

insert into urlTrack (url) values("http://www.Informix.com");

insert into urlTrack (url) values("http://www.yahoo.com");

다음 질의를 실행하여 유효하고 활성화된 주소를 모두 찾을 수 있습니다.

select * from urlTrack where isValidUrl(url);

사용자 정의 함수가 질의의 WHERE 절에 사용되었다는 점에 유의하십시오.

함수 오버로딩

Foundation은 함수 오버로딩을 지원합니다. 이는 객체 지향적 프로그래밍에서 상당히 중요한 개념입니다. 다시 말해, 이름은 같지만 서명(signature)은 다른 루틴을 정의할 수 있습니다. 루틴 서명은 루틴 이름, 매개변수의

수, 매개변수의 데이터 형식으로 구성됩니다. 함수 오버로딩은 ORDBMS에서 매우 중요한데, 이는 사용자들로 하여금 동일한 이름으로 서로 다른 데이터 형식에서 작동하는 함수를 많이 생성하도록 해주기 때문입니다. 예를 들면, 하나는 circle 데이터 형식에서 작동하고 다른 하나는 polygon 데이터 형식에서 작동하는 두 개의 "equal" 함수를 정의할 수 있습니다.

```
create function equal(circle, circle) returning boolean with

(not variant, parallelizable)

external name "$USERFUNCDIR/circle_datablade.udr(circle_eq)" language c;
```

```
create function equal(polygon, polygon) returning boolean with

(not variant, parallelizable)

external name "$USERFUNCDIR/poly_datablade.udr(poly_eq)" language c;
```

사용자 정의 액세스 메서드(User-Defined Access Methods)

사용자 정의 액세스 메서드는 두 개의 범주로 나누어집니다. 기본(primary) 액세스 메서드와 보조(secondary) 액세스 메서드가 그것입니다. 가상 테이블 인터페이스(Virtual Table Interface: VTI)라고도 하는 기본 액세스 메서드는 Informix 데이터베이스 외부에 있는 외부 데이터를 데이터베이스 내부에 있는 가상의 테이블인 것처럼 실체화할 수 있도록 해 줍니다. 일반적으로 이러한 인터페이스는 Informix 이외의 데이터를 Informix의 가상 테이블로 보이도록 하는 게이트웨이 형식의 액세스 메서드를 설계하고 개발하는 데 사용됩니다. 또한 이 인터페이스를 사용해서 관계형이 아닌 데이터를 관계형 테이블처럼 보이도록 할 수 있습니다.

가상 인덱스 인터페이스(Virtual Index Interface: VII)라고도 하는 보조 액세스 메서드는 도메인-데이터, 텍스트 데이터, 이미지 데이터 등과 같은 특정 데이터에 대해 색인을 구축할 수 있게 해 줍니다. 두 액세스 메서드 모두 가상의 테이블과 가상의 색인을 구축하는 메서드, 가상의 테이블과 가상의 색인에서 행을 스캔하고 삽입하고 갱신하고 삭제하는 메서드 등과 같이, 테이블과 색인 연산을 처리하는 일련의 UDR에 의해 정의됩니다.

그림 3은 Foundation에서 지원하는 액세스 메서드입니다.

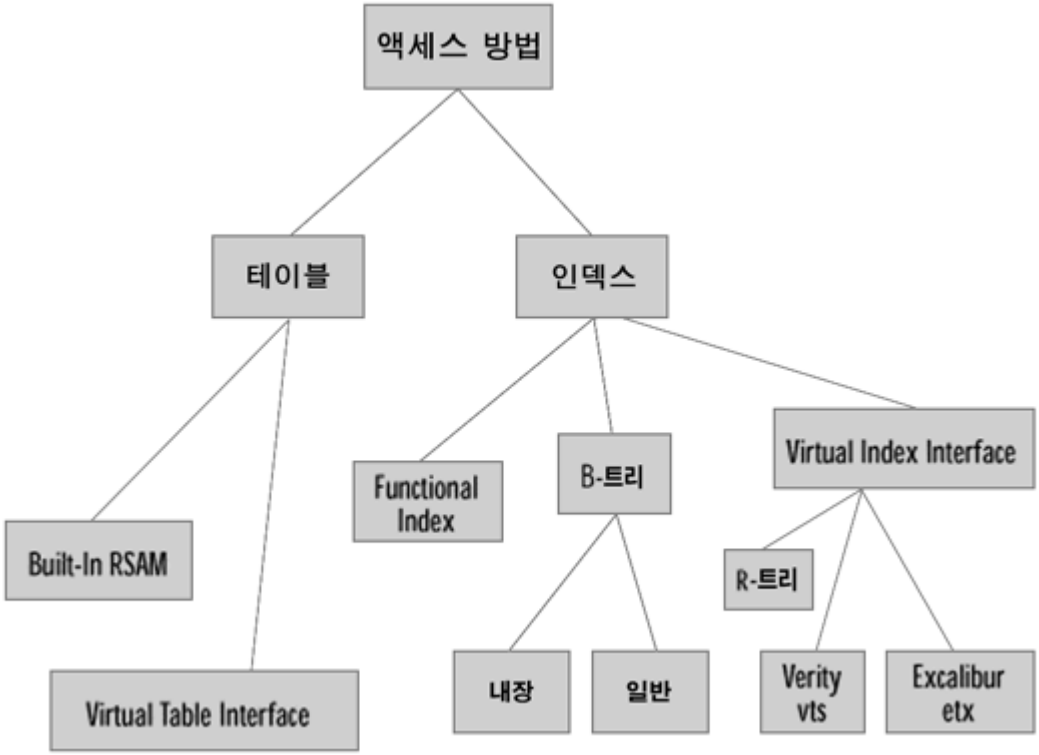

그림 3: Foundation 에서 지원하는 액세스 메서드

색인 액세스 메서드

색인은 책에서 사용하는 것과 마찬가지로, 행을 빨리 찾기 위해 사용합니다. 색인 액세스 메서드는 연산자 클래스와 연관되는데, 이는 SQL 옵티마이저가 SQL 서술절(WHERE 절)에서 인식할 수 있는 일련의 전략 함수와, 색

인 액세스 메서드가 내부적으로 사용하는 일련의 지원 함수를 정의합니다. 전통적인 RDBMS에는, B-트리 색인이라는 단 하나의 색인 액세스 메서드만이 있을 뿐입니다. B-트리 색인 메서드는 INTEGER 와 CHARACTER 같은 내장 형식에서만 생성할 수 있습니다. 그리고 대부분의 RDBMS에서는 BLOB 컬럼을 생성해서 텍스트 문서와 이미지 사진을 저장할 수 있지만, 텍스트나 이미지 기반의 형식에 대해서는 색인을 생성할 수 없습니다. 그 결과 텍스트와 이미지 컬럼에 대한 검색 질의 속도가 느려집니다.

Foundation에서는 다양한 데이터 형식에 대한 다양한 색인 액세스 메서드를 생성할 수 있습니다. 예를 들면, 사용자는 B-트리 색인 메서드를 생성하여 자신만의 은폐 데이터 형식을 색인화할 수 있습니다. 이를 일반적인 B-트리 색인이라고 부릅니다. 또한 공간적인(spatial) 데이터 형식에 대해 R-트리 색인을 생성할 수도 있습니다. 텍스트, 이미지, 기타 다양한 데이터 형식에 대한 색인 뿐 아니라 함수 색인(함수 실행 결과에 기초한 색인)까지도 생성할 수 있습니다. 더 나아가 VII에서는 색인 액세스 메서드를 생성해서 DataBlade 모듈에 정의된 도메인 특정 데이터 형식을 색인화할 수 있습니다. 사실상 R-트리 색인, 텍스트 기반 색인, 이미지 기반 색인들은 DataBlade 모듈, 즉 Informix Spatial DataBlade 모듈, Excalibur와 Verity Text Search DataBlade 모듈, Excalibur Image DataBlade 모듈로 등록된 가상 색인 액세스 메서드에 지나지 않습니다. 가상의 색인이 정의되면, SQL 옵티마이저는 이러한 데이터 형식 상의 질의에 대한 테이블 스캔 대신 색인 스캔을 사용할 수 있습니다.

일반적인 B-트리 색인

RDBMS에서 B-트리 색인 액세스 메서드는 내장 데이터 형식만을 처리합니다. Foundation에서는 UDT를 지원하기 위해 B-트리 색인의 확장 버전인 일반적인 B-트리 색인을 제공합니다. 기본 연산자 클래스와 연결된 B-트리 색인 액세스 메서드에 대한 전략 함수는 lessthan, lessthanorequal, equal, greaterthan, greaterthanorequal입니다. 지원 함수는 compare입니다. <, <=, =, >, >= 같은 절을 포함하는 질의는 매우 효율적으로 처리되는데, 이는 SQL 옵티마이저가 이들을 B-트리 색인에 대한 전략 함수로 인식해서 B-트리 색인 스캔을 사용하기 때문입니다. 사용자는 일반적인 B-트리 색인을 생성함으로써 자신의 UDT에서 이것을 이용할 수 있습니다.

예를 들면, 사용자는 앞에서 설명한 circle 은폐 데이터 형식에 대해 일반적인 B-트리 색인을 생성할 수 있습니다. circle 컬럼에서 이러한 일반적인 B-트리 색인을 생성하려면 circle 데이터 형식에서 다음 함수를 정의해야

합니다.

compare(circle, circle),

lessthan(circle, circle),

lessthanorequal(circle, circle),

equal(circle, circle),

greaterthan(circle, circle), and

greaterthanorequal(circle, circle).

다음은 위에서 정의된 circle_demo 테이블에 대한 circle 컬럼에서 일반적인 B-트리 색인을 생성하는 예입니다.

```
create index on circle_demo(cir)
```

-- 일반적인 B-트리 색인은 circle 컬럼에서 생성됩니다.

이제 SQL 옵티마이저는 다음 질의에 대해 일반적인 B-트리 인덱스 스캔을 참조할 수 있습니다.

```
select id, cir, getArea(cir)
```

```
from circle_demo

where cir > "3 4 6.78";
```

함수 색인(Functional Index)

함수 색인이 생성되면, Foundation은 특정 함수를 실행하고 색인에 출력 값을 키 값으로 저장합니다. 함수 색인은 테이블의 컬럼에서 파생된 값을 색인화하는 데 유용합니다.

다음은 함수 색인을 생성하는 예입니다.

먼저 "sales" 테이블을 생성합니다.

```
create table sales

(itemid serial, item_sold_quantity int, item_sold_price decimal);
```

이제 sales에 대한 함수 색인을 생성할 수 있습니다.

```
create index item_revenue on

sales(revenue(item_sold_price, item_sold_quantity));
```

결과적으로, 다음 질의에 대해 SQL 옵티마이저는 함수 색인 스캔 사용을 고려할 수 있습니다. 함수 색인이 생성되지 않으면 데이터베이스는 테이블의 모든 행을 스캔하고 각 행에 대한 revenue 값을 계산해야 합니다.

```
select * from sales

where revenue(item_sold_price, item_sold_quantity) > 2000.28
```

R-트리 인덱스

R-트리 색인 메서드는 Spatial 데이터 형식과 같은 다차원 데이터에 대한 액세스 속도를 높이기 위해 사용됩니다. 이 메서드는 질의를 통해 다른 객체 안에 있는 객체, 다른 객체를 포함한 객체, 다른 객체와 상호 작용하고 중첩되는 객체를 찾을 때 가장 유용하게 사용됩니다. 이런 이유로 기본 연산자 클래스와 연결된 R-트리 색인 액세스 메서드에 대한 전략 함수는 contains, overlap, within, equal 함수입니다. 맵에서의 특정 위치는 R-트리 색인을 구축하는 데 더할 나위 없이 좋은 이차원 데이터 요소입니다.

다음은 Informix Spatial DataBlade 모듈과 함께 R-트리 색인을 사용하는 예입니다.

먼저 "restaurant" 테이블을 생성하고 restaurant 위치를

이차원 circle 형식 SP2CIRC 로 저장합니다.

```
create table restaurant (id serial, name char(20), location SP2CIRC);
```

SQL 옵티마이저는 테이블에 행을 로드하고 통계 수치를 업데이트한 후에, R-트리 색인을 이용해서 근원지로부터 5 마일 내에 있는 모든 레스토랑을 찾는 질의를 처리할 수 있습니다.

```
select * from restaurant

where contains( '(0, 0, 5)'::SP2CIRC,location);
```

Excalibur와 Verity 텍스트 색인 액세스 메서드

Excalibur 와 Verity 색인 액세스 메서드는 HTML, PDF, Microsoft Word, Adobe같은 다양한 텍스트 기반의 문서에 대한 색인을 생성하기 위해 사용할 수 있습니다. Excalibur 텍스트 색인 액세스 메서드는 Informix Excalibur Text DataBlade 모듈에서 정의됩니다. Excalibur 텍스트 색인 메서드는 etx라고 하며 이의 전략 함수는 etx_contains입니다. Verity 텍스트 색인 액세스 메서드는 Informix Verity Text DataBlade 모듈에서 정의됩니다. Verity 텍스트 색인 메서드는 vts라고 하며 이의 전략 함수는 vts_contains입니다.

다음은 Excalibur 색인 메서드를 사용하는 예입니다.

```
create table docTab ( id serial, desc ifxdocdesc);
```

텍스트 문서에 Excalibur 색인을 생성합니다.

```
create index etx_idx on docTab(desc)

   using etx(WORD_SUPPORT = 'PATTERN',
```

```
PHRASE_SUPPORT = 'MINIMUM', CHAR_SET = 'ASCII');
```

SQL 옵티마이저는 Excalibur 색인 etx_idx를 사용하여 "help"라는 단어를 포함하고 있는 모든 문서를 검색하는 질의를 처리할 수 있습니다.

```
select id from docTab where etx_contains(desc, 'help');
```

또는 텍스트 문서에 Verity 색인을 생성할 수 있습니다.

```
create index vts_idx on docTab(desc) using vts(...);
```

SQL 옵티마이저는 Verity 색인 vts_idx을 사용하여 "help"라는 단어를 포함하고 있는 모든 문서를 검색하는 질의를 처리할 수 있습니다.

```
select id from docTab where vts_contains(desc, 'help');
```

맺음말

Informix Internet Foundation은 관계형 모델의 단점을 보완하는 한편 핵심적인 객체 지향적 개념의 장점을 이용하여 강력한 객체 관계형 데이터베이스 관리 시스템을 제공합니다. Foundation은 특히 RDBMS의 형식 시스템을 확장해서 복합 데이터 형식을 효율적으로 저장하고 처리할 수 있도록 해 주고, 응용 프로그램 작성자에게 데이터베이스 측면에서 다층(multi-tier) 응용 프로그램의 성능을 높일 수 있는 새로운 기회를 제공해 줍니다. 이러한 것은 이 글에서 설명한 다양한 UDT, UDR, 사용자 정의 액세스 메서드에 대한 지원을 통해 이루어집니다. 도메인 내의 데이터를 처리하는 DataBlade 모듈을 지원함으로써 도메인 특정(domain-specific) 데이터를 처리할 수 있는 틀을 제공하므로, Foundation은 어떤 특수한 데이터 시장에서도 성공으로 향하는 최선의 해결책이라는 사실이 증명되고 있습니다. 결론적으로 Foundation은 전통적인 RDBMS에 비해 더욱 다양한 기능과 뛰어난 성능을 제공함으로써 데이터베이스 기술의 새로운 물결을 일으키고 있습니다.