Tech Notes (vol.8)

Informix Database 네트워킹

소개

오늘날의 분산 컴퓨팅 환경에서는 여러 컴퓨터와 데이터베이스 서버를 서로 연결하는 능력이 필수적이며, 이러한 능력은 적절한 네트워킹 프로토콜을 사용함으로써 가능합니다. 다른 서버의 데이터베이스에 액세스하는 소프트웨어라면 반드시 네트워크 연결을 처리할 수 있어야 합니다.

Informix에서는 이러한 네트워킹을 가능하게 하는 다양한 방법을 제공합니다. 이 글에서는 다음과 같은 내용을 소개할 것입니다.

오늘날의 클라이언트/서버 환경에서 네트워킹이 필요한 이유

네트워킹은 서로 다른 서버에 있는 데이터베이스가 하나로서 작동하도록 합니다. 또한 서로 다른 운영 체제, 하드웨어 및 연결이 상호 원활하게 작동하도록 합니다.

Informix 네트워킹 기능의 구성과 사용법

Informix 사용자는 다른 서버의 데이터베이스에 투명하게 액세스할 수 있습니다. Informix용으로 관리해야 하는 서로 다른 데이터베이스 연결 및 파일 유형이 있습니다.

운영 체제용 네트워킹 파일 구성법

원격 서버에 액세스하려면 운영 체제의 네트워킹 파일들을 Informix에 맞게 적절히 업데이트 해야 합니다. 이 파일들은 네트워크를 통해 통신이 이루어지는 방법을 정의합니다.

Informix 구버전의 네트워킹 방법

Informix에서는 모든 버전에서 데이터베이스의 네트워킹 방법을 제공해 왔습니다. Informix OnLine과 Informix SE를 비롯한 Informix 구버전의 네트워킹 방법을 소개합니다.

클라이언트/서버 컴퓨팅과 네트워킹 이해하기

분산 컴퓨팅은 컴퓨터 네트워크를 최대한 활용합니다. 예를 들면, 하나 이상의 서버에 데이터베이스를 상주시키고 여기에 액세스할 프로그램은 사용자의 PC나 다른 서버에서 실행할 수 있습니다. 이것이 바로 클라이언트/서버 컴퓨팅입니다. Informix에서는 데이터베이스와 응용 프로그램을 다양한 방법으로 분산하여 사용할 수 있습니다. 이 글에서 그러한 여러 방법을 소개할 것입니다. 그림 1에서는 관계형 데이터베이스의 힘을 확장하기 위해 어떻게 네트워킹을 사용할 수 있는지를 보여 줍니다.

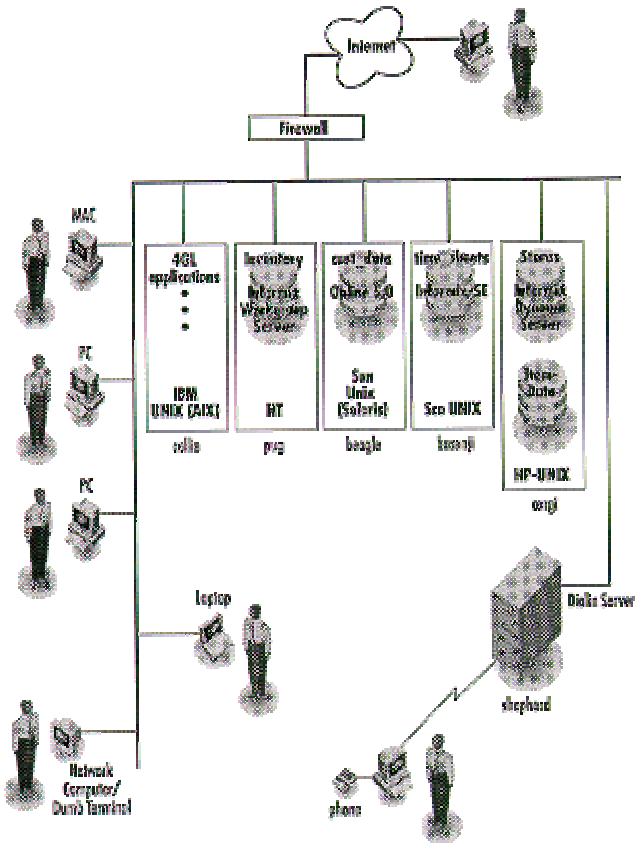

그림 1. 네트워크로 연결된 데이터베이스의 힘

클라이언트/서버의 힘이 우리의 업무 방식을 어떻게 향상시켜 주는지를 이 예제에서 잘 알 수 있습니다. 몇 가지 보기를 들면 다음과 같습니다.

- stores 데이터베이스는 corgi 서버에 상주시키고, 이것의 일부 데이터는 pug, beagle, 및 basenji를 비롯한 다른 데이터베이스와 서버에 저장합니다.
- 데이터베이스 서버를 서로 다른 UNIX 및 Windows NT 시스템의 조합으로 구성합니다. UNIX 운영 체제에는 AIX, Solaris, SCO 및 HP-UX 등이 있습니다. 서로 다른 하드웨어 플랫폼(IBM, Sun, HP 등)에서 서로 다른 운영 체제를 실행합니다.
- 어떤 사용자는 PC와 Macintosh에서 GUI 기반의 응용 프로그램을 실행하고, 또 어떤 사용자는 문자 기반의 4GL 응용 프로그램에 액세스합니다. 문자 기반의 응용 프로그램은 모든 단순 단말기(네트워크 컴퓨터 포함)에서 또는 텔넷이나 기타 직접 연결 방식을 사용하는 모든 PC에서 실행할 수 있습니다.
- 사용자들은 네트워크, 전화선 및 인터넷을 비롯한 여러 다양한 방법을 사용하여 데이터베이스에 접속합니다.
- 터미널에는 PC, 노트북 컴퓨터 및 네트워크 컴퓨터를 비롯하여 다양한 종류가 있습니다.
- Workgroup Edition of Informix Dynamic Server, OnLine 5.0, Informix SE, 및 Informix Dynamic

Server를 비롯하여 여러 Informix 버전을 실행합니다.

Informix는 여러 방법으로 데이터베이스를 분산합니다. 서로 다른 여러 서버에 분산되어 있는 테이블에 액세스하는 하나의 데이터베이스를 사용할 수 있습니다. 이렇게 하면 처리 능력이 분산되므로 성능이 크게 향상됩니다. 또한 하나 이상의 부서에서 특정 데이터를 필요로 할 경우에 그것을 별도의 서버에 두기가 더 쉬울 것입니다. 예를 들면 그림 1에서 pug의 inventory 데이터베이스에 있는 데이터는 stores의 데이터와 분리되어 있지만 stores에서 액세스할 수 있습니다. 사실상 적절히 액세스하기만 한다면 네트워크의 어디에서도 inventory에 도달할 수 있습니다.

네트워킹 소프트웨어는 클라이언트(예를 들면 PC)와 다른 서버에서 실행할 수 있습니다. 이 소프트웨어는 특정 네트워크 프로토콜 언어(예를 들면 TCP/IP)를 이해하므로 다른 컴퓨터들과 통신할 수 있습니다. 클라이언트측에서는 데이터베이스 서버의 제품과 통신할 수 있어야 합니다.

(주의)

전통적으로 클라이언트라고 하면 PC를 떠올립니다. 그러나 꼭 그런 것만은 아닙니다. 클라이언트란 단순히 응용 프로그램을 실행하는 컴퓨터를 말합니다. 예를 들어, 클라이언트 응용 프로그램은 Informix 4GL로 작성하여 그림 1의 컴퓨터들 중 하나에서(corgi 포함) 실행할 수 있습니다. 사실 collie는 Informix 4GL 응용 프로그램을 실행하는 "응용 프로그램 서버"입니다. 물론 클라이언트는 Informix Dynamic 4GL과 같은 제품으로 개발한 응용 프로그램을 실행하는 PC일 수도 있습니다.

이제 네트워크를 사용하는 것은 매우 일반적인 일이기 때문에 대부분의 컴퓨터에는 네트워크를 사용하기 위해 필요한 소프트웨어를 이미 포함하고 있는 경우가 많습니다. 새로운 운영 체제는 일반적으로 TCP/IP 방식의 제품들을 포함합니다. Informix는 6.0 버전부터 모든 Informix 서버 제품에 네트워킹 소프트웨어를 포함하고 있습니다.

Informix에는 두 가지 유형의 네트워크 제품이 있습니다. PC에서 실행하는 것과 데이터베이스 서버에서 실행하는 것이 그것입니다. 그림 1에서 Mac과 PC용 소프트웨어는 데이터베이스 서버에서 사용되는 소프트웨어와 명백하게 다릅니다. 이 두 가지 유형의 소프트웨어에 대해서는 따로 설명을 할 것입니다.

물론 보안은 필수적입니다. 그림 1에서 collie, pug, beagle, basenji 그리고 corgi에 대한 로그인 보안은 운영 체제에 의해 강화되고 있습니다. 다이얼-인 서버인 shepherd는 자체의 인증 기능을 가지고 있습니다. 물론 방화벽은 들어오는(incoming) 인터넷 접속에 대한 보안 기능을 제공합니다. 데이터베이스 측면에서 볼 때 Informix는 여러 계층의 보안 기능(데이터베이스, 테이블, 칼럼 등)을 제공합니다. 다음 단원에서는 Informix와 운영 체제 파일들을 사용하여 Informix 네트워크 데이터베이스를 만드는 방법에 대해 설명할 것

입니다. 다음은 이 기능에 필요한 파일 목록을 요약한 것입니다.

- Informix sqlhosts 파일
- Informix onconfig 파일
- hosts 파일
- services 파일
- 네트워크 보안 파일: hosts.equiv, .rhosts, 그리고 .netrc

분산 데이터베이스에서 Informix SQL 사용하기

그림 1에서 볼 수 있듯이 corgi의 stores 데이터베이스는 실제로 pub, beagle 및 basenji의 데이터베이스에

InformixSERVER가 설정되어 있지 않다면 다음과 같이 표기해야 합니다.

```
select * from time_sheet@basenji_tcp:time_in;
```

2. SQL CONNECT 문을 사용한 이후.

Informix SQL의 CONNECT 문을 사용하여 InformixSERVER 변수의 값을 무시할 수 있습니다. 이것의 구문은 connect to dbname@dbservername인데, 여기에서 dbname은 선택적으로 사용할 수 있습니다. 이렇게 연결한 다음, 이어지는 SQL 문에서 dbservername의 데이터를 참조할 것입니다. 예를 들면 다음과 같습니다.

```
connect to @pug_tcp;

connect to inventory@pug_tcp;
```

3. 질의에서 명시하기.

질의에 "@"을 포함하면, Informix에서는 그 데이터베이스 서버에 연결하려고 시도합니다. 이런 질의 유형의 구문은 database@dbservername:table입니다. 예를 들면 다음과 같습니다.

```
select * from cust_data@beagle_tcp:customer;
```

4. SQL 동의어 사용하기.

Informix SQL을 이용하면 현재 서버 또는 다른 서버에 있는 데이터베이스 테이블에 대해 동의어(synonym)를 만들 수 있습니다. 이렇게 하면 사용자는 위의 예처럼 "@" 표기를 입력하지 않아도 되며 어떤 서버에 어떤 테이블이 포함되어 있는지 알 필요도 없습니다. 동의어 문은 create synonym synonym_name for table의 구문으로 생성할 수 있으며 다음의 SQL문이 그 예입니다. 이러한 동의어는 위 예의 어떤 서버에서도 실행할 수 있습니다.

```
create synonym customer for

cust_data@beagle_tcp:customer;
```

이것은 동의어를 만든 서버로부터 단순한 "select * from customer"를 실행하도록 해 줍니다.

방법이 작동하도록 하려면 해당 운영 체제 파일이 적절히 구성되어 있어야 합니다. 이 방법은 뒤에 나오는 " 보안 파일 사용하기" 단원에서 소개합니다.

Informix 네트워킹 이해하기

네트워킹은 여러 컴퓨터가 공통의 프로토콜을 사용하여 통신할 수 있도록 해 줍니다. 가장 일반적인 프로토콜은 수 년 전에 개발되어 현재 인터넷 통신의 기초가 된 TCP/IP(Transmission Control Protocol/ Internet Protocol)입니다. 요점을 말하면, TCP/IP와 같은 네트워킹 프로토콜은 데이터 "패킷"의 구성체를 정의하는 표준을 제공합니다. 이렇게 하여 네트워크의 다른 컴퓨터에서 그곳으로 보내진 데이터를 이해할 수 있게 됩니다. 또한 TCP/IP와 같은 여러 프로토콜은 모든 데이터가 적절히 보내질 것임을 확신할 수 있는 믿을 만한 전송 능력을 제공합니다. 클라이언트/서버 컴퓨팅이 그렇게 강력한 것은 이러한 이유 때문입니다.

Informix에서는 두 가지 연결 유형을 제공하고 있음을 이해하는 것이 중요합니다.

- **네트워크 연결** - 서로 다른 컴퓨터에 상주하는 데이터베이스에 대해서는 반드시 이 연결 유형을 사용해야 합니다. 같은 컴퓨터에 있는 데이터베이스에 연결할 때에도 이것을 사용할 수 있지만 공유 메모리 연결 방식보다는 속도가 더 느립니다. 같은 컴퓨터의 데이터베이스에 대한 네트워크 연결은 "로컬 루프백" 연결이라고 합니다.
- **공유 메모리** - 데이터베이스가 응용 프로그램과 같은 컴퓨터에 있을 때에만 이 연결 유형이 사용됩니다. 이것은 공유 메모리에서 실행되므로 네트워크 연결보다 훨씬 빠릅니다. 그러나 네트워크 연결보다 오류 발생 가능성이 더 많습니다. 오류가 너무 많이 발생할 경우에는 로컬 루프백의 사용을 고려해 보는 것이 좋습니다.

nettype 매개변수 분석

Informix nettype 매개변수는 Informix 데이터베이스들 사이의 통신 방법을 정의하기 위해 사용합니다. 이 매개변수는 sqlhosts와 onconfig 파일에서 찾아볼 수 있는데, 이 파일들에 대해서는 "Informix 네트워킹 파일 설정하기" 단원에서 소개합니다. nettype의 형식은 다음과 같습니다.

dd는 데이터베이스 서버 제품으로 다음 중 하나가 될 수 있습니다.

- on or ol - OnLine 서버(on을 더 많이 사용함)
- se - Informix SE 서버
- dr - Informix Gateway with DRDA(TM)

iii는 네트워크 인트페이스로 다음 중 하나가 될 수 있습니다.

- ipc - 프로세스간 통신 (Interprocess communication, 공유 메모리 연결 시에만 사용)
- soc - 소켓(sockets)
- tli - 전송 단계 인터페이스(transport level interface)

ppp는 네트워크 프로토콜 또는 ipc 유형으로 다음을 포함할 수 있습니다.

- tcp - 네트워크 연결에서의 TCP/IP 프로토콜
- spx - 네트워크 연결에서의 IPX/SPX 프로토콜
- shm - 공유 메모리 연결(shared memory connection, ipc 유형에만 사용)

nettype의 몇몇 예를 들면 다음과 같습니다.

- onsoctcp - 소켓과 TCP/IP를 사용하는 OnLine 엔진
- ontlispx - TLI와 IPX/SPX를 사용하는 OnLine 엔진
- ontlitcp - TLI와 TCP/IP를 사용하는 OnLine 엔진
- onipcshm - IPC와 공유 메모리를 사용하는 OnLine 엔진
- setlitcp - TLI와 TCP/IP를 사용하는 Informix SE

이제 네트워킹 필드에 대해서 설명합니다. 이해를 돕기 위해 단원의 순서를 일부 변경하여 프로토콜을 먼저 설명하고 다음에 인터페이스를 소개하도록 하겠습니다.

네트워크 프로토콜 사용하기: TCP/IP, IPX/SPX

네트워크 프로토콜은 네트워크를 통해 데이터가 전달되는 방법을 정의합니다. Informix와 사용할 수 있는 두 개의 네트워크 프로토콜은 TCP/IP와 IPX/SPX입니다. 물론 UNIX 사용자에게는 TCP/IP가 매우 친근하게

느껴질 것입니다. IPX/SPX는 Novell 네트워킹에 사용됩니다.

무엇보다도 클라이언트와 서버가 같은 네트워크 프로토콜을 사용하도록 연결을 구성해야 합니다. 예를 들어, Novell 네트워크를 사용하는 환경에서는 모든 PC가 IPX/SPX 연결을 지원할 것이므로, 연결의 양쪽에서 IPX/SPX를 사용할 수 있습니다. 반면, 대부분의 UNIX와 Windows NT 시스템에는 TCP/IP가 이미 포함되어 있으므로, Novell 네트워크를 사용하지 않는다면 IPX/SPX를 사용할 이유가 없을 것입니다.

어느 경우이건 서버에 대해 하나 이상의 연결 유형이 사용될 수 있는데, 각 연결은 클라이언트와 일치해야 합니다. 모든 컴퓨터가 두 프로토콜을 다 지원하지는 않는다는 점에 유의해야 합니다. Informix와 네트워크 구성을 확인하여 어떤 것이 적합한지 결정해야 합니다.

(주의)

하나 이상의 프로토콜을 사용하여 Informix 서버를 구성할 수 있습니다. Informix의 onconfig 파일을 사용하면 가능합니다. 실제로 네트워크에서 TCP/IP와 IPX/SPX를 모두 지원한다면, 둘을 모두 사용하는 것이 유리한 경우가 있을 것입니다.

공유 메모리 연결 사용하기

클라이언트 응용 프로그램과 데이터베이스 서버 사이의 공유 메모리 연결은 클라이언트와 데이터베이스 서버가 모두 같은 컴퓨터에 있을 경우에만 사용 가능합니다. 공유 메모리 연결은 TCP/IP 연결보다 훨씬 빠르며 대부분의 경우 같은 컴퓨터의 데이터베이스를 사용하는 응용 프로그램에 대해 사용합니다. 공유 메모리 연결의 유일한 문제점은 컴퓨터의 메모리를 사용하므로 메모리에 관련한 오류가 있을 수 있다는 것입니다. 공유 메모리 응용 프로그램의 예로는 dbaccess와 Informix 4GL 프로그램에서 수행하는 질의들을 들 수 있습니다.

데이터베이스 서버는 공유 메모리와 네트워크 연결의 조합을 사용하여 구성할 수 있습니다. 두 유형의 연결이 공존할 경우에는 이렇게 할 필요가 있습니다. 데이터베이스에 액세스하는 질의가 로컬과 네트워크 양쪽에서 모두 수행된다고 가정해봅시다. 이럴 경우에는 TCP/IP와 공유 메모리 연결 구성을 모두 필요로 할 것입니다. 구성은 onconfig와 sqlhosts에서 할 수 있는데, 여기에 대해서는 뒤에 나오는 단원에서 설명합니다.

네트워크 프로토콜 사용하기: soc, tli, ipc

네트워크 인터페이스는 네트워크 프로토콜을 사용하여 컴퓨터들 사이에서 통신이 이루어지는 서로 다른 방법을 정의합니다. 이는 통신에 사용되는 실제 API(Application Programming Interface)를 정의하는 것으로, TCP/IP와 IPX/SPX는 약간 다릅니다.

네트워킹 연결에는 soc나 tli 중 하나를 사용할 수 있습니다. 소켓(soc)은 오래 전부터 사용되어 왔습니다. TLI(tli)는 소켓보다는 이후에 발표되었으나 신뢰할 수 있는 인터페이스입니다. 대부분의 시스템이 둘 중 하나를 또는 두 인터페이스 모두를 제공합니다. 어떤 것이 현재의 Informix 버전을 지원하는지 확인하려면 Informix 제품에 대한 릴리즈 노트(release notes) 파일을 참고하십시오. $InformixDIR/release 디렉토리의 ONLINE_7.x 또는 IDS_7.x(x는 현재 사용하는 Informix 버전을 말함) 파일에서 이러한 내용을 확인할 수 있습니다. 소켓과 TLI가 모두 지원되면 네트워크 또는 시스템 관리자와 상의하여 현재의 운영 체제에 가장 적합한 것을 결정할 수 있습니다. IPX/SPX에서 지원되는 인터페이스 유형은 TLI 뿐입니다.

공유 메모리 연결 시에는 ipc(Interprocess communication, 프로세스간 통신)만을 사용할 수 있습니다.

Informix 네트워킹 파일 설정하기

Informix 구성 파일들은 네트워크에서 어떤 데이터베이스와 서버를 이용할 수 있는지를 Informix에 알려 줍니다. 앞으로 보게 되겠지만, 이 파일들은 네트워크 관리 파일들과 밀접하게 관련이 있습니다.

onconfig 파일

각각의 Informix 인스턴스는 그 인스턴스의 특성을 정의하는 파일을 가지고 있습니다. 바로 onconfig 파일입니다. 이 파일은 $ONCONFIG 환경 변수로 지정하며 $InformixDIR/etc 디렉토리에서 찾을 수 있습니다. 그림 1의 각 데이터베이스 서버는(예를 들면 stores와 cust_data) 각각의 onconfig 파일을 가집니다. 다음의 onconfig 매개변수들은 네트워크 데이터베이스와 함께 작동합니다.

- DBSERVERNAME

sqlhosts 파일의 dbservername 매개변수와 같습니다. 이것으로 각각의 고유한 데이터베이스 서버를 구별합니다.

- DBSERVERALIASES

이것은 DBSERVERNAME 대신 사용할 수 있는 별칭을 제공합니다. 일반적으로 이 별칭들은 DBSERVERNAME과 다른 네트워크 프로토콜을 사용하고자 할 때 해당 프로토콜을 지정하기 위해서 사용됩니다. 예를 들어, DBSERVERNAME이 공유 메모리를 통한 연결을 사용한다면 DBSERVERALIASES는 TCP/IP 연결을 정의하여 사용하도록 할 수 있을 것입니다.

- NETTYPE

이 변수는 선택적이며, 이 매개변수로써 데이터베이스 엔진, 인터페이스 유형 및 네트워크 프로토콜과 관련하여 연결의 유형을 정의합니다.

다음은 basenji에 대한 onconfig 항목들의 보기입니다.

```
DBSERVERNAME basenji_shm # 공유 메모리 연결 - 기본
DBSERVERALIASES basenji_tcp # tcp/ip 연결 - 다른 서버에서
                            # 사용함
```

그림 2는 sqlhosts와 onconfig 사이의 관계를 보여줍니다.

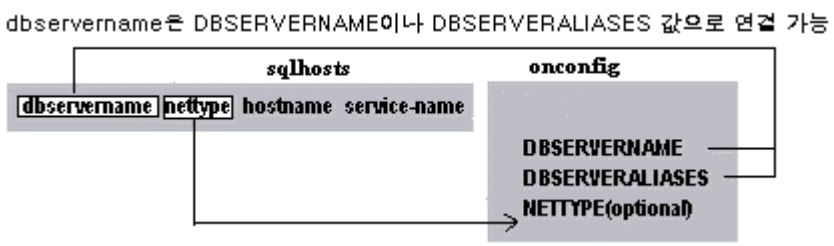

그림 2: sqlhosts와 onconfig 사이의 관계

운영체제 파일 구성하기

Informix 네트워크 연결을 구성하려면 관련된 운영 체제 파일을 적절히 설정해야 합니다. Informix 관리자는

서 services 파일에는 다음의 내용이 반드시 있어야 합니다.

```
ftp 21/tcp

telnet 23/tcp

smtp 25/tcp mail

time 37/tcp timserver

time 37/udp timserver

name 42/udp nameserver

whois 43/tcp nicname

domain 53/udp

... ...

prod_tcp

2001/tcp
```

처음의 몇 항목은 services 파일의 일부 공통 항목입니다. 눈에 익은 TCP/IP 서비스와 포트 번호(예를 들어 ftp는 포트 21)를 볼 수 있을 것입니다.

Informix에 대한 항목들은 관리자가 추가합니다. 이것들의 포트 번호/프로토콜은 고유한 것이어야 하며 알아보기 쉬워야 합니다. 이 Informix 인스턴스에 대한 서비스 이름은 prod_tcp이며 이것의 고유한 포트 번호는 2001입니다. 이 서비스 이름은 앞서 설명한 바 있는 sqlhosts 파일에서도 사용됩니다. 공유 메모리 연결 시에는 services에 항목이 필요하지 않습니다. 그림 3은 Informix와 운영 체제 네트워킹 파일 사이의 관계를 보여줍니다.

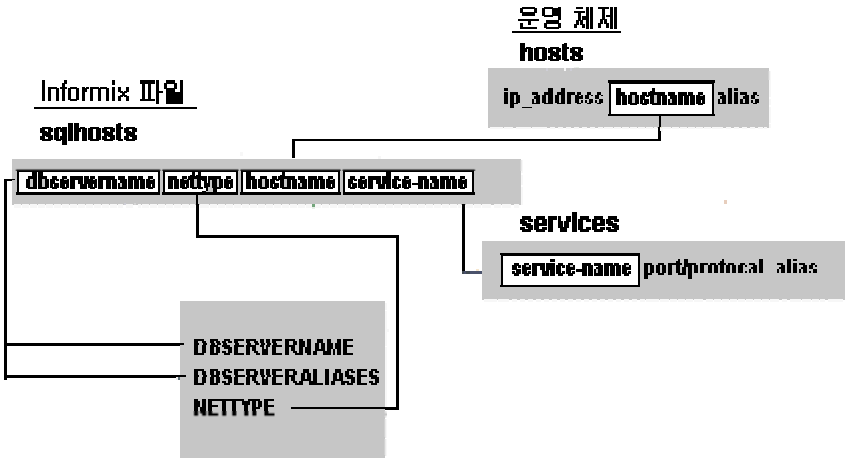

그림 3: Informix와 운영 체제 파일 사이의 관계

보안 파일 사용하기

특정 서버의 데이터베이스를 사용하려면 그 컴퓨터에 대한 로그인 ID가 있어야 합니다. 사용자가 로그인하여 다른 컴퓨터에 대한 셸 프롬프트를 얻을 수 있도록 해야 하는 것은 아니지만 유효한 로그인 ID는 반드시 있어야 합니다(다음의 "주의"를 참고하십시오). 요점을 말하자면 로그인 했을 때 어떤 일이 발생할 것인가 하고는 상관 없이 사용자에게 반드시 유효한 ID가 있어야 한다는 것입니다.

(주의)

일반적으로 데이터베이스 서버로는 사용자들이 로그인하지 못하게 하는 것이 바람직합니다. 이러한 서버들은 보통 데이터베이스 용도 전용이므로 관리자만 사용할 수 있도록 해야 합니다.

셸 프롬프트에는 실제로 액세스하지 못하도록 사용자 로그인 ID를 만들려면 즉시 종료되는 셸 스크립트를 만들고 그것을 각 사용자의 초기 셸로 사용합니다. UNIX에서는 한 줄에 "exit"라는 단어가 있는 스크립트를 만들 수 있습니다. 일부 UNIX 시스템은 이러한 용도로 사용할 수 있는 /dev/false 장치를 제공합니다.

로그인 ID 이외에도 사용자가 다른 컴퓨터로 Informix 연결을 할 수 있도록 해야 합니다. 다음과 같은 방법이 있습니다.

1. Informix SQL CONNECT 명령을 사용합니다. CONNECT 명령은 사용자 ID와 암호(선택적으로 CONNECT 명령과 함께 제공됨)를 가지고 연결을 할 수 있도록 해 줍니다. CONNECT를 사용하는 것은 더 안전하기는 하지만 덜 실용적인데, 그 이유는 매번 실행해야 하기 때문입니다. CONNECT의 구문은 CONNECT @dbservername입니다. 다른 항목들에 대해서는 앞에 나온 "분산 데이터베이스에서 Informix SQL 사용하기" 단원에서 설명하였습니다.

2. 연결하고자 하는 원격 컴퓨터에서, hosts.equiv 파일(UNIX에서는 /etc/hosts.equiv)에 연결이 필요한 모든 파일들을 포함시킵니다. hosts.equiv에 포함된 컴퓨터의 사용자들은 암호 없이 로그인 할 수 있으므로 주의해야 합니다(앞의 주의에서 설명한대로 하여 로그인 하려고 하는 사용자들이 할 수 있는 작업 내용을 제어할 수 있습니다). hosts.equiv에 있는 컴퓨터들은 믿을 만한 호스트로 인식됩니다. 예를 들어 corgi의 사용자가 pug의 데이터에 액세스하기 위해서는 pug의 hosts.equiv 파일에 단지 "corgi"라는 단어가 포함된 항목만 있으면 됩니다.

3. 연결하고자 하는 원격 컴퓨터에서, 액세스가 필요한 각 사용자의 홈 디렉토리에 .rhosts 파일을 만듭니다.

hosts.equiv와 마찬가지로 .rhosts 파일도 믿을 만한 호스트를 만들지만, 이 경우에는 지정한 사용자에 대해서만 그러합니다. 따라서 만약 Dan이라는 사용자가 beagle에 액세스하려면, 관리자는 원격 컴퓨터에서 Dan의 홈 디렉토리에 .rhosts를 만들고 이 파일의 한 줄에 "corgi"라는 단어를 포함시키면 됩니다.

4. .netrc 파일을 사용합니다. 이 파일은 연결을 초기화하는 컴퓨터(여기에서는 corgi)의 사용자들에 대해 만들 수 있습니다. 이 파일은 특정한 연결을 위해 사용자들이 로그인 ID와 암호를 지정할 수 있도록 합니다. 이 파일 역시 부적절하게 사용하면 보안상의 문제를 일으킬 수 있습니다.

5. 클라이언트 응용 프로그램(예를 들면 PC 응용 프로그램)에서 적절한 Informix 네트워킹 유틸리티를 사용합니다. 예를 들어 Informix setnet 명령을 사용하면 사용자들은 각자의 PC에서 로그인 ID와 암호를 구성할 수 있습니다.

일단 사용자가 성공적으로 원격 컴퓨터에 연결하면, 그 데이터베이스 세션 동안에는 다시 연결할 필요가 없습니다.

(주의)

대부분의 사용자 데이터베이스 응용 프로그램에서는 사용자가 처음 로그인할 때 암호를 한 번만 입력하도록 요구합니다. 복수 사용자의 응용 프로그램을 사용할 것이라면 이 방법이 분명 최상의 해결책입니다. "믿을 만한 호스트"를 만들 경우에는, 비인증 사용자가 시스템에 쉽게 액세스할 수 있는 방법이 없다는 점을 기억해야 합니다.

파일들을 모으기

지금까지 연결을 구성하는 운영 체제 파일과 Informix 파일에 대해 설명했으므로, 이제 몇몇 예제를 가지고 작업할 수 있을 것입니다. 우선 corgi 서버에 있는 파일로 시작할 것입니다. 목록 1은 hosts 파일을 보여줍니다.

목록 1. corgi의 hosts 파일

```
127.0.0.1 localhost # "로컬 루프백"에 사용됨 - 데이터는
                    # 이 서버로 다시 보내짐
```

```
192.1.1.4 basenji

192.1.1.5 corgi
```

네트워크의 모든 서버는 hosts에 항목으로 포함되어 있습니다. 물론 각각의 서버에서 IP 주소가 같은 방식으로 구성되어 있어야 합니다.

목록 2는 services 파일에서 우리의 데이터베이스 연결과 관련된 부분을 보여줍니다.

목록 2. services 파일의 일부

```
prod_tcp 2001/tcp
```

다른 서버들이 corgi에 액세스해야 하는 경우를 제외하면, 보안 파일(.rhosts와 hosts.equiv)에 항목을 만들 필요가 없습니다. 실제로 보안 파일에 호스트를 덜 만들수록 더 좋습니다.

목록 3에서와 같이, corgi의 sqlhosts 파일에는 네트워크의 각 호스트에 대한 항목이 포함되어 있어야 합니다.

목록 3. corgi의 sqlhosts 항목들

```
# 공유 메모리를 통한 로컬 연결

corgi_shm onipcshm corgi corgi_shm

# 원격 연결 ; 로컬 루프백 연결에 대해

corgi_tcp onsoctcp corgi prod_tcp

pug_tcp onsoctcp pug prod_tcp

beagle_tcp onsoctcp beagle prod_tcp

basenji_tcp ontlitcp basenji prod_tcp
```

corgi의 데이터베이스 인스턴스에 대한 onconfig 파일에는 목록 4에 있는 항목들이 포함되어 있어야 합니다.

목록 4. corgi의 onconfig 파일 항목들에 대한 보기

```
DBSERVERNAME corgi_shm  # 공유 메모리 연결 - 기본

DBSERVERALIASES corgi_tcp # tcp/ip연결- 다른 서버에서 사용함
```

이제 다른 서버의 항목들을 살펴볼 것입니다. 여기에서는 beagle을 보기로 사용합니다. hosts와 services 파일 항목들은 목록 1과 목록 2에서 보듯이 corgi의 항목들과 똑같습니다. 그러나 네트워크 보안 파일들은 다릅니다. 관리자가 corgi의 모든 사용자들에게 액세스를 허용하기로 했다고 가정해 봅시다(다시 말해서 corgi가 믿을 만한 호스트가 된 것입니다). beagle의 /etc/hosts.equiv 파일에 다음 항목이 있을 것입니다.

```
corgi
```

관리자가 사용자 별로 액세스를 허용하기로 방침을 정했다면, beagle에 있는 각 사용자의 .rhosts 파일에 똑같은 항목을 대신 만들었을 것입니다.

목록 5에서 볼 수 있듯이 beagle에 대한 sqlhosts 파일은 로컬 공유 메모리 연결이 다르다는 것을 제외하고는 corgi의 파일과 거의 똑같습니다. 공유 메모리 연결은 같은 컴퓨터에서 실행되는 클라이언트에 대해서만 이용이 가능하다는 점을 기억하십시오.

목록 5. beagle의 sqlhosts 항목들

```
# 공유 메모리를 통한 로컬 연결
beagle_shm onipcshm beagle beagle_shm
corgi_tcp onsoctcp corgi prod_tcp
pug_tcp onsoctcp pug prod_tcp
# 원격 연결 ; 로컬 루프백 연결에 대해
beagle_tcp onsoctcp beagle prod_tcp
basenji_tcp ontlitcp basenji prod_tcp
```

끝으로 beagle의 onconfig 파일에 있는 항목들은 corgi의 항목들과 유사하지만, beagle 서버 이름을 반영하여 항목 이름이 변경됩니다.

목록 6. beagle의 onconfig 항목 예

```
DBSERVERNAME beagle_shm  # 공유 메모리 연결 - 기본
DBSERVERALIASES beagle_tcp # tcp/ip 연결 - 다른 서버에서 사용함
```

이제 앞의 "분산 데이터베이스에서 Informix SQL 사용하기" 단원에서 설명한 방법 중에서 하나를 사용하여

연결을 할 수 있습니다. 사용자는 corgi에서 다음의 SQL 문 중에서 하나를 실행할 수 있습니다.

```
select * from cust_data@beagle_tcp:customers;

create synonym customers for

cust_data@beagle_tcp:customers;

connect @beagle_tcp;
```

그림 4는 이번 예제의 연결과 관련된 모든 파일들을 보여 줍니다.

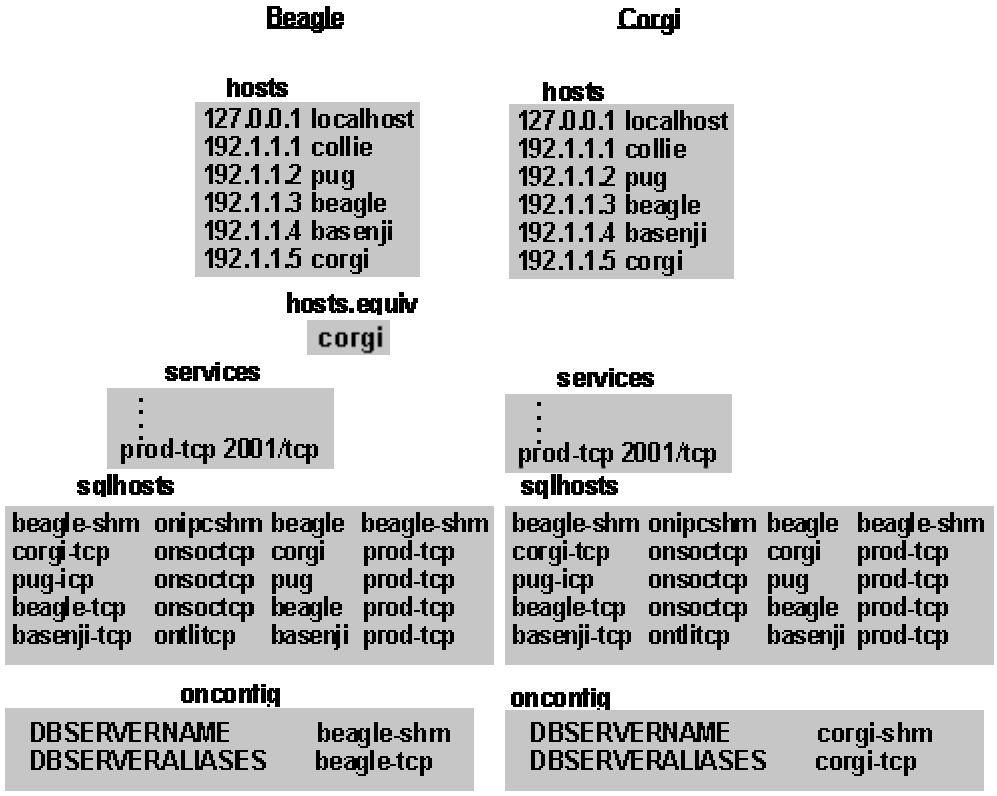

그림 4: 두 서버의 연결을 위해 필요한 모든 파일들의 예제

collie, pug 및 basenji 서버는 유사한 방식으로 구성할 수 있습니다.

Informix 구버전과의 네트워킹

Informix 구버전에 연결하는 데에는 비슷하기는 하지만약간 다른 점이 있습니다. 운영 체제 및 기타 파일들을 설정하는 방법은 이전 단원들에서 설명한 것과 근본적으로 똑같습니다. 그림 1의 beagle과 basenji에서 실행되는 Informix Online 5.x와 6.0 이전의 Informix SE 같은 제품에는 네트워킹이 내장되어 있지 않습니다.

Informix OnLine 5.0에서는 네트워킹을 위해 Informix STAR라는 제품을 사용합니다. 이 제품은 6.0 버전과

OnLine 이후 버전에 내장된 네트워킹과 유사하여, 다른 컴퓨터로부터 Informix 네트워크 연결이 가능하게 해 줍니다. 5.x 서버로 연결이 되도록 하려면 Informix STAR 제품이 실행되고 있어야 합니다. NFORMIX-STAR를 시작하는 방법에 관한 자세한 내용은 Informix STAR 설치 안내서를 참조하십시오. 그림 5는 beagle의 5.x 데이터베이스에 연결하기 위해 필요한 과정을 보여 줍니다.

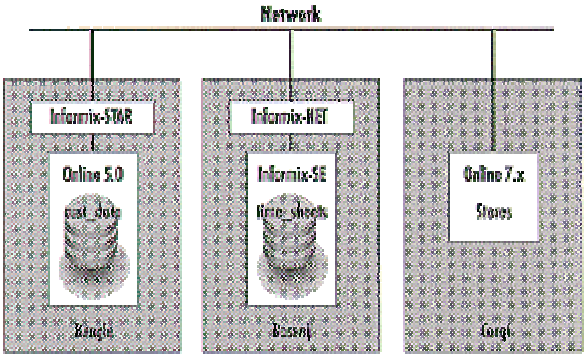

그림 5: Online 5.x 데이터베이스에 대한 Informix STAR 사용법

마찬가지로 Informix SE(Standard Engine)의 6.0 이전 버전에서도 Informix NET라는 네트워킹 제품을 사용합니다. Informix STAR와 마찬가지로 이 제품도 Informix SE 데이터베이스 서버가 있는 컴퓨터에서 실행됩니다. 역시 자세한 내용은 관련 제품의 안내서를 참조하십시오.

Informix는 기술력과 함께 변화하였으며 이제는 네트워킹 제품들과 강력하게 통합되었습니다. 6.0 버전 이후부터는 데이터베이스 엔진에 필요한 네트워킹 제품이 포함되어서 Informix STAR나 Informix NET 같은 별도의 제품이 필요없게 되었습니다.

클라이언트 파일 구성하기

지금까지는 서버들 사이의 연결을 위해 Informix 환경을 준비하는 과정을 소개하였습니다. 이러한 과정을 따르면 그림 1의 데이터베이스 서버들에 서로 연결할 수 있습니다. 그러나 PC나 Macintosh 클라이언트는 어떻게 사용할 수 있을까요?

클라이언트 컴퓨터에서는 Informix Dynamic 4GL 또는 ERWin이나 Powerbuilder 같은 다른 회사 제품을 사용하여 응용 프로그램을 실행할 수 있습니다. Java와 Visual Basic 같은 새로운 제품들도 여기에 포함됩니

클라이언트용 컨넥트 모듈이 필요합니다. Informix 7.2 버전 이전에는 모든 서버에 대해 PC에서 Informix NET 유틸리티를 사용했습니다. 이 유틸리티는 별다른 설명이 필요없는 제품입니다. 7.2와 그 이후의 버전에서는 Informix CONNECT와 Informix CLI, Informix Client-SDK 같은 제품이 개발되었습니다.

이러한 제품을 PC에 설치한 후 DB Server에 연결할 수 있도록 setnet32와 같은 모듈을 설정해야 합니다. setnet32의 설정 방법은 Informix TechNotes Vol2를 참고하십시오. URL은 http://www.Informix.com/kr/service/cs/tnotes/vol2입니다.

맺음말

이 글에서는 클라이언트/서버 통신의 이점에 대해 설명하였습니다. 그리고 Informix와 운영 체제에 필요한 다양한 구성에 대한 설명과 함께 예를 들어 네트워크 데이터베이스를 설정 방법을 살펴보았습니다.

오늘날의 컴퓨터 사용 환경에서 네트워크가 많은 발전을 불러온 것처럼 데이터베이스 환경에서의 네트워크 연결이 데이터베이스 사용에 있어서 보다 넓은 영역으로의 확대를 가져올 것입니다.

전자 상거래를 촉진시키기 위한 Informix Datablades 사용

소개

업계의 비전을 제시하는 사람들은 지난 수 년동안 전자 상거래의 도래를 예고해 왔지만 최근까지도 정평있는 기업 가운데에 파일럿 프로젝트 이상으로 전자 상거래 구현을 위해 자원을 투자한 곳은 거의 없었습니다. 이제 가시적으로 놀라운 성공을 거둔 몇몇 웹 사이트, 전반적인 기술 발전 및 지속적으로 성장하는 웹 공동체로 인해 전자 상거래는 성숙기로 접어들고 있습니다. 이제 인터넷은 오늘날의 비즈니스 성공을 위한 가장 중요한 요소가 됐습니다. 올바르게 구축될 경우 기업은 막강한 경쟁 우위를 확보함으로써 시장 확대, 운영 효율성 향상, 우량 고객 확보, 매출 증대 등을 실현할 수 있습니다.

전자 상거래에 대한 엄청난 관심에도 불구하고 모든 상황에 적합한 간편한 전자 상거래 솔루션은 없습니다. 그러나 전자 상거래 문제의 특정 분야들에 대해서는 훌륭한 솔루션들이 많이 있습니다. 문제는 이러한 최상의 솔루션들을 빈틈없이 융통성 있게 그리고 확장 가능하도록 한 곳에 결합하는 것입니다. 한 가지 방법은 Informix Internet Foundation 을 사용하는 것입니다.

완벽한 전자 상거래 솔루션의 특성

완벽한 전자 상거래 솔루션은 전통적인 구매 방식의 모든 면을 포괄합니다.

완벽한 전자 상거래 솔루션은 전통적인 구매 방식의 모든 면을 포괄합니다.

광고 및 마케팅

광고 및 마케팅에는 도처에 널려 있는 배너 광고와 브로셔로부터 전통적 마케팅을 겸비한 상호 판촉 활동, 공동체 생성, 부상하고 있는 1 대 1 마케팅 패러다임에 이르기까지 모든 수요 창출 활동 및 계획들이 포함됩니다.

매매

매매는 물품을 출시하고, 한데 모으고, 가격을 책정하고 판매하는 방법에 대한 것입니다. 웹 사이트가 중요한 판매 수단으로 간주되려면 쿠폰, 환불, 회원 할인, 친화 프로그램, 경품, 번들 제공, 대량 구입 시 할인, 자주 구입할 경우 할인, 추천 상품, 시험 사용 기간 및 데모 제품 등과 같은 다양한 매매 방식을 지원할 수 있어야 합니다. 더 중요한 것은, 성숙한 매매 모델을 가진 웹 사이트라면 새로운 거래 방법이 생길 경우 이를 쉽게 통합할 수 있도록 융통성이 있어야 한다는 것입니다.

대금 지불 절차

신용 카드 사용이 안전하다는 것을 대중에게 납득시키는 문제가 아직 남아 있기는 하지만, 판매업체들이 인터넷에서 신용 카드 주문 방식을 채택하기 위한 몇 가지 좋은 솔루션이 있습니다. 하지만 이 분야도 아직 시작 단계입니다. 웹 사이트 소유주들은 구매 주문, 전자 지갑, 자동 및 반자동 갱신 및 보다 비밀스런 기타 지급 프로토콜로 대금 지불 절차를 확대하기 위해 여러가지 방법을 실험하고 있습니다.

가격 계산

구매시 사용자에 대한 적정 세금과 운송료를 결정하는 것은 사소한 문제가 아닙니다. Taxware, UPS, Federal Express 와 같은 몇몇 회사들은 이 문제를 해결하기 위해 많은 자원을 할애하고 있습니다. 각자의 것을 직접 만들기 보다는 사이트 설계자들이 이러한 솔루션들을 통합할 수 있도록 하는 것이 바람직할 것입니다.

주문 이행

고객은 제품의 재고가 있는지, 언제 도착할지 등 자신의 주문에 대한 정보를 기대합니다. 간단히 말해서, 앞서가는 웹 사이트에서는 머지않아 판매 회사의 전통적 주문 이행 시스템과 수준 높은 통합이 이루어져야

할 것입니다.

추적

사용자 활동에 대한 정보를 확보하는 것은 매우 중요합니다. 현재 확인 가능한 일부 정보로는 클릭 경로, 조회 및 하루의 시간별 활동 등이 있습니다.

프로파일링

대부분의 추적 솔루션들은 제품에 대해서는 많은 보고를 제공하지만 마케팅 전문가들이 요구하는 시장 단위 분석은 제공하지 않습니다. 훌륭한 프로파일링은 1 대 1 마케팅 약속을 실현하기 위한 핵심적인 기술입니다.

고객 서비스/재판매/판매 증가

일 회 구매자를 지속적인 구매자로 만들려면 회사가 자신의 요구에 응답을 잘 한다는 것을 사이트 사용자가 느끼도록 해야 합니다. 아직 자동으로 되지는 않지만 웹은 이러한 인식을 심어 주기 위해 여러 가지 놀라운 방법을 제공합니다. FAQ, 포럼, 뉴스 그룹 및 전자 우편이 모두 이러한 인식을 심어 주는 방법입니다.

Informix Internet Foundation; 전자상거래 솔루션

전자 상거래를 지원하는 플랫폼은 확장 가능하며 편리한 관리 방식으로 여러 시스템들을 한데 모을 수 있어야 합니다. 이를 위해 플랫폼에 요구되는 것은 많지만 쉽게 구현할 수 있는 것은 아닙니다. Informix Internet Foundation 은 세계적인 수준의 전자 상거래 응용 프로그램에 요구되는 통합성, 확장성, 융통성 및 관리 편이성을 제공합니다.

통합성

Informix Internet Foundation 은 DataBlade 기술을 통해 상상할 수 있는 어떤 기능과 데이터 유형도 매끄럽게 결합할 수 있습니다. 이 기술은 다른 회사들의 데이터와 기능을 한데 묶어서 마치 그것들이 내장되어 있는 것처럼 매끄럽게 작동하도록 하는 데에 적합합니다. 다른 데이터베이스로의 게이트웨이는 이러한 매끄러운 결합의 좋은 예입니다. 또한 고급 그래픽 알고리즘을 사용자에게 친숙한 인터페이스로 포장하는 다수의 고급 이미지 및 비디오 프로세싱 블레이드(blade)도 그러합니다. 전자 상거래에서의 대금 지불 처리와 복잡한 실체들(고객, 제품, 회사와 거래 유형)은 DataBlade 의 훌륭한 대상이 됩니다. 그래픽, 비디오 프로세싱 및 다른 풍부한 데이터 조작 블레이드는 풍부한 내용으로 부가 가치를 창출하려는 많은

사이트들에 특히 유용합니다.

확장성

외부 시스템을 데이터베이스의 일부로 만듦으로써 시스템 설계자는 풍부한 기능으로 업계를 선도하는 엔터프라이즈 RDBMS 에서 제공하는 모든 확장 기능(복제 및 분산 서버 환경과 같은 기능)을 사용할 수 있습니다.

성능

Informix Internet Foundation 에 내장된 전자 상거래 솔루션들은 확장성이 있으므로 훌륭한 성능을 발휘합니다(이 솔루션들은 엔터프라이즈 RDBMS 의 풍부한 기능들을 활용합니다). 또한 Informix Internet Foundation 은 대역폭을 더 잘 보존하기 때문에 보다 특별한 솔루션을 제공하며 더 나은 질의 최적화를 제공합니다.

대역폭 보존

방대한 양의 데이터에 대해 복잡한 통계 분석을 수행해야 하는 회계 응용 프로그램을 상상해 보십시오. 많은 경우 데이터는 계산에 대한 로직을 포함하는 서버에 저장된 데이터베이스로부터 파이프 아웃되어야 할 것입니다. 로직을 데이터와 함께 둠으로써 데이터 파이핑은 필요하지 않으며 결과만 네트워크를 통해 보내면 됩니다.

질의 최적화

여러 데이터 유형에 걸쳐 질의가 일어날 때 Informix 의 통합 접근법은 진가를 발휘합니다. 미식가에게 집 근처에서 좋아하는 음식을 찾을 수 있는 서비스를 제공하는 웹 사이트가 있다고 가정해 봅시다. 미식가는 서울의 강남구와 서초구에 위치한 이태리 식당에서 Penne ala Vodka 를 검색하려 합니다. 질의에 대한 결과를 돌려주기 위해 웹 사이트의 백엔드에서는 'Italian' 열거값에 대한 전통적인 RDBMS 검색 결과를, 레스토랑 기록의 메뉴 필드에서 'Penne'과 'Vodka'에 대한 문자열 검색 결과 및 강남구와 서초구 내에 있는 레스토랑에 대한 공간 검색 결과와 결합해야 합니다. 다른 시스템에서는 비표준 검색이 데이터베이스 외부에서 일어나므로 질의 최적화 범위에서 벗어나게 될 것입니다. 그러나 Informix 에서는 모든 검색이

동등하며 쉽게 최적화됩니다. 어떤 경우에는 공간 검색이 문자열 검색 이전에 발생하며 또 다른 경우에는 그 반대가 될 수도 있습니다. 질의에 따라 순서를 바꾸는 이러한 능력은 정해진 순서로 질의를 평가하는 시스템에 비해 엄청난 성능을 발휘할 수 있습니다.

융통성

Informix Internet Foundation 은 객체 관계형 데이터베이스이므로, 코드 재사용 및 용이한 코드 수정과 관련된 대부분의 이점을 취할 수 있습니다. 다형성(polymorphism)은 미래의 모듈을 위해 사용할 인터페이스를 정의하는 데 특히 유용합니다. 다른 회사들이 도입한 객체 관계형 접근법(IBM 의 DB2 는 예외)과는 달리 Informix Internet Foundation 은 다른 응용 프로그램에 대해 함수를 호출하는 플러그인 코드를 통해, 또는 별도의 API 와 서버 하위 시스템의 추가를 통해 객체 지향적 특성들을 상대적 토대 위에 붙잡아 두지 않습니다.

관리 편이성

Informix Internet Foundation 의 전자 상거래 구현에서는 웹 자산으로부터 다른 회사의 기능성에 이르기까지 모든 것이 데이터베이스에 상주합니다. 이 특성은 파일 시스템에 상주할 경우보다 훨씬 쉽게 내용에 액세스할 수 있음은 물론 내용에 대한 인터페이스가 사이트 전반에 걸쳐 일관성이 있음을 의미합니다. 결과적으로 Informix Internet Foundation 으로 운영되는 웹 사이트는 다른 웹 사이트에 비해 고객이 세부적인 내용에 익숙해지기까지의 학습 곡선이 더 짧습니다.

DataBlade 모델

Informix Internet Foundation 이 왜 전자 상거래에 적합한가를 알려면 DataBlade 모델을 이해하는 것이 중요합니다. 이 단원에서는 DataBlade 모델을 소개하고 간단한 전자 상거래 응용 프로그램을 통해 이러한 원리를 설명할 것입니다.

DataBlade 아키텍처는 객체 관계형 데이터베이스 시스템이나 범용 서버가 되기 위해 RDBMS 에 필요한 수정 사항을 통합합니다. 이 두 가지 용어는 근본적으로는 동의어지만 서로 다른 기능을 강조하고 있습니다. "객체 관계형"은 패러다임이 기본적인 관계형 개념은 물론 상속과 다형성을 지원한다는 사실을 강조합니다. "범용 서버"는 모든 유형의 데이터를 동일한 방법으로 지원한다는 사실을 강조합니다. Informix 의 기술 담당 이사이며 Illustra Technologies 의 설립자인 Michael Stonebraker 박사가 그의 백서에서 언급한 것처럼

액세스 방법, 실행자, 병렬 시스템, 옵티마이저 및 해석기 등 전통적 RDBMS 의 구성 요소들을 진정으로 객체 관계형이 되도록 하려면 반드시 수정해야 합니다.

새 유형의 비교 방법을 지원하기 위해서는 색인 액세스의 B 트리 방법이 일반화되어야 하는 것은 물론 스스로를 B 트리 검색에 내 주지 않는 데이터 유형을 지원하기 위해 완전히 새로운 액세스 방법이 허용되어야 합니다. B 트리는 노드 당 두 개의 child 를 가지는 트리인 바이너리 트리의 축약어입니다. 각각의 노드에는 데이터의 한 부분이 존재합니다. 트리 전반에 걸쳐 다음 기준이 유지됩니다. 즉, 노드 오른쪽 가지의 하위 노드(descendents)는 노드 자신의 값보다 크며 노드 왼쪽의 하위 노드는 노드의 값보다 작습니다. R 트리는 설명하기가 더 복잡합니다. 근본적으로는 비교 연산자가 트리의 짝수 레벨에서는 X 축 데이터를 사용하고 홀수 레벨에서는 Y 축 데이터를 사용한다는 점만 제외하면 B 트리입니다. 2 차원 데이터를 처리할 수 있는 바이너리 트리의 변형이라고 해도 될 것입니다. R 트리는 모든 크기의 데이터를 처리하도록 일반화할 수 있습니다.

마찬가지로 데이터의 함수에 대한 색인 또한 중요합니다. 그래픽 데이터의 필드를 생각해 보십시오. 개별 색인 자체는 직접적으로 거의 의미가 없습니다. 그러나 통계 함수에 따라 이미지의 순위를 정할 경우에는 큰 의미를 지닐 수 있습니다.

실행자는 사용자 정의 함수와 데이터 유형을 포함하는 질의를 해결할 수 있어야 합니다. 또한 새로운 데이터 유형과 함수가 추가될 때 마다 시스템을 멈추고 시작하는 것은 바람직하지 않으므로, 함수의 동적 연결과 연결 해제가 허용되어야 제대로 구현되었다고 할 수 있습니다.

병렬 시스템은 사용자 정의 집계를 지원하도록 확장돼야 합니다. 옵티마이저는 하드 코드된 것보다는 동적인 통계를 사용하여 경험적 축적 결과(heuristics)를 실행할 수 있어야 합니다. 또한 옵티마이저는 레코드 수에 대한 일반적인 고려 사항 이외에도 함수 색인에 대한 함수의 CPU 비용을 고려해야 합니다.

물론 해석기도 상속의 개념을 받아들이고, 생성된 시스템 카탈로그 내에 동적으로 읽어 들여진 메타데이터를 처리하기 위해 전반적으로 재작성해야 합니다.

Spacely Publishing 온라인 카탈로그

Informix Internet Foundation 의 데이터베이스 활성 기술이 전자 상거래 솔루션에서 어떻게 사용되는가를 설명하기 위해 이 단원에서는 출판 회사의 간단한 온라인 카탈로그의 구현에 대해 설명합니다.

응용 프로그램 설명

예제 응용 프로그램을 만든 Spacely Publishing, Inc.는 전문 회계 관계자들을 위해 무역 저널, 교본, 프리젠테이션 보조물, 회의록 및 기타 자료들을 출판합니다. Spacely 사는 전자 상거래를 도입하고자 하였고, 신속하지만 보수적인 방법으로 시작하기 위해 신용 카드로 지불할 수 있는 간단한 카탈로그 응용 프로그램을 선택했습니다. 궁극적으로 Spacely 사는 이 응용 프로그램이 보다 완벽한 전자 상거래 시설로서 발전하기를 희망합니다. 그림 1 에서 볼 수 있듯이, 응용 프로그램의 기본적인 페이지 흐름은 단순합니다. 사이트의 첫 페이지에서 사용자는 카탈로그의 범주를 선택하여 직접 탐색하거나 키워드 검색어를 입력합니다. 검색 결과는 글자와 함께 9 개의 작은 그림 그룹으로 나타납니다. 사용자는 제품의 세부 사항으로 들어갈 수도 있고, 다음 구매를 위해 항목을 장바구니에 추가할 할 수도 있습니다. 구매를 결정할 때 사용자는 다른 등록 정보와 함께 신용 카드 정보를 제출해야 합니다. 사용자가 구매한 내용을 다시 보여주는 구매 확인 화면이 나타납니다. 사용자가 승인하면 구매가 이루어집니다.

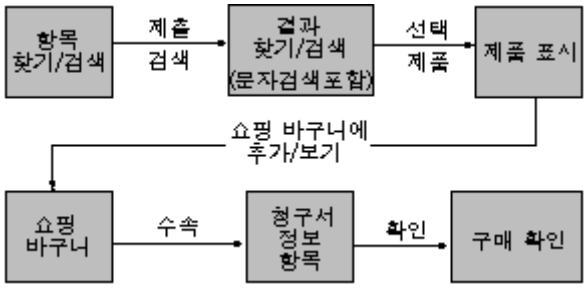

그림 1: 사용자는 관심있는 제품의 선택 목록을 보기 위해 몇 가지 범주를 선택할 수 있습니다. 이 목록에서 사용자는 특정 제품에 대해 보다 상세한 정보를 얻을 수 있으며 궁극적으로 하나 이상의 제품을 구입할 수 있습니다.

웹 연결

Web DataBlade 덕분에 HTML 코드, 이미지, 사운드 파일, 비디오 클립, 자바 애플릿 등 웹 사이트에서 요구하는 모든 웹 자산을 데이터베이스에 저장할 수 있습니다. Spacely 출판사 사이트에서는 모든 자산을 데이터베이스에 저장합니다. 사용자의 브라우저로 이러한 자산을 가져오는 방법은 다음과 같습니다. 사용자가 응용 프로그램의 Netscape Enterprise Server 로부터 페이지를 요청하면, 이 요청은 NSAPI(Web Driver 의 CGI 버전도 이용 가능함)를 통해 Web Driver 공유 객체에 대한 호출로 번역됩니다. Web Driver 는 입력 매개변수로부터 SQL 문을 만들어 내고 이를 데이터베이스로 불러옵니다. 그림 2 는 이

과정을 설명합니다. 데이터베이스는 웹 서버가 클라이언트로 보내는 HTML 페이지를 반환합니다.

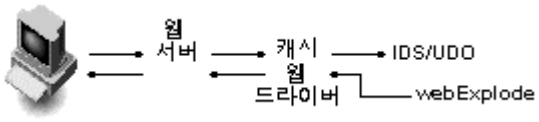

그림 2: Web Datablade 아키텍처-Web Driver 는 사용자의 입력 매개변수로부터 SQL 문을 만들어내고 데이터베이스에 질의를 보냅니다. 이에 대한 반응으로 데이터베이스는 웹 서버를 통해 클라이언트에게 보내지는 HTML 페이지를 생성합니다.

예를 들어, 사용자가 다음 요청을 하면,

Web Driver 는 이 요청을 분석하여 다음 질의를 생성합니다.

```
select webExplode(object,"prod_id=10023") from webPages
where ID='product_display';
```

WebExplode 는 C 로 만든 함수로서, 다른 표준 SQL 함수와 마찬가지로 질의 처리를 하는 동안 실행됩니다. "객체"는 내장된 WebBlade 태그와 함께 HTML 을 포함하는 문자열 필드입니다. WebBlade 태그에는 환경 변수정의와 모든 사용 가능한 SQL 명령을 포함할 수 있습니다. WebExplode 는 태그 내의 명령을 데이터베이스의 내용으로 대체하거나, 웹 서버 환경을 수정하거나, 데이터베이스의 데이터를 적절히 수정하기 위해 WebBlade 태그를 분석하고 실행합니다.

대금 지불 처리

사이버 시장에 처음 진입한 Spacely 사는 간단한 것을 원합니다. 첫 단계에서는 신용 카드 지불에 초점을 맞추려 합니다. 지불을 접수하기 위해 CyberCash 기능을 데이터베이스에 연결하는 Informix Internet Foundation 의 일부 DataBlade 생성 기능을 활용합니다

CyberCash 는 구매 시 실시간으로 권한을 주기 위해 웹 사이트를 신용 카드 처리기와 연결합니다. 판매 후 CyberCash 는 거래를 확정하고 완료하기 위해 완벽한 HTML 인터페이스를 제공합니다.

CyberCash 를 Informix Internet Foundation 과 통합하는 일은 CyberCash API 의 함수들을 데이터 타입

변환 코드(data conversion code)로 포장(wrapping)하고, 이렇게 포장된 함수를 공유 객체로 컴파일하고, 이를 데이터베이스에 등록하는 단순한 과정을 통해 이루어집니다.

다음은 데이터 타입 변환 코드를 사용한 예입니다.

```
#include


#include "credit.h"


int authorize (tsorder_id, tscurrency_amount,
            tscard_number, tscard_type,
            tscard_exp, tscard_fname,
            tscard_lname, tscard_address,
            tscard_city, tscard_zip,
            tscard_state, tscard_country)


    mi_text *tsorder_id;


    mi_text *tscurrency_amount;


...
```

```
   char *cscard_country
      = mi_lvarchar_to_string(tscard_country);

   char *transtype = "auth";

   credittransaction *ct;

ct=(credittransaction *)mi_alloc(sizeof(credittransaction));

ct = transaction(ct, transtype, csorder_id,
            cscurrency_amount, cscard_number,
            cscard_type, cscard_exp, cscard_fname,
            cscard_lname, cscard_address, cscard_city,
            cscard_zip, cscard_state, cscard_country);
return stcmp(ct->ravs_code;
}
```

코드는 공유 객체로 컴파일됩니다. 예를 들어 make 파일에는 다음과 같은 문자열이 포함될 수 있습니다.

```
create function function name(input para)
return data_type;
```

생성하고 컴파일한 함수는 반드시 등록을 해야 합니다.

```
create function authorize(transaction_t)
returns integer
as external name 'MI_HOME/functions/authorize.so';
```

문자열 검색

Spacely 사는 다수의 문서를 가지고 있는데, 특히 Spacely 사의 고객 입장에서 볼 때 알아보기 쉽게 분류되어 있지 않습니다. 이러한 이유로 Spacely 사는 계층적 검색 외에 1 단계에서 고급 문자열 검색을 제공하려 합니다. 이 검색을 구현하기 위해 Spacely 사는 그림 3 에서와 같이 Excalibur Text Search DataBlade Module 을 사용합니다.

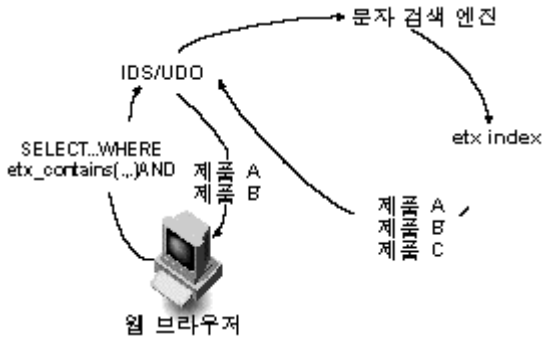

그림 3: 문자열 검색 블레이드. 문자열 검색 작업을 동반한 질의가 문자열 검색 블레이드 내에서 특수 문자열 검색 색인을 교대로 사용하는 함수에 의해 실행됩니다.

문자열 검색 엔진과 etx 색인은 모두 Informix Internet Foundation 의 일부이지만 여기서는 명확히 하기 위해 분리하여 표시했습니다.

```
...where etx_contains(description, row('multimedia'));
```

문자열 검색 블레이드는 데이터베이스내에서 새로운 유형, 연산자 및 액세스 방법을 정의합니다. 실제 응용에서 필자는 이것을 특정 유형으로 정의하여 제품의 설명 필드에 대한 검색을 할 수 있도록 했습니다.

```
CREATE TABLE products (
...
dscrpt IfxDescDoc,
...


);
```

다음으로 필드에 대한 색인 작업을 했습니다.

```
CREATE INDEX desc_idx ON products
(description etx_doc_ops)
USING etx in sbspl;
```

"Etx"는 문자열 검색 블레이드를 위해 만든 사용자 정의 액세스 방법입니다. 이것으로 색인을 만드는 과정에서, 빈도에 따른 해시테이블(hashtables)과 단어 순서를 비롯하여 몇몇 부수적인 데이터 구조가 형성됩니다. 색인이 사용될 때마다 DataBlade 에서 제공하는 다른 함수를 사용해 호출되므로, 세부적인 배열 상태는 Informix Internet Foundation 에서 알고 있지 않으며 알 필요도 없습니다.

```
...where etx_contains(description, row('multimedia'));
```

검색 설계자는 Etx_contains 및 기타 포함된 연산자를 사용하여 응용 프로그램에 부울(Boolean), 키워드(keyword), 구문(phrase), 근접성(proximity) 및 퍼지(fuzzy) 검색을 추가할 수 있으며 관련성에 따라 결과의 순위를 매길 수도 있습니다.

미래를 위한 계획

Spacely 사에서 응용 프로그램의 1 단계를 개발하는 동안 관리팀은 2 단계에서 추가하고자 하는 여러 향상된 기능들에 대한 제안을 수집합니다. 예를 들어 주문 처리 팀은 웹 사이트를 자신의 백엔드 시스템과 더 밀접하게 연결하고 싶어할 것이며, 마케팅 팀은 사이트에서 훨씬 더 다양한 거래를 할 수 있기를 바랄 것입니다. 설계자는 설계시 객체 지향적 프로그래밍 방식을 도입하여 장차 반드시 개선해야 할 부분에 대한 준비를 해야 합니다. 그림 4 에서처럼 설계자는 상속과 다형성을 통해 하위 형식(subtype)의 메서드에서 하위 형식 전용 코드를 격리시킴으로써 깔끔하고 수준 높고 안정적인 상태로 웹 페이지 코드를 유지할 수 있습니다.

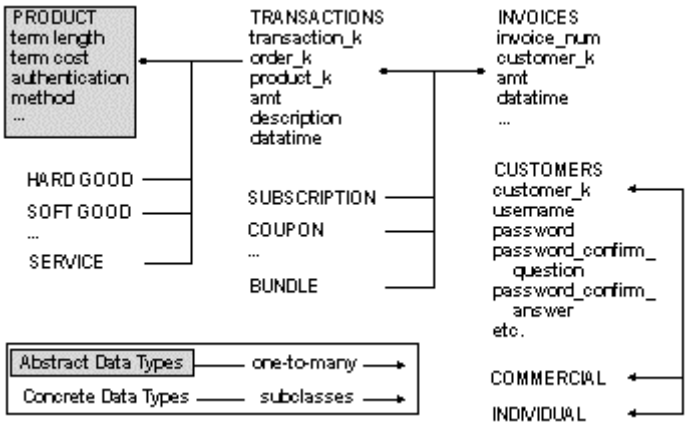

그림 4: Spacely 카탈로그를 위한 객체 관계형 스키마. 객체 지향적 프로그래밍을 채택하였으므로 미래의 기능 향상을 수용할 수 있도록 설계에 융통성이 있습니다. 상속과 다형성을 통해 설계자는 하위 형식의 메서드에서 하위 형식 전용 코드를 격리시킴으로써 깔끔하고 수준 높고 안정적인 상태로 웹 페이지 코드를 유지할 수 있습니다.

이것의 작동법에 대한 예로, 제품의 개념을 확장하기 위한 잠재적 코드를 고려해 보십시오.

```
create table products of new type product_t (
```

```
) under product_t;
```

다른 종류의 제품에 대해서는 운송료가 다르게 처리됩니다. 단순하게 하기 위해 여기에서는 운송료를 제품의 함수로만 간주합니다.

```
create function get_shipping(product_t)
returns numeric(6,2)
as return 0;


create function get_shipping(product_hard_t)
returns numeric(6,2)
as   return (((.2)*$1.size * $1.weight) /20+ 10) ;


create function get_shipping(product_soft_t)
returns numeric(6,2)
as return $1.size/1048576;
```

고객 장바구니 항목들에 대한 전체 운송료를 표시해야 할 때($cust_id 는 Web DataBlade 의 환경 변수라는 것을 염두에 두십시오)
호출은 다음과 같을 것입니다.

```
select sum(i.qty*get_shipping(p)) from products p, shopping_cart_items i
where p.id = i.prod_id and i.cust_id = $cust_id;
```

이 코드는 사용자 장바구니의 어떤 상품에 대해서도 작동하며, 쉽게 이해할 수 있고, 새로운 종류의 제품이 추가될 때에도 바뀌지 않을 것입니다.

맺음말

완벽한 전자 상거래를 한다는 것은 수많은 기능들을 밀접하게 하나로 결합하는 것을 의미하며, 이는 상당한 플랫폼 기반을 요구하는 대작업입니다. 플랫폼은 각각의 개별적인 부분들을 빈틈없이 통합해야

합니다. 마케팅과 거래, 대금 지불 처리와 주문 이행, 추적과 프로파일링, 고객 서비스 기능 등이 동일한 데이터에 대해 함께 작동해야 합니다. 인기 있는 전자 상거래 사이트에는 하루에 수백만에서 수천만의 고객이 방문할 수 있으므로 플랫폼의 규모 조절이 가능해야 합니다. 플랫폼은 반드시 융통성 있고 확장이 가능해야 합니다. 기업의 요구에 따라 성장해야 하며 급속히 부상하는 시장에서 경쟁력을 확보해야 합니다. 또한 속도가 빨라야 합니다. 성능이 떨어지면 인터넷 사용자들의 인내심 또한 저하됩니다. 무엇보다도 플랫폼은 관리가 편리해야 합니다. 제품을 갱신하고 판촉 내용을 변경하거나 배너를 편집하는 일에 프로그래머가 필요한 시스템은 실패할 수 밖에 없습니다.

Informix Internet Foundation 은 이러한 요구 사항을 충족시킬 수 있습니다. 이것이 강건한 기업 RDBMS 상에서 구축되었다는 사실은 확장 가능성과 뛰어난 성능을 의미합니다. 이것은 모든 데이터와 기능을 데이터베이스에 일급 데이터유형으로 가져올 수 있는 능력을 가지고 있으므로 통합을 잘 할 수 있는 능력 또한 갖게 됩니다. 모든 데이터를 동일한 데이터베이스에 넣고 일관된 방법으로 처리할 수 있으므로 Informix Internet Foundation 기반의 시스템은 관리가 편리합니다. 객체 관계형이므로 인터페이스와 구현을 분리할 수 있어 Informix Internet Foundation 은 융통성이 있고 확장이 가능합니다.

서버의 문제 해결 및 유지 보수

소개

이 글에서는 Informix 데이터베이스 서버를 조정하고 감시하는 데 도움을 줄 전략에 대해 소개합니다. 시스템을 오랫동안 오작동 없이 유지하기 위해 해야 할 일상적인 작업과 문제가 발생할 경우 문제 해결 방법을 이해할 수 있게 될 것입니다. 이 장에서는 다음과 같은 내용을 다룹니다.

- **Informix 인스턴스를 만들기 전에 고려해야 할 사항**

  디스크 사용, 메모리, CPU 및 데이터베이스 구성에 대한 장기적인 계획이 훌륭하게 구성된 시스템을

- **데이터베이스를 조정하고 검사하는 방법**

  시스템이 계속해서 최적의 상태를 유지하면서 오류 없이 작동하도록 해주는 여러 가지 유용한 방법들이 있습니다. 이러한 작업은 일 주일에 한 번 또는 한 달에 한 번 정도 실행해 주면 됩니다.

- **문제를 해결하는 방법**

  문제를 해결하다 보면 대부분 많은 것을 배우게 됩니다. 그러나, 사태에 대해 심사숙고하고 기본적인 절차를 이해하게 되면 보다 쉽게 문제의 원인을 파악할 수 있습니다.

건전한 환경 구축

"예방이 최선의 치료법이다"라는 말이 있습니다. Informix 서버 관리에서도 마찬가지입니다. Informix 제품을 설치하고 온라인으로 연결하고 데이터베이스를 생성하는 것은 그리 어려운 일이 아닐 수도 있습니다. 그러나 그것은 단지 시작에 불과합니다. 장기적인 안목에서 성공의 열쇠는 항상 필요한 상황에 맞게 끊임없이 조정할 수 있는 환경을 구축하는 것입니다. 이 단원에서는 데이터베이스의 미래를 위해 필요한 몇몇 방법들을 소개할 것입니다.

있습니다(**onconfig** 파일).

- 루트(ROOTPATH)를 비롯하여 DB 영역의 크기와 위치
- 동시에 사용가능한 로크(LOCKS) 수
- 논리 로그와 물리 로그의 위치와 크기(LOGFILES, LOGSIZE, PHYSDBS, PHYSFILE)
- 임시 DB 영역(DBSPACETEMP)의 위치와 크기
- 버퍼와 LRU의 구성과 기타 공유 메모리 매개변수(BUFFERS, CKPTINTVL, CLEANERS, LRU_MAX_DIRTY, LRU_MIN_DIRTY, LRUS)
- 시스템의 CPU 수와 Informix의 CPU 사용법(AFF_NPROC, AFF_SPROC, MULTIPROCESSOR, NUMAIOVIPS, NUMCPUVPS, SINGLE_CPU_VP)
- 시스템과 Informix 인스턴스의 메모리 양(SHMADD, SHMTOTAL, SHMVIRTSIZE)
- DSS(예를 들면, 데이터 웨어하우스) 응용 프로그램에 대한 요구 사항(DS_MAX_QUERIES, DS_MAX_SCANS, DS_TOTAL_MEMORY)

물론, 실제로 시스템에서 운영되기 전에 이 매개변수들에 대한 완벽한 값을 설정하기는 거의 불가능합니다. 필요할 경우 이러한 매개변수들을 감시하고 변경할 수도 있으며 이에 관한 방법도 이 글에서 다루고 있습니다. 여기서는 인스턴스 생성 초기에 구성을 잘 함으로써 나중에 해야 할 작업량과 그에 따른 시간을 상당히 줄일 수 있다는 점만 기억해 주십시오. 초기 설정을 적절하게 할 경우 앞으로 계속되는 유지 보수 과정은 고장을 수리하는 과정이라기 보다는 오히려 시스템을 향상시키는 과정이 될 것입니다.

주의*:*

새로운 Informix 인스턴스를 생성하는 과정에는 반복하기 어려운 많은 단계가 포함될 수도 있습니다. 물론, 어떤 작업을 했는지 메모할 수도 있지만 사정상 인스턴스를 재생성해야 할 경우, 한 단계라도 빠뜨릴 수가 있을 것입니다. **onmonitor** 를 사용하거나 수동으로 모든 명령을 입력하기보다는, 셸 스크립트를 작성하여 초기 DB 영역과 기타 매개변수를 만들 것을 권장합니다. 예를 들면, 다음의 스크립트 내용은 2 기가바이트 크기의 새로운 DB 영역을 추가하고 그것에 청크를 추가하는 내용입니다.

```
onspaces -c -d dbspace1 -p /dev/disk1 -o 0 -s 2000000
```

```
onspaces -a dbspace1 -p /dev/disk2 -o 0 -s 2000000
```

스크립트에는 시스템에 필요할 것으로 생각되는 모든 DB 영역과 청크를 포함해야 합니다. Informix 인스턴스를 추가하거나 변경할 때마다 스크립트를 반드시 업데이트해야 합니다. 이렇게 하면, 인스턴스를 재생성해야 할 경우 스크립트를 실행하기만 하면 됩니다.

하드웨어와 운영 체제에서 가장 최신의 Informix 버전을 사용하는 것이 가장 바람직할 것입니다. 모든 버전은 지난 버전에서 지속적으로 결함을 고치면서 향상된 것입니다. 제품의 가장 최신 버전에 관한 정보는, 고객 지원부로 전화하거나 Informix 웹 사이트를 방문하십시오.

경보 프로그램 설치

Informix 는 문제가 발생할 경우 적절한 조치를 취할 수 있는 프로그램이나 셸 스크립트를 제공할 수 있도록 해 줍니다. 특정 Informix 제품이나 운영 체제에 오류가 발생하면 이 프로그램이 자동적으로 실행됩니다. 이 프로그램의 완전 경로명은 **onconfig** 파일에서 ALARMPROGRAM 매개변수로서 제공됩니다. 이 프로그램을 사용하여 데이터베이스 관리자(DBA)를 호출하거나 시스템 콘솔에 메시지를 표시할 수 있습니다. 이 프로그램은 표 1 에 요약된 순서대로 매개변수를 전달합니다.

요약	설명

ID)	번호와 설명이 요약되어 있습니다. 해당되는 오류 메시지의 내용은 다음 매개변수 "클래스 메시지"에 설명되어 있습니다.
>클래스 메시지 (Class Message)	>클래스 ID에 대해 설명하는 메시지 텍스트. 일례로 "완전히 백업된 논리 로그가 필요합니다"가 있습니다.
>특정 메시지 (Specific Message)	>발생된 오류에 대한 보다 자세한 정보. 이것은 Informix 로그 파일에 쓰여지는 것과 동일한 메시지일 수도 있습니다.
>추가 정보 경로 (Extra Info Path)	>오류에 대한 자세한 정보를 포함하는 파일의 경로명.

표 1: ALARMPROGRAM 에 전달된 값.

예를 들어 최소 3 레벨의 오류에 대해서는 Informix 관리자에게 전자 우편을 보내고, 4 와 5 레벨의 오류에 대해서는 Informix 관리자를 호출하도록 하고자 할 경우, 목록 1 과 같이 셸 스크립트를 작성하고 ALARMPROGRAM 에서 이를 가리키도록 설정할 수 있습니다.

목록 1: 샘플 경보 프로그램.

```
#/bin/sh
SEVERITY=$1                              # 이 셸 스크립트의 첫 번째 인수
CLASS_ID=$2
CLASS_MSG=$3
SPECIFIC_MSG=$4
INFO_PATH=$5
# /home/Informix/dba_email_list.txt 파일에 기초하여 DBA 전자
# 우편 목록 설정.
# 이 목록에 있는 모든 사용자에게 전자 우편이 보내질 것임.
# 목록을 변경하려면 이 파일만 편집하면 됨.
```

```
DBA_EMAIL_LIST=`cat /home/Informix/dba_email_list.txt`
if [$SEVERITY -ge 4]                    # 4보다 크거나 같은 Severity-긴박한 상황
then
   SEV_MSG="**THIS ERROR REQUIRES IMMEDIATE ATTENTION!**"
else
   SEV_MSG="WARNING:"
fi
# datetime과 오류 메시지 정보를 MY_MSG 변수에 둠
# 임시 파일에 메시지를 씀
echo "
Informix MESSAGE AT 'date':
$SEV_MSG
SEVERITY LEVEL: $SEVERITY
CLASS ID: $CLASS_ID CLASS MSG: $CLASS_MSG
SPECIFIC MSG: $SPECIFIC_MSG
EXTRA INFO: $INFO_PATH" > /tmp/$$.log
if [$SEVERITY -ge 3]                    # 3보다 크거나 같은 Severity
then
# DBA 목록으로 전자 우편 발송
email $DBA_EMAIL_LIST > /tmp/$$.log
if [$SEVERITY -ge 4]                    # 4보다 크거나 같은 Severity
then                                    # 응답 전화 번호와 함께 dba를 호출하는
  dba_pager.sh 5551212911               # 셸 스크립트 실행
  fi
fi
```

```
# 시스템 로그 파일에 씀
cat /tmp/$$.log >> /usr/Informix/logs/err_msg.log
/bin/rm /tmp/$$.log                    # 지우기
```

목록 1: 샘플 경보 프로그램(계속).

경보를 보낼 상황이 발생하면, 이 셸 스크립트가 호출되어 적절한 이벤트가 일어납니다. 이 스크립트를 설치하면 사람들에게 즉시 통보되어, 문제를 해결하고 유해한 시스템 중단을 방지할 수 있습니다.

일상적인 Informix 작업 수행

관리자가 데이터베이스 서버의 성능을 정기적으로 감시하기 위해 할 수 있는 일들이 몇 가지 있습니다. 정기적으로 데이터베이스와 서버를 체크함으로써 시스템 성능의 병목 현상과 보다 심각한 문제들을 방지할 수 있습니다. 이 단원에서는 정기적으로 수행할 수 있는 작업들을 대략적으로 설명할 것입니다. Informix 의 기술 지원 엔지니어 Clem Akins 의 경험에서 시스템을 조정하는 과정에 대한 몇 가지 기본적인 비결을 얻을 수 있습니다.

> "시스템의 사용 내역을 숫자로 표시할 수 있는 자동화된 도구를 사용하는 것이 중요합니다. 2 주전에 우리가 설정을 변경했기 때문에 응용 프로그램 속도가 반으로 줄었다고 불평하는 IT 관리자로부터 전화를 받은 고객 사이트를 방문한 적이 있었습니다. 응용 프로그램 속도를 나타내는 그래프를 보았을 때, 일일 그래프는 거의 수평한 기울기였고 전혀 시스템의 성능 손실을 나타내지 않았습니다. 잠깐 동안 보통 때의 두 배 정도 나쁜 날이 있었지만 그것조차도 우리가 변경하기 전의 속도만큼 낮은 속도는 아니었습니다.
>
> 이 이야기의 교훈은 시스템을 조정하는 데 있어서 본능적인 직관을 믿어서는 안되며, 과학적인 근거를 기본으로 해야 한다는 것입니다. 컴퓨터 프로그램을 통해 데이터를 수집하고, 통계를 그래프로 나타내고, 양호한 시간대와 불량한 시간대를 추적하며, 시스템에 대해 측정 가능한 결과에 대한 변경 내용을 도표화해야 합니다."

배너 라인 사용하기

Informix 인스턴스의 상태를 빠르게 검사하는 방법은 "onstat -" 명령을 실행하는 것입니다. 이 명령은 현재

상태를 표시하는 간단한 배너 라인을 나타냅니다. **onstat** 명령 중에는 배너 라인을 표시하는 것들이 많이 있습니다. 다음 예는 **onstat -** 명령을 실행한 출력 결과입니다.

```
Informix OnLine Version 7.24.UC7 - - On-Line - - Up 2 days
13:58:15 - - 10336 Kbytes
```

여기에는 다음과 같은 정보가 포함되어 있습니다.

- Informix 버전: 7.24.UC7
- 인스턴스 상태: On-Line
- 인스턴스가 온라인으로 있었던 시간: Up 2 days 13:58:15
- 인스턴스에 의해 사용된 메모리 양: 10336 Kbytes

"**onstat -**" 명령의 결과가 화면에서 여러 줄에 표시될 수 있습니다. 이것은 Informix 가 검사점, 긴 트랜잭션, 매체 오류 및 기타 이유로 인해 "차단"되었을 경우를 나타냅니다.

사용자 이용하기

정규적으로 시스템을 사용하는 사람들은 시스템의 작동에서 나타나는 어떤 변화도 대부분 쉽게 파악합니다. 그들이야말로 질의를 수행하고 정상적인 응답 시간이 평소와 다름을 알아차릴 사람들입니다. 시스템의 작동에 있어서 다른 점이 발견되었다는 말을 시스템 사용자들에게서 듣지 않았다면, 이제부터라도 가끔 그들에게 물어보는 것이 좋을 것입니다. Informix 인스턴스에서 문제점을 발견하지 못한다면, 조정이 필요한 질의 또는 프로그램을 찾아야 할 것입니다.

onmonitor 사용하기

onmonitor 명령은 **onstat** 및 이 글에서 다루는 다른 여러 명령들에 대한 프론트 엔드 툴입니다. 이 명령을 사용할 경우 각 명령의 구문을 기억할 필요는 없지만, 제공되는 정보는 다소 제한적입니다. 이 명령은 사용자에게 다양한 상태 정보를 제공하고 관리자에게는 관리 기능을 수행할 수 있도록 해줍니다. 그러나 가장 중요한 명령들(onstat -d, onstat -u, onstat -g sql, etc.)의 명령줄 버전은 배워야 할 것입니다. 여기에서는 이 글의 목적상 **onmonitor** 의 정보 부분만을 다룰 것입니다. 시작하려면 UNIX 명령줄에서 **onmonitor** 를 입력하십시오. *Informix* 또는 *root* 사용자일 경우 첫 번째 메뉴에서 "Status"를 선택하십시오.

메뉴 옵션은 다음과 같이 요약할 수 있습니다.

- Profile

  Informix 인스턴스에 대한 작업 프로필을 나타냅니다.

  이것은 onstat -p와 상당 부분 동일한 정보를 제공합니다.

- Userthreads

  현재의 활성 사용자 스레드에 대한 정보를 나타냅니다.

  이것은 onstat -u 명령을 실행한 결과와 매우 비슷합니다.

- Spaces

  현재 인스턴스의 모든 DB 영역에 대한 정보를 나타냅니다.

  이것은 onstat -d 명령을 실행한 결과와 비슷하지만 보다 편리한 방법으로 DB 영역의 청크를 볼 수 있습니다.

- Databases

  현재 인스턴스에 모든 데이터베이스를 표시합니다. 데이터베이스 이름, 소유자, DB 영역, 생성일과 로그 상태가 포함됩니다. 이 정보는 onstat 명령을 통해 직접 얻을 수 없는 것입니다. 목록 2의 예를 참조하십시오.

- Logs

  현재의 물리 로그와 논리 로그 상태를 나타냅니다.

  이것은 onstat -l 명령과 같습니다.

- Archive

  마지막 레벨 0, 1 및 2 보관(archive)에 대한 정보를 나타냅니다.

  이것은 oncheck -pr을 통해 얻을 수 있지만 찾기가 더욱 어렵습니다.

- Data-replication

  onconfig 파일에 있는 매개변수를 비롯하여 현재 데이터 복제 매개변수에 대한 정보를 제공합니다.

- Output

  상태 정보를 받을 파일을 선택할 수 있습니다. .

- Configuration

  onconfig 파일을 표시할 파일을 선택하도록 합니다. 이 출력 결과는 onconfig 파일을 읽거나 onstat

-c를 사용하여 얻을 수 있습니다.

- **Exit**

  메뉴를 종료합니다.

Press ESC to return to the Status Menu. Use arrow keys to move the cursor.

DATABASES

Database Name	Owner	In Dbspace	When Created	Log Status
sysmaster	Informix	rootdbs	01/01/97	B
stores7	Informix	rootdbs	09/17/97	N
warehouse1	ron	rootdbs	11/16/96	N

목록 2: onmonitor 에서 Databases 명령의 출력 결과.

메시지 로그 감시

Informix 메시지 로그는 Informix 인스턴스에 대한 메시지를 추적하는 파일입니다. 이것은 **onconfig** 파일의 MSGPATH 매개변수에 의해 지정된 파일에서 찾아볼 수 있습니다. Informix 의 각 인스턴스는 서로 다른 메시지 파일을 사용합니다. Informix 는 서버 성능에 대한 메시지가 포함된 파일을 업데이트합니다. 다음과 같은 내용이 포함되어 있습니다.

- 검사점의 시간과 지속 시간.
- Informix가 시작되었거나 종료된 시점에 대한 정보.
- Informix를 종료시킨 치명적인 오류에 대한 정보.
- 시스템 백업과 복원에 대한 정보.
- 논리 로그 백업 상태.
- **onconfig** 파일의 매개변수에 대한 변경 사항.

메시지 파일은 삭제하거나 보관할 때까지 계속해서 커질 것입니다. 작업 기록을 시스템에 보관하려면, 이전의 메시지 로그 파일을 압축하여 저장하는 것이 종종 가장 바람직합니다.

Informix 인스턴스에서 오류가 발생하면 로그 파일에 많은 정보가 포함됩니다. 이 정보는 Informix 기술

지원에 유용합니다. 때로는 로그 파일에 덤프와 기타 정보 파일 뿐만 아니라 문제를 해결하는 방법에 대해 권고하는 내용이 포함되어 있을 것입니다. 다음은 심각한 시스템 문제를 표시하는 로그를 출력한 예입니다.

10:51:10	Assert Failed: Page Check Error in btcurrent:bad current node

```
0x080b37a8    scan_next
0x080c49bd    hjoin_open
0x080b5d43    prepselect
0x0815f0ab    open_cursor
0x081a7241    ip_scurstart
0x081a79e4    ip_evalcursor
0x081aa1c7    ip_fetch
0x080b4a71    getrow
0x0816e555    sqmain
```

이 출력 결과는 난해해 보이지만 유용합니다. 화면에 출력된 정보는 다음과 같은 내용을 나타냅니다.

- 오류 유형: Page Check Error in btcurrent:bad current node
- 사용자 이름, 세션 및 스레드 :Who:Session(131, ron, 23994, 554376032) Thread(1371, sqlexec, 21090fec, 4)
- 테이블 이름: 'xx01abcd:"Informix".customers'
- 조정 가능한 작업: Action: Run 'oncheck -cDI 6449916"
- 보다 많은 정보가 포함된 파일: /DUMPDIR/af.56cb0d, gcore.56cb0d.0, /DUMPDIR/prob/core
- 스택 추적: 1402 sqlexec base: 0x21a66010

로그 파일은 Informix 기술 지원에 사용할 수 있습니다.

메시지 로그의 메시지들은 지속적으로 감시하는 것이 좋습니다. 다음과 같은 여러 방법들을 사용할 수 있습니다.

1. **onstat -m** 명령을 사용합니다. 이것은 모든 것이 정상적인지를 신속하게 알아 볼 수 있는 방법입니다. 이 명령은 또한 메시지 로그 파일에 대한 완전 경로명을 나타냅니다. 여기에서는 메시지 파일의 마지막 10줄만 나타난다는 점에 유의해야 합니다. 나머지 줄들은 텍스트 편집기나 다른 명령을 통해 볼 수 있습니다. 다음은 이 명령을 출력한 예입니다.

```
Message Log File: /usr/Informix/online.log

Wed Sep 17 00:01:28 1997


00:01:28 Checkpoint Completed: duration was 0 seconds.

00:31:05 Checkpoint Completed: duration was 1 seconds.

01:01:06 Checkpoint Completed: duration was 1 seconds.

01:01:25 Level 0 Archive started on rootdbs, dbspace1

01:31:27 Checkpoint Completed: duration was 0 seconds.

02:01:27 Checkpoint Completed: duration was 0 seconds.

02:02:12 Archive on rootdbs, dbspace1 Completed.

02:02:58 Logical Log 95 Complete.

02:31:28 Checkpoint Completed: duration was 0 seconds.
```

- 파일을 분석할 수 있는 다른 운영 체제 명령을 사용합니다. 예를 들면, UNIX에서는, **grep**, **tail**, **lp** 등의 명령이 있습니다.

- 텍스트 편집기에서 파일을 읽고 특정 메시지를 검색합니다.

- 특정 오류 메시지에 대해 파일을 감시하는 운영 체제 프로세스를 만듭니다. 이것은 항상 실행되는 백그라운드 작업으로서 만들 수 있습니다(UNIX에서는 **cron**을 사용할 수 있습니다). 예를 들면, 다음 스크립트는 로그에서 "Assert Failed"가 나타나는지 지켜보고 있다가 이 메시지가 나타나면 dba_list로 전자 우편을 보내는 것입니다.

```
#!/bin/sh

while [ true ]

do      x=`tail /usr/Informix/online.log | grep "Assert Failed"`

        if [ "$x" ]

        then

      echo ASSERT FAILED FOUND IN ERROR LOG! | mail dba_list
```

```
        fi
      sleep 60
done
```

- 검사점의 간격과 지속 시간을 지켜봅니다. 간격은 **onconfig** 매개변수 CKPTINVL로 지정합니다. 검사점의 지속 시간이 2초 이상 되거나 더 길어지기 시작하면, 이 때가 엔진을 조정해야 할 시기입니다. 또한 검사점의 간격이 CKPTINVL보다 짧아지기 시작하면 물리 로그 매개변수를 조정해야 할 것입니다. 앞의 예에서 알 수 있듯이, 검사점 간격은 대략 30분이고 지속 시간은 0 또는 1초입니다. 간격이나 지속 시간이 변경된다면 조정이 필요할 것입니다. 현재 로그 파일의 모든 검사점 간격을 살펴보려면 다음의 UNIX 명령을 실행할 수 있습니다.

```
grep /usr/Informix/online.log "Checkpoint Completed"
```

또는 마지막 20 개의 검사점을 보려면, tail 명령을 사용합니다.

```
grep /usr/Informix/online.log "Checkpoint Completed" | tail -20
```

- 논리 로그가 너무 빠르게 채워지고 있지 않은지 살펴봅니다. 예를 들면, "Logical Log xx Complete" 라는 메시지가 너무 자주 나타나면 논리 로그의 수 또는 크기를 변경해야 할 것입니다.

시스템 성능 프로필 감시

Informix 에서는 onstat -p 명령으로 다양한 성능 통계를 한 눈에 볼 수 있습니다. 다음은 출력 예제입니다.

```
Profile
```

ovlock	ovuserthread	ovbuff	usercpu	syscpu	numckpts	flushes	
0	0	0	107.22	92.05	175	9590	
bufwaits	lokwaits	lockreqs	deadlks	dltouts	ckpwaits	compress	seqscans
208	0	163575	0	0	34	259	115
ixda-RA	idx-RA	da-RA	RA-pgsused	lchwaits			
308	9	35	313	6			

여기에는 다음과 같은 항목들이 포함되어 있습니다.

- 디스크 읽기 수: dskreads
- 디스크 읽기 캐시 비율(일반적으로 90% 대의 중간 정도여야 합니다): %cached
- 디스크 쓰기 수: dskwrits
- 디스크 쓰기 캐시 비율(일반적으로 80% 대의 중간 정도여야 합니다): %cached
- 다른 항목의 오버플로(0 또는 0에 가까워야 합니다): "ov"가 들어 있는 제목
- 대기(적당한 수치여야 합니다): "waits"가 들어 있는 제목
- 순차적 스캔(너무 많을 경우 색인이 필요할 수도 있습니다): seqscans

프로필 정보를 사용하면 시스템을 한 눈에 볼 수 있습니다.

논리 로그와 물리 로그의 상태

논리 로그와 물리 로그는 시스템의 일관성을 유지하는 데 사용됩니다. 이 로그의 현재 상태는 onstat -l 명령을 실행하면 다음과 같이 볼 수 있습니다.

```
Physical Logging
```

Buffer	bufused	bufsize	numrecs	numpages	numwrits	recs/pages	pages/io
L-2	0	16	2654	285	241	9.3	1.2
address	number	flags	uniqid	begin	size	used	%used
a1b3ee6	1	U-B--	61	100300	250	250	100.00
a1b3ee7	2	U--C-L	62	100550	250	249	99.60
a1b3ee8	3	U-B--	57	100800	250	250	100.00

주의 깊게 보아야 할 정보는 다음과 같습니다.

- 물리 로그 쓰기와 입출력 정보: numwrits, pages/io
- 사용된 물리 로그 정보(값이 너무 높으면 물리 로그 매개변수의 조정을 고려해야 합니다): phyused (pages)와 %used
- 플래그에 의해 표시되는 논리 로그 상태. 백업되지 않고 채워지는 로그가 너무 많을 경우, 논리 로그의 크기와 수를 변경해야 합니다. 로그는 B 상태일 때 백업됩니다.
- 사용된 비율. 백업되지 않은 로그(B 플래그)에 대해 높은 수치가 나타나면, 직접 로그를 백업해야 하며 로그 파일 매개변수를 변경할 필요가 있다는 뜻입니다.

논리 로그의 사용법은 논리 로그의 백업 방법과 데이터베이스 로깅 방법에 따라 다릅니다. 데이터베이스에 트랜잭션(로그된)이 많으면 많을수록, 로그는 더 빠르게 채워질 것입니다. 로그 백업 방법의 일환으로써, 연속 로그 백업을 사용할 것인지 자동 로그 백업을 사용할 것인지를 고려해야 합니다.

주의:

> 논리 로그가 절대 꽉 차지 않도록 하는 것이 매우 중요합니다. 논리 로그가 꽉 차게 되면 모든 데이터베이스에 대해 인스턴스의 프로세싱이 멈추게 되며 최악의 경우에는 테이프로부터 모든 데이터베이스를 복원해야 할 수도 있습니다. 로그를 감시하고 적절한 로그 백업 절차를 따른다면 이와 같은 문제가 발생하는 것을 방지할 수 있습니다.

또한 어떠한 단일 업데이트 트랜잭션도 **onconfig** 매개변수인 LTXHWM 과 LTXEHWM 에 나열된 비율보다 더 많은 로그 영역을 차지하지 않도록 해야 합니다. 트랜잭션이 LTXHWM 의 수보다 많은 로그 비율을 차지할

경우, 그 트랜잭션은 롤백될 것입니다.

버퍼 감시

업데이트가 진행되는 동안 물리 로그 버퍼는 데이터의 "이전 이미지(before image)"를 보존하기 위해 물리 로그와 함께 작동됩니다. 이 매개변수들을 적절히 조정하면 Informix 성능을 크게 향상시킬 수 있습니다. 또한 물리 로그와 버퍼는 Informix 의 복구 방법에 있어서 매우 중요한 요소입니다.

물리 로그 버퍼의 조정은 **onconfig** 매개변수인 BUFFERS, LRUS, CLEANERS, LRU_MAX_DIRTY 및 LRU_MIN_DIRTY 와 함께 이루어집니다. BUFFERS 는 LRU 큐의 수로 나누어지는데, 이 수는 LRUS 매개변수에 의해 표시됩니다. 각 큐에는 사용가능한(f) 버퍼와 수정된(m) 버퍼가 있습니다. 페이지 클리너(CLEANERS)는 버퍼가 너무 많이 찼을 때 큐를 플러시할 수 있게 해줍니다. 수정된 버퍼의 비율이 LRU_MAX_DIRTY 의 비율을 초과하면, 페이지 정리가 시작됩니다. 이 페이지 정리가 제대로 조정되지 않으면, Informix Dynamic Server 의 성능이 저하될 수도 있습니다. **onstat -R** 명령은 버퍼 사용을 감시하는 데 사용되며 다음과 같이 나타납니다.

```
8 buffer LRU queue pairs
```

9	m	0	0.0%	
10	f	500	100.0%	500
11	m	0	0.0%	
12	f	500	100.0%	500
13	m	0	0.0%	
14	f	500	100.0%	500
15	m	0	0.0%	

0 dirty, 4000 queued, 4000 total, 4096 hash buckets, 2048 buffer size

start clean at 60% (of pair total) dirty, or 300 buffs dirty, stop at 50%

수정된(m) 버퍼의 비율이 LRU_MAX_DIRTY 에 가까워지게 되면, 페이지 정리 작업이 더 활발히 이루어질 것입니다.

검사점 작업도 버퍼의 영향을 받습니다. 버퍼가 너무 "더티(dirty)"하면, 버퍼를 정리하기 위해 검사점이 더 길어질 것입니다.

Informix Dynamic Server 에서 버퍼 풀을 플러시할 때 실행되는 세 가지 유형의 쓰기가 있습니다. 가장 비효율적인 것에서 가장 효율적인 것의 순서로 소개를 하면, Foreground, LRU 그리고 Chunk 가 그 세 유형입니다. 이것들은 다음과 같이 **onstat -F** 명령을 사용하여 감시할 수 있습니다.

Fg Writes	LRU Writes	Chunk Writes	
0	6	2403	
address	flusher	state	data
a262334	0	I	0 = 0X0

states: Exit Idle Chunk Lru

버퍼 플러싱 작업을 조정하는 것은 많은 실험과 조정이 요구되는 까다로운 일입니다. 한 번에 한 개의

매개변수만을 변경한 후에 이러한 감시 도구들을 사용하여 변경 내용을 미세하게 조정할 것을 권장합니다.

맺음말

이 장에서는 Informix 서버를 지속적으로 유지하는 프로세스에 대해 설명했습니다. 서버 설정과 관련된 기본적인 명령들과 미래의 지속적인 안정 운영을 위한 준비에 관해서도 생각해 보았습니다. 서버에 문제가 발생하는지 또는 병목 현상이 있는지 감시하기 위해 지속적으로 수행할 수 있는 몇 가지 명령도 함께 다루었습니다.

이러한 내용들이 Informix DB Server 를 사용하는 모든 운영자들에게 도움이 되기를 바랍니다.