로컬 서열 정렬과 트리거 기반의 단백질 버전 정보 관리 기법

정 광 수[†] · 박 성 희[††] · 류 근 호[†††]

요 약

하나의 아미노산 서열의 기능이 밝혀지면, 그와 유사한 서열 구조를 가지고 있는 서열의 기능도 유추해 낼 수 있다. 또한 기능이 밝혀진 단백질의 아미노산 서열을 변화시키거나 유용한 단백질을 만드는 것도 가능하다. 이 과정에서 하나의 원본 단백질 서열에 대하여 다른 서열 구성을 가지고 있는 여러 가지 단백질 서열이 생겨 날 수 있다. 여기서, 원본 단백질을 변화시켜 만든 단백질 버전 서열과 단백질의 주석정보를 저장 및 관리하는 체계적인 기법이 요구된다.

따라서 이 논문에서는 로컬 서열 정렬 기법을 적용한 단백질 아미노산 서열의 버전관리 기법과 트리거를 적용한 단백질 주석데이터의 이력 관리 기법을 제시하였다. 제안된 기법을 통하여 원본 서열과 버전서열의 유사도 측정 및 버전 관리의 자동화와 저장 공간을 감소시킬 수 있다. 또한 단백질 정보의 이력을 저장하고 서열 변화 정보를 분석하여 돌연변이 연구에 의한 유용한 단백질 개발 및 신약 개발이 가능하다.

A management Technique for Protein Version Information based on Local Sequence Alignment and Trigger

Kwang-Su Jung[†] · Sung-Hee Park[††] · Keun-Ho Ryu[†††]

ABSTRACT

After figuring out the function of an amino acid sequence, we can infer the function of the other amino acids that have similar sequence composition. Besides, it is possible that we alter protein whose function we know, into useful protein using genetic engineering method. In this process, an original protein amino sequence produces various protein sequences that have different sequence composition. Here, a systematic technique is needed to manage protein version sequences and reference data of those sequences.

Thus, in this paper we proposed a technique of managing protein version sequences based on local sequence alignment and a technique of managing protein historical reference data using Trigger. This method automatically determines the similarity between an original sequence and each version sequence while the protein version sequences are stored into database. When this technique is employed, the storage space that stores protein sequences is also reduced. After storing the historical information of protein and analyzing the change of protein sequence, we expect that a new useful protein and drug are able to be discovered based on analysis of version sequence.

키워드 : 바이오인포메틱스(Bioinformatics), 단백질(Protein), 버전관리(Version Management), 갱신 시스템(Update System)

1. 서 론

90년대 인간 유전체 사업(Human Genome Project)이 시작되면서부터 시퀀싱 기술의 급격한 발달로 유전체 서열 데이터들이 급증하고 있다. 그리고 2000년 6월, 30억 개의 Base Pair(A, G, T, C)로 이루어진 인간 유전자 지도가 완성되었다. 무작위로 배열된 것처럼 보이지만 여기에 모든 생명의 신비와 비밀이 숨어 있다. 이 속의 유전정보를 해독해 내는 것이 이제부터 시작될 포스트 지놈(Post Genome)시대의 과제다. DNA 염기 서열은 RNA 전사를 거쳐 단백질 아미노산 서열로 번역되며, 비로소 단백질을 통하여 기능이 발현되게 된다. 하나의 아미노산 서열의 기능이 밝혀지면, 그와 유사한 구조를 가지고 있는 서열의 기능 또한 유추해 낼 수 있다. 또한 기능이 밝혀진 단백질의 아미노산 서열을 유전 공학 기법을 이용하여 변화시키거나 새로운 잔기 서열을 만들어 유용한 단백질을 만드는 것도 가능하다. 이 과정에서 하나의 원본 단백질 서열에 대하여 다른 서열 구성을 가지고 있는 여러 가지 단백질 서열이 생겨 날 수 있다. 이러한 연구는 돌연변이 및 신약 개발 연구에 응용되고 있다. 현재 NCBI[4]를 제외한 대부분의 생물학 데이터베이스

※ 이 연구는 KISTEP 특정 연구개발과제, KISTI 바이오인포메틱스 센터 및 한국과학재단 RRC(청주대 ICRC)의 연구비 지원으로 수행되었음.
† 준 회 원 : 충북대학교 데이터베이스/바이오인포메틱스 연구실
†† 준 회 원 : 충북대학교 데이터베이스/바이오인포메틱스 연구실
††† 종신회원 : 충북대학교 전기전자컴퓨터공학부 교수
논문접수 : 2003년 12월 5일, 심사완료 : 2004년 2월 5일

는 가장 최신 버전의 데이터만을 가지고 있다. 또한 NCBI의 버전 관리의 경우 버전 정보를 저장 할 때 중복 저장되어 시스템의 자원을 낭비를 초래한다. 그리고 원본 및 버전 서열간의 유사도를 측정할 때 별도의 도구를 사용해야 한다. 여기서, 데이터의 중복성을 제거하여 저장 공간을 현저히 줄이고, 서열간의 유사도 측정 원활히 하기 위하여 단백질 버전 정보를 효율적으로 저장 및 관리하는 체계적인 기법이 요구된다. 또한 NCBI에서 이러한 버전 정보 관리는 수작업으로 이루어져 많은 비용이 소모되므로 이를 자동화하는 작업이 요구된다.

따라서 이 논문에서는 단백질 버전 정보 데이터를 데이터베이스에 저장할 때, 저장 공간을 줄이는 것은 물론 버전 관리와 동시에 유사도 측정이 가능하게 하였다. 또한 수작업으로 이루어지던 버전 정보 관리를 자동화하였다. 단백질 데이터는 크게 단백질 서열 데이터와 단백질 주석 데이터로 분류 할 수 있다. 이 논문에서는 위 단백질 데이터의 두 가지 분류에 각각 다른 버전 관리 기법을 제시하였다.

단백질 서열 데이터는 단백질의 아미노산 서열 자체를 말하는 것으로써 Local similarity 분석 방법 중 Smith-Waterman 알고리즘[26]을 사용하여 서열의 버전 가능성 여부를 판별하고 데이터베이스에 저장한다. 그리고 서열 전체를 저장하는 것이 아니라 과거 서열과의 차이점을 분석하여 스크립트를 만든 뒤 스크립트를 데이터베이스에 저장한다. 이렇게 함으로써 원본 서열과 버전 서열간의 유사도를 측정할 수 있고, 데이터베이스의 저장 공간을 감소시킬 수 있다.

단백질 서열 주석 데이터는 단백질 서열 데이터를 제외한 단백질 정보로써 단백질 서열의 생명체 정보, 저자 정보, 서열 데이터의 실험 정보 등의 주석 데이터를 말한다. 주석 데이터의 버전 관리는 능동데이터베이스[9]의 트리거[16]를 이용한다. 트리거는 단백질 식별자에 대해 데이터가 변경된 부분만 갱신 로그 테이블에 저장되며, 갱신 로그 테이블의 정보를 사용하여 데이터가 변화된 이력을 추적 할 수 있다. 트리거를 사용함으로써 응용 프로그램에서의 원본 소스 수정에 따른 버전관리 메커니즘의 오류를 미연에 방지 할 수 있다. 또한 분산 환경의 경우 응용프로그램과 데이터베이스 서버 사이의 통신량을 줄일 수 있다.

2. 관련 연구

■ NCBI의 서열 버전 관리

생물학 데이터베이스[1, 3, 4, 5, 6, 7]의 서열 버전 관리시스템[13]에서 정확한 서열을 검색하기 위해서 제일 먼저 선행되어야 할 부분은 서열의 식별자를 설정하는 것이다. 서열 레코드는 서열과 그 서열에 대한 주석정보를 포함하고 있고 하나의 레코드 안에 여러 개의 다른 서열 식별자를 갖고 있다. NCBI의 GenBank[5] 경우는 서열 레코드는 추가적으로 "LOCUS" 이름과 "ACCESSION" 넘버를 필요로 한다. GenBank의 모든 서열은 "gi" 넘버를 부여받는다. 이 "gi" 넘버는 하나의 내부 키로써 Entrez[12]에서 모든 서열들을 검색하는데 사용된다.

실험실에서 DNA 조각으로부터 서열을 뽑아내어 데이터베이스에 저장할 때 ACCESSION 넘버가 할당되고 DNA 조각을 표현하게 된다. 그 서열을 포함하고 있는 레코드가 ID(Integrating Database) 데이터베이스에 저장 될 때 처음으로 'gi'가 그 서열에 할당된다. 시간이 흐른 뒤에 다시 실험을 했을 때 그 DNA의 가장 정확하다고 판단된 서열이 바뀔 수 있다. 이처럼 같은 DNA 조각에 대한 새로운 서열이 NCBI에 입력이 되었을 때 새로운 'gi'가 할당된다. ACCESSION은 서열이 변하더라도 변하지 않으므로 원본 DNA 조각을 식별하는 데 사용되며, 'gi'는 서열 유사도등 특정한 서열 자체에 초점을 맞추고 있는 경우 사용된다.

GenBank[5]는 특정한 DNA 조각에서 산출된 서열 변화 정보와 버전 정보를 데이터가 예치되는 시점에 따라 모두 저장한다. 따라서 과거 버전 정보와 중복되어 저장될 수 있다. 또한 과거 서열과의 유사도를 측정하기 위해서는 별도의 유사도 측정 도구를 사용해야 하는 불편함이 있다. 이 논문에서는 Smith-Waterman 알고리즘[26]과 트리 구조 기반의 반 구조 데이터의 변경 정보검출 기법, 그리고 능동데이터베이스의 트리거를 적용하여 이 문제를 해결 하였다. 이로써 버전 데이터의 공간 사용량을 줄이고 버전 관리와 동시에 자동으로 서열의 유사도 측정을 할 수 있게 하였다. 또한 트리거를 사용함으로써 주석 데이터의 버전 관리 메커니즘을 응용프로그램의 개입 없이 가능하게 하였다.

■ 유사성 검색을 사용한 서열 분석 방법

단백질 서열의 변경 검출시 기존의 스트링 문자열을 비교 하는데 사용되는 LCS(Longest Common Subsequence)[18] 알고리즘을 적용하기에는 다소 무리가 있다. 단백질 서열데이터는 일반 스트링과는 달리 유전적인 정보를 담고 있고, 서열의 변화는 생물학적인 특성 변화를 의미하므로 서열 정렬 알고리즘[11, 21, 24, 26, 29, 30]을 이용하는 것이 더 효과적이다.

초기의 유사성 검색 도구가 Needelman & Wunch[30]와 Sellers[29]에 의해 개발되었다. 이들은 비교할 서열들의 전체의 길이에 대한 포괄적인(global) 유사성 점수를 계산하였다. 이러한 형태의 알고리즘은 다양화된 서열에 민감하지 않으므로 유사성 검색을 위해 사용될 방법들은 지역적인(local) 유사성을 가지는 지역에 초점을 맞추어야 한다. 가장 광범위하게 사용되는 Local Alignment 알고리즘은

Smith-Waterman[26], BLAST[21]과 FASTA[24]이다. 알고리즘은 Dynamic Programming[31]을 이용하여 전체 서열에서 유사성 검색을 수행하고, FASTA와 BLAST는 모든 가능한 배열들을 다 조사하지 않는 Heuristic 알고리즘을 이용한다. 실제 검색에 있어서 Smith-Waterman 알고리즘이 FASTA나 BLAST보다 민감한 것으로 알려져 있다. 최근에 개발된 BLAST 2.0[11]은 gap filling 기능이 보강되어 입력한 서열 전체에서의 유사성을 보여주는 기능을 가지고 있다.

이 논문의 경우 단백질 버전 서열의 변환 정보 작성과 검색은 데이터베이스의 모든 서열과 질의 서열과의 유사성을 검색하는 것이 아니라, 원본 단백질 서열과 원하는 시점의 서열과의 유사성만을 검색하여 차이점을 분석해 낸다. 따라서 시간은 다소 길더라도 민감한 Smith-Waterman 방법을 적용하였다.

■ 반구조적 데이터의 변경 검출

단백질 데이터는 단백질의 서열 자체뿐만 아니라 단백질의 생명체에 대한 정보, 실험자 정보, 단백질이 어떤 분류에 해당하는 가를 알 수 있는 클래스 정보, 유전적 기능 정보, 다른 생물학적 데이터베이스와의 Cross Reference 정보, 논문이나 저널에 게재된 정보 등의 여러 가지 정보가 해당된다. 단백질 주석 데이터는 같은 유전체 샘플이라도 여러 가지 실험 환경과 샘플 데이터의 변이에 따라 다르게 생성된다. 이러한 단백질 데이터는 반구조적 데이터이고 이러한 정보를 버전관리에 적용하기 위해서 과거의 반구조적 데이터의 변경 검출 기법 적용이 가능하다.

오래 전부터 LCS 알고리즘을 이용하여 같은 문서의 두 개의 버전사이에서 변경 정보를 검출을 수행하는 메커니즘[27, 28]이 제시되어 왔다. 초창기에는 이러한 연구는 비구조적 문서의 변경을 검출하는데 초점을 맞춰왔다. 여기서는 단지 insertion, deletion, update 연산자만을 고려하였다. 이 연산자들을 사용하여 하나의 문자열에서부터 다른 문자열로 변환하는 과정을 수행할 수 있다. 그 뒤 스키마를 포함하는 데이터의 변화를 검출하는 메커니즘[14, 15]의 경우 문장 안에 구분자를 두어서 데이터의 스키마 정보를 간단히 표현하고, 구분자로 구분된 데이터는 LCS 알고리즘을 통하여 변화정보를 검출하였다. 그러나 이런 경우 스키마 정보의 변화 정보는 검출하지 못하였다. 이런 점을 보완하기 위하여 트리를 구성하여 스키마의 변화를 검출하는 메커니즘[12, 17, 19, 22]이 제시되었다. 이 경우는 두 버전의 데이터와 스키마 정보를 트리 형태로 구성하여 트리 사이의 차이점을 비교한다. 이 메커니즘에서는 트리 사이의 변화 정보를 스크립트로 작성한다. 스크립트는 변경 연산자 insert, delete, update, move, glue의 조합이며, 가장 좋은 성능의 스크립트는 최소 비용이 소비되는 스크립트이다. 그러나 이러한 트리 구조의 변경 검출 방법도 문서 내에 시간에 따라 반복적으로 나타나는 데이터 등의 반구조적 데이터가 가지고 있는 모든 스키마나 룰을 표현하지는 못하였다. 이 점을 보완하기 위해 스키마 그래프를 이용한 변경 검출 방법[8]이 제시되었다.

이 논문에서는 일정한 기간에 FTP로 배포되는 단백질 데이터를 사용하게 된다. 단백질 서열 데이터의 경우는 이전 서열과 현재 서열의 차이점을 검출하게 되고, 단백질 주석 데이터의 경우는 배포되는 시점의 스키마나 데이터 변화 정보만을 다루게 되므로 좀 더 복잡한 방법의 스키마 그래프를 이용한 변경 검출 방법은 관련 연구에서 제외하였다.

3. 단백질 서열 데이터의 버전관리

이 절에서는 주기적으로 PIR-PSD[3]에서 배포하고 있는 XML 파일로부터 단백질 아미노산 서열을 추출하여 버전을 관리하는 것에 대해 주로 설명한다.

3.1 버전 서열의 정의

기능이 밝혀진 단백질의 아미노산 서열을 유전 공학 기법을 이용하여 변화시키거나 새로운 서열을 만들어 유용한 단백질을 만드는 과정에서 하나의 원본 단백질 서열에 대하여 다른 서열 구성을 가지고 있는 단백질 서열이 생겨날 수 있다. 이 때, 원본 단백질 아미노산 서열을 조작하여 생성되는 아미노산 서열이 버전 서열이 되기 위해서는 다음과 같은 정의가 필요하다.

[정의 1] 서열 식별자

어떤 단백질 샘플 D가 있을 때, D로부터 추출한 원본 서열 $S = \{k|k_1k_2k_3k_4k_5\cdots k_n\}$이고 S는 D로부터 상속받은 서열 식별자 Sid(S)를 가지고 있다. 원본 서열 S를 여러 가지 조작 기법으로 변형시킨 서열 $S' = \{k'|k_1'k_2'k_3'k_4'k_5'\cdots k_n'\}$가 존재하고, 서열 S′ 또한 서열 식별자 Sid(S′)를 가지고 있다. 이때, 서열 식별자 Sid(S), Sid(S′)는 서로 동일하여야 한다.

서열의 식별자는 서열이 어떤 단백질 샘플에서부터 추출되었는지를 알아낼 수 있는 정보이고, 배포 파일로부터 서열과 함께 추출한다. 한 번 부여된 서열 식별자는 변경 될 수 없으며, 서열 식별자를 이용하여 데이터베이스에서 해당 서열 식별자를 가진 데이터를 검색하고, 적절한 변경 관련 연산을 수행한다. 따라서 서열의 식별자가 없다면 서열의 버전관리 자체가 불가능하다. 그리고 원본 서열의 식별자와 버전 서열의 식별자가 서로 다르다면 두 서열은 같은 샘플에서 추출 된 것이 아니며, 따라서 버전 서열은 유효하지 않다.

[정의 2] 서열 변경

S와 S′의 길이를 Length(S)와 Length(S′)라 하고, 순서 정보를 Order(S)와 Order(S′)라 하고, 서열의 구성정보를 각각 Composition(S)와 Composition(S′)라 한다. 여기서 S′가 버전서열이 될 수 있는 조건은 다음 세 가지 경우 중 하나 이상을 만족해야 한다.

$$\begin{pmatrix} Length(S) \neq Length(S') \\ Order(S) \neq Order(S') \\ Composition(S) \neq Composition(S') \end{pmatrix}$$

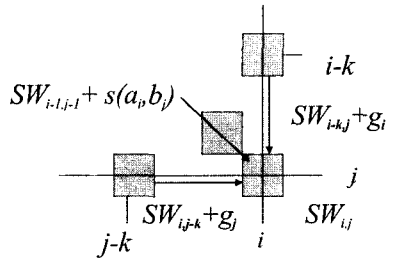
즉, 서열 $S' = \{k'|k_1'k_2'k_3'k_4'k_5'\cdots k_n'\}$은 어떤 샘플 DNA 조각 D로부터 새롭게 생성된 서열이고, 서열집합 $SS = \{S \mid S_1 \cdots S_n\}$는 D로부터 생성된 버전 서열의 집합이라고 할 때, SS의 원소 서열들은 바로전의 서열과 비교하여 아미노산 서열의 구성정보 또는 순서 정보나 길이가 달라야 한다. 여기서 새로운 서열 S′ 또한 SS의 원소 서열들 중 가장 최신의 단백질 버전 서열과 비교하여 베이스의 구성정보 및 순서정보 또는 길이가 달라야 한다.

3.2 Smith-Waterman 알고리즘을 적용한 단백질 서열 데이터의 버전 관리

단백질 서열 데이터의 경우는 서로 다른 두 버전 사이에서 Local Alignment 알고리즘을 적용하여 변화된 정보를 얻어낸다.

$$SW_{i,j} = \max \begin{pmatrix} SW_{i-1,j-1} + s(a_i, b_j), \\ SW_{i-k,j} + kg_j, (k < i) \\ SW_{i,j-k} + kg_i, (k < j) \\ 0 \end{pmatrix}$$

(a)



(b)

(그림 1) SW 알고리즘의 Unit Cost Model

Smith-Waterman 알고리즘은 부분적인 정렬 스코어를 가진 2차원의 테이블을 만들고 테이블 안의 셀은 각 셀의 최적의 부분 정렬을 찾아 낼 수 있는 스코어를 포함하고 있다. 또한 전산학 알고리즘의 한 유형으로 일단 가장 간단한 해답을 알고 있고, 그 해답을 이용해 점점 더 전체로 확장시켜 나가면서 마지막 대답을 얻는 경우에 이용되는 알고리즘인 Dynamic Programing[31] 방법을 사용한다.

(그림 1)은 Smith-Waterman 방법을 사용하여, (그림 2)의 각 셀을 채워 나가는 방법을 설명하고 있다. 각 셀은 (그림 1)의 (a)의 세 가지 후보 값을 가지게 된다. 후보 값 중 최대 값이 해당 셀의 값이 되며, 최대 값이 음수인 경우 해당 셀 값은 0을 취하게 된다. 두 개의 단백질 서열 S={RNMKGITATYLS}와 T={MKRGIIAYLSWP}가 있을 때, S와 T를 각각 행렬의 축으로 하고 일치=5, 치환=-4, 삽입 또는 삭제=-7를 적용한다. 일치, 치환, 삭제에 대한 값은 유사도의 허용 범위에 따라 사용자가 설정 가능하다. (그림 2)에서 셀(S_4, T_2)의 경우 해당하는 S와 T의 아미노산 서열은 K(lysine)으로 같다. 이처럼 각 셀마다 해당되는 아미노산 서열이 같은 경우 (그림 1) (a)의 제일 윗 항에서 $s(a_4,b_2)$ 값은 일치=5이다. (그림 1) (b)에서 알고자 하는 셀의 후보 값이 대각선 방향으로 온 경우이다. 셀(S_3,T_1)의 값은 5이므로 (a)의 제일 윗 항의 값은 10이다. (그림 1) (a)에서 둘째 항의 k 값을 1이라 하면, 셀(S_2,T_1)의 값은 0이고 갭 감점은 -7이므로 (그림 1) (a)의 둘째 항은 -7이지만 0보다 작으므로 후보 값은 0으로 기록된다. (그림 1) (b)에서 수직으로 내려 온 셀의 후보 값이 해당된다. 세 번째 항에서 역시 k값을 1이라 하면, 셀(S_4,T_1)의 값은 0이고 갭 감점은 -7이므로 세 번째 항의 값은 -7이고 역시 0보다 작으므로 후보 값은 0으로 기록된다. (그림1) (b)에서 수평으로 온 셀의 후보 값이 해당된다. 세 항의 후보 값 중 최대 값을 선택하므로 셀(S_4, T_2)의 값은 10이다. 이처럼 (그림 1) 이용하여 각각의 행렬의 항들을 채워나가면, (그림 2)의 결과를 얻을 수 있다.

			SEQUENCE S												
			j	1	2	3	4	5	6	7	8	9	10	11	12
				R	N	M	K	G	I	T	A	T	Y	L	S
SEQUENCE T	i		0	0	0	0	0	0	0	0	0	0	0	0	0
	1	M	0	0	0	5	0	0	0	0	0	0	0	0	0
	2	K	0	0	0	0	10	3	0	0	0	0	0	0	0
	3	R	0	0	0	0	3	6	0	0	0	0	0	0	0
	4	G	0	0	0	0	0	8	1	0	0	0	0	0	0
	5	I	0	0	0	0	0	1	13	6	0	0	0	0	0
	6	I	0	0	0	0	0	0	6	9	2	0	0	0	0
	7	A	0	0	0	0	0	0	0	2	14	7	1	0	0
	8	Y	0	0	0	0	0	0	0	0	7	10	12	5	0
	9	L	0	0	0	0	0	0	0	0	1	3	6	17	10
	10	S	0	0	0	0	0	0	0	0	0	0	0	10	22
	11	W	0	0	0	0	0	0	0	0	0	0	0	3	15
	12	P	0	0	0	0	0	0	0	0	0	0	0	0	8

(그림 2) Smith-Waterman Algorithm with DP

(그림 2)의 결과를 얻었다면 이제 행렬을 바탕으로 최적의 경로를 산출해야 한다. 셀 값 중 높은 점수를 기록한 영역은 유사도가 큰 부분이다. 적절한 갭을 포함하여 정렬된 서열의 경로를 추적할 때 (그림 2)의 셀에서 높은 값 영역 중 셀 값 22의 (S_{12}, T_{10})부터 시작한다. 경로 추적은 오른쪽 아래에서 왼쪽 위의 대각선방향으로 셀을 선택하게 된다. 선택 방법은 왼쪽, 대각선 또는 위쪽의 셀 값 중 해당 셀의 값이 온 방향의 셀을 선택한다. (그림 2)의 회색 바탕의 셀들은 경로추적의 결과로 선택된 셀들을 나타내고 있다. 서열 S에 대하여 대각선은 일치 혹은 치환을, 수평이동은 삽입을, 수직 이동은 삭제를 나타낸다. 때때로 적절한 배열을 위한 최소한의 경로는 단 한 개가 아니고 여러 개가 될 수 있다. 대각선에 의해 표시된 행로에 의해 S와 T를 배열하면 (그림 3)과 같다. 기존의 Smith-Waterman 알고리즘은 유사도가 높은 부분을 서로 매치 하여 두 서열을 정렬하는 것이므로 (그림 3)의 결과를 얻음으로써 처리과정을 종료한다.

```
S = MK-GITATYLS
T = MKRGIIA-YLS
```

(그림 3) Smith-Waterman 서열 정렬 결과

이제부터 (그림 3)의 결과를 버전 관리에 적용하기 위한 과정을 설명한다. 서열 S의 하위 조각 서열 {RN}과 서열 T의 하위 조각 서열 {WP}는 매핑 결과에 포함되지 않는다. 하지만, 서열 T의 제외된 하위 서열은 추후에 버전을 검색할 때 필요한 정보이므로 데이터베이스에 제외된 서열 정보를 저장해야 한다. (그림 3)과 같은 서열 정렬 결과를 얻었다면 이제 서열 S에서부터 서열 T로 변환하는 스크립트를 작성하여야 한다. 서열의 변화에 필요한 연산의 정의를 보면 다음과 같다.

[정의 3] 서열 삽입

삽입 연산은 중간에 하나의 아미노산이 삽입되었음을 나타내며 ins(삽입위치, 삽입 값)으로 나타낸다.

[정의 4] 서열 삭제

삭제 연산은 전체 서열 중 하나의 아미노산이 제거되었음을 나타내며 del(삭제 위치, 삭제 값)으로 나타낸다.

[정의 5] 서열 치환

치환 연산은 해당 위치의 아미노산이 서로 다름을 나타내며 rep(위치, 치환 값)으로 나타낸다.

따라서 (그림 3)의 결과와 정의된 연산자를 적용하여, 서열 S와 서열 T의 정렬된 서열의 변환 스크립트는, ε={ ins(2,R), rep(5,I), del(7,T) }로 나타낼 수 있다.

```
입력 : S ← 원본 서열, S' ← 새롭게 입력된 서열
출력 : ε ← Edit Script
     idx_first ← Matched Sequence의 처음 인덱스
     idx_end ← Matched Sequence의 끝 인덱스
     seq_front ← SW matching에 의해 잘려나간 S'의 앞부분 서열
     seq_behind ← SW matching에 의해 잘려나간 S'의 뒷부분 서열

01 : BEGIN
02 :
03 : S'의 식별자 iden(S')을 이용하여 데이터베이스로부터 S 검색 ;
04 :
05 : IF (S가 존재하면)
06 : {
07 :   S와 S'을 Smith-Waterman matching 실행 ;
08 :    SW matching의 결과 (idx_first, idx_end, seq_front, seq_behind) 반환 ;
09 :   S_match ← S의 매칭된 서열 ;
10 :   S'_match ← S'의 매칭된 서열 ;
11 :
12 :    WHILE (S_match의 처음 인덱스부터 끝 인덱스까지 Scanning) // Edit Script 작성
13 :   {
14 :         IF (S'에 삽입된 서열이 있으면) THEN ε에 ins() 연산의 결과 추가 ;
15 :         IF (S'에 삭제된 서열이 있으면) THEN ε에 del() 연산의 결과 추가 ;
16 :         IF (S'에 치환된 서열이 있으면) THEN ε에 rep() 연산의 결과 추가 ;
17 :   }
18 : }
19 :
20 : END
```

(그림 4) Algorithm DiffSeq

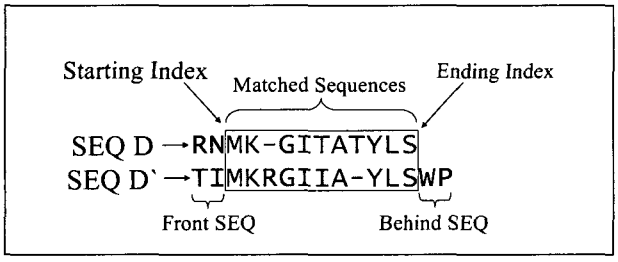
(그림 4)는 데이터베이스의 이전 서열과 PIR-PSD XML 문서에서 추출한 서열을 SW 기법을 이용하여 버전 정보를 생성하는 DiffSeq 알고리즘을 나타내고 있다. 입력 값으로는 정의 1을 만족하는 원본 서열 또는 버전 서열 중 가장 최신의 버전과 새롭게 입력되는 후보 버전 서열이다. 그리고 출력 값으로 두 서열간의 변경 정보를 반환한다. 원본 서열 또는 버전 서열 중 가장 최신의 버전을 S라 하고 새롭게 입력되는 서열을 S'이라 할 때, S' 서열의 식별자를 이용하여 S를 데이터베이스에서 검색한다. 여기서 데이터베이스에 S의 정보가 있는 경우 DiffSeq을 적용한다. 만약, S의 정보가 데이터베이스에 없는 경우 (즉, 데이터베이스에 처음 입력된 서열 엔트리인 경우)는 DiffSeq 알고리즘은 적용하지 않고, 원본 서열로써 데이터베이스에 저장된다. S가 존재하는 경우 두 서열 S와 S'을 입력 값으로 SW 정렬을 실행한다. SW 정렬의 결과로써 두 서열간의 정렬된 서열이 얻어지고, 정렬된 서열로부터 제외될 S'의 하위 서열을 알아낼 수 있다. 다음으로 정렬된 두 서열의 시작 부분부터 끝 부분까지 서로 다른 부분을 검출해 낸다. 이때 각각 삽입, 삭제, 치환의 이벤트가 발생하였을 때, 적절한 연산자를 반환하여 Edit Script에 이 정보를 추가한다. 이렇게 만들어

진 Edit Script와 제외된 S'의 하위 서열은 데이터베이스에 저장되어 버전 서열을 검색하는 정보로 사용된다.

3.3 단백질 버전 서열 검색

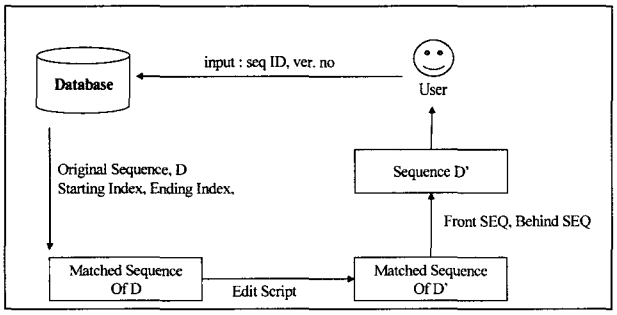
단백질의 버전 서열을 검색하는 것은 원본 서열 테이블에 있는 단백질 서열 데이터를 원하는 버전 서열로 변환하는 것을 의미한다. 3.2절에서 단백질 버전 서열을 생성할 때, 원본 서열에서 버전 서열로 변환하기 위한 정보 추출해 내는 것을 알아보았다. 이 절에서는 데이터베이스에 저장된 변환 정보를 이용해 어떻게 원하는 버전의 서열로 변환하는지 설명한다.

데이터베이스의 원본 서열을 D라 하고 버전 서열(즉, 검색이 되어야 할 서열)을 D'라 한다면, 이 때 검색하기 위해 필요한 정보는 (그림 5)과 같다.



(그림 5) 버전 서열 검색에 필요한 정보

Matched Sequences는 D와 D'을 SW 알고리즘을 적용하여 정렬된 서열을 말한다. Starting Index는 D의 Matched Sequence가 시작되는 인덱스이고, Ending Index는 D의 Matched Sequence가 끝나는 인덱스이다. Front SEQ는 D'의 하위 서열로써 Matched Sequence에 의해 잘려나간 D' 서열의 앞부분이고, Behind SEQ는 역시 D'의 하위 서열로써 Matched Sequence에 의해 잘려나간 D' 서열의 뒷부분이다. 여기서 Edit Script는 D의 Matched Sequence를 D'의 Matched Sequence로 변환하기 위한 정보를 말한다.



(그림 6) 단백질 버전 서열의 검색

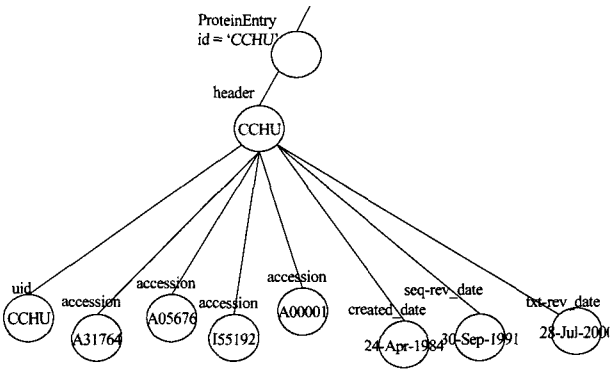
(그림 6)은 단백질 버전 서열을 검색하기 위한 과정을 나타내고 있다. 첫째, 원본 서열 테이블에서 검색을 원하는 서열을 가져온다. 둘째, Starting Index와 Ending Index 값을 이용하여 유사도가 높은 부분을 선택한다. 이때 Staring Index와 Ending Index 값은 SW 알고리즘을 적용하기 전의 (즉, 갭이 포함이 안 됨) Index 값을 사용한다. 셋째, Edit Script를 분석하여 삽입, 삭제, 치환 정보를 적용하여 D의 Matched Sequence를 D'의 Matched Sequence로 변환한다. 넷째, 변환된 서열의 앞과 뒷부분에 Front SEQ와 Behind SEQ를 삽입한다.

이로써 사용자는 원본 서열로부터 원하는 버전의 단백질 서열을 검색할 수 있다. 또한 치환 행렬을 이용하여 두 서열의 유사도를 점수로 산출해 낼 수 있다. 이 논문에서는 기본적으로 BLOSUM50[20] Scoring Matrix을 적용하여 유사도 점수를 산출할 수 있게 하였다.

4. 단백질 주석 데이터의 버전 관리

4.1 단백질 주석 데이터의 갱신

단백질 주석 데이터는 단백질 서열을 제외한 단백질 정보를 말한다. 즉, 생명체 정보, 실험자 정보, 단백질이 어떤 분류에 해당하는가를 알 수 있는 클래스 정보, 유전적 기능 정보, 다른 생물학적 데이터베이스와의 Cross Reference 정보, 학회나 저널에 게재된 정보 등의 여러 가지 정보가 이에 해당된다. 단백질 주석 데이터는 반구조적 데이터의 성격을 가지고 있다. 갱신시, 단백질 서열 데이터와 마찬가지로 PIR-PSD에서 정기적으로 배포하고 있는 XML 파일을 사용한다. 그리고 주석 데이터의 버전 관리는 능동데이터베이스[9]의 트리거[16]를 이용하여 관리한다.



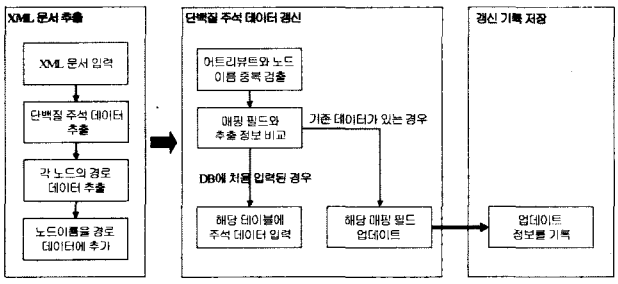
(그림 7) 트리형태의 XML 문서구조

단백질 주석 데이터 갱신을 수행하는데 있어서 몇 가지 가정이 있다. 첫째, 단백질 주석 데이터는 XML 형태로 제공되어야 한다. 둘째, PIR-PSD에서 제공하는 XML DTD의 정의서에 따라 각 어트리뷰트 값 또는 엘리먼트 값과 데이터베이스의 테이블의 필드는 서로 매핑되어야 한다. 셋째, (그림 7)의 트리 형태로 표현된 XML 문서에서 데이터베이스와 매핑되는 노드가 없다면 데이터베이스 매핑 필드의

값이 널 값임을 의미한다.

XML 문서 내에는 노드의 경로는 다르나 엘리먼트의 이름이 같은 노드가 존재한다. 예를 들어 'note'라는 엘리먼트 이름을 가진 노드는 organism, reference, accinfo, genetics, function, feature 등의 노드의 하위에도 동일하게 존재한다. 만약 노드 경로 정보가 없다면 이 노드들의 값은 서로 충돌되어 잘못된 값으로 갱신 될 수 있다. 따라서 구조 정보를 추출하여 (그림 7)처럼 트리형태로 XML 문서의 구조를 표현함으로써 XML 문서 내에서 같은 엘리먼트 이름을 가진 엘리먼트의 값을 정확히 갱신 할 수 있다.

단백질 주석 데이터를 갱신하는 과정은 크게 세 부분으로 나눌 수 있다. 첫째, XML 문서 추출하는 모듈, 둘째 추출된 정보와 데이터베이스 테이블 필드간의 매핑 정보를 비교하여 최신 정보로 갱신하는 모듈, 셋째 갱신한 기록을 저장하여 주석데이터의 이력을 관리하는 모듈로 나눌 수 있다. (그림 8)은 PIR-PSD에서 배포하고 있는 XML 포멧의 단백질 서열 주석 데이터를 이용하여 버전 관리를 수행 과정을 나타내고 있다. 주석 갱신 모듈은 각 추출된 어트리뷰트 이름, 엘리먼트 이름, 어트리뷰트 값, 엘리먼트 값과 경로 정보를 DTD를 토대로 미리 만들어진 매핑 테이블의 정보를 이용하여 비교한다. 매핑 테이블에서 해당 정보를 찾아내면 해당 데이터베이스 테이블의 필드를 새로운 값으로 갱신한다.



(그림 8) 단백질 주석 데이터 버전 관리

갱신 기록 저장 모듈은 데이터베이스의 해당 필드를 갱신 한 기록을 데이터베이스에 저장한다. 이렇게 작성된 갱신 기록을 근거로 하여 과거 시점의 주석 데이터를 추적할 수 있고 이렇게 추적된 결과는 해당 시점의 단백질 서열과 함께 사용자에게 전달되어 진다.

여기서, 주석 갱신 모듈과 갱신 기록 저장 모듈은 능동 데이터베이스[9]의 트리거[16]를 이용하여 프로세스가 진행된다.

4.2 트리거를 이용한 단백질 주석 데이터의 이력 정보 저장

단백질 주석 데이터의 새로운 버전이 입력될 때 트리거[16]를 이용하여 과거의 노드값과 새롭게 입력된 값을 비교한다. 트리거는 능동 데이터베이스 관리 시스템[9]에서 사용되는 개념으로 능동 데이터베이스는 계속해서 데이터베이스의 상태를 모니터하고 미리 정의된 사건이 발생했을 때 스스로 반응한다. 그리고 데이터베이스 사건에 의해 트리거 되는 조건들을 모니터 하다가 만약 그 조건이 만족되면 연관된 행위가 수행된다.

```
CREATE TRIGGER logKEYWORDS ON KEYWORDS
INSTEAD OF INSERT AS  // 트리거 이벤트
BEGIN
    DECLARE @nPIRID varchar(10), @nKYORDER varchar(10),
        @nKEYWORD varchar(50)  // 변수선언
    SELECT @nPIRID=PIRID, @nKYORDER=KYORDER,
        @nKEYWORD=KEYWORD FROM INSERTED  // 변수
        할당

    IF EXISTS (SELECT KEYWORD FROM KEYWORDS
        WHERE PIRID=@nPIRID AND  // 트리거 조건
                KYORDER=@nKYORDER) // 기존
        KEYWORD가 있는 경우
        BEGIN  // 트리거 액션 시작 ( 과거의 키워드 값과 비교 )
        DECLARE @oKEYWORD varchar(50)
        SELECT @oKEYWORD=KEYWORD FROM KEYWORDS
        WHERE PIRID=@nPIRID AND
                KYORDER=@nKYORDER
            IF ( @oKEYWORD <> @nKEYWORD )  // 트리거
                조건 ( 과거의 키워드 값과 다르다면 )
            BEGIN  // 트리거 액션 시작
            DECLARE @nRELEASE VARCHAR(10)  // 갱신
                전의 릴리즈 정보
            SELECT @nRELEASE=RELEASE FROM RELINFO
                ORDER BY RDATE ASC // 릴리즈 정보 할당
            INSERT INTO LOGS VALUES // 로그 정보를 삽입
                        (@nPIRID,@nKYORDER,
                         @nRELEASE,'KEYWORDS',
                         'KEYWORD', @oKEYWORD)
            UPDATE KEYWORDS SET KEYWORD =
                @nKEYWORD  // 키워드 테이블 갱신
                        WHERE PIRID = @nPIRID AND
                         KYORDER = @nKYORDER
            END
        END
    ELSE  // 트리거 조건 (현재 삽입한 PIR 식별자가 키워드
        테이블에 처음 삽입 되는 경우)
        BEGIN // 트리거 액션 (키워드 테이블에 현재 입력된 값을
            삽입)
        INSERT INTO KEYWORDS
            VALUES(@nPIRID,@nKYORDER,@nKEYWORD)
        END
END;
```

(그림 9) KEYWORDS 테이블에서 주석 데이터 관리 SQL 트리거의 예

트리거는 타겟 테이블의 조작 연산(삽입, 삭제, 수정)을 모니터 하여 적절한 행동을 정의하는데 사용된다. 테이블에 미리 정의된 사건(삽입, 삭제, 수정)이 일어나면 데이터베이스는 능동적으로 이를 감지하고 적절한 행동을 취한다. 트리거를 이루는 기본 요소로는 트리거 이벤트, 트리거 조건,

트리거 액션으로 나눌 수 있다. 트리거 이벤트는 INSERT 트리거, DELETE 트리거, UPDATE 트리거가 있다. 단백질 주석을 관리하는 트리거는 INSERT 트리거만이 사용된다. 트리거 조건과 액션은 새롭게 입력된 값이 과거의 값과 동일하다면 갱신은 실행하지 않고, 다를 경우에만 갱신을 수행하는 것이다. 또한 트리거는 갱신을 수행하고 과거의 값을 버전 정보와 함께 저장한다.

(그림 9)는 이 논문에서 사용된 트리거의 예를 보여주고 있다. 해당 테이블 (KEYWORDS 테이블)에 삽입이 발생하였을 때, 트리거는 키워드의 PIR ID와 키워드 순서 정보를 이용해 삽입되기 직전의 값과 비교를 한다. 삽입되는 값이 직전의 값과 다르다면 트리거는 키워드 테이블의 값과 버전 정보를 갱신하고 로그 테이블에 갱신 정보를 입력한다. 또한 삽입되는 값이 데이터베이스에 처음 입력되는 경우에는 트리거가 해당 값을 바로 키워드 테이블에 삽입한다. 만약, 삽입되는 값이 데이터베이스의 현재 값과 같다면 삽입 이벤트는 무시되며 아무 작업도 수행하지 않는다.

트리거를 사용하여 데이터베이스가 능동적으로 새로운 데이터의 버전 가능성 여부를 검출함으로써 응용 프로그램에서 주석 이력 데이터를 관리하는 모듈을 따로 구현하지 않아도 된다. 즉, 응용 프로그램에서의 원본 소스 수정에 따른 버전관리 메커니즘의 오류도 미연에 방지 할 수 있다. 또한 응용 프로그램의 오버헤드를 현저히 줄이고, 응용프로그램과 데이터베이스 서버간의 통신량을 줄여 네트워크의 과부하로 인한 오동작을 방지할 수 있다.

5. 구현 및 실험

5.1 시스템 구성

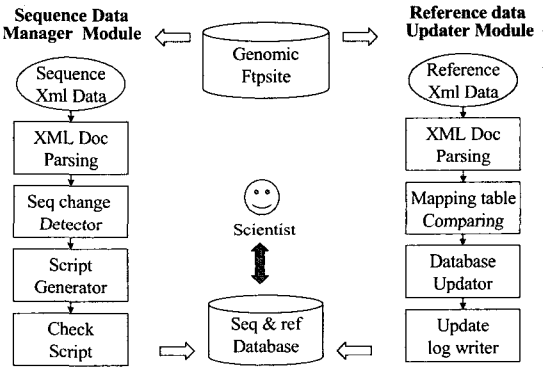
이 논문에서는 단백질 데이터를 단백질 서열 데이터와 주석 데이터를 구분하여 버전관리 메커니즘을 적용한다. (그림 10)은 단백질 서열 데이터와 주석 데이터의 버전관리 시스템의 전체 구조를 표현하고 있다.

PIR-PSD[3]에서 배포하고 있는 파일 타입으로는 XML, FASTA, NBRF, CODATA, MYSQL 등 여러 가지 포맷으로 배포하고 있어, 유저가 사용하는 응용 프로그램에 맞게 선택할 수 있다. PIR-PSD FTP에서 XML 파일을 다운로드한 후 시스템에서 XML 파일을 읽어 들일 때, 서열 데이터와 주석 데이터는 서로 다른 프로세스를 통하여 버전관리 및 저장 메커니즘이 적용된다.

단백질 서열 데이터는 Sequence Change Detecting Module에서 처리가 되며 이 모듈의 하위 컴퍼넌트들은 다음과 같다. XML DOC Parsing은 XML 문서로부터 PIR ID와 해당 서열 추출한다. Seq Change Detector은 추출한 서열은 서열의 ID를 이용하여 데이터베이스의 원본 서열 검색하고 DiffSeq 알고리즘을 통하여 변환 정보를 수집한다. 이때 두 서열 사이에 Smith-Waterman 알고리즘이 적용된다. Script Generator은 Smith-Waterman 알고리즘을 적용하여 수집된 변환 정보를 토대로 변환 스크립트를 작성한다. Check Script은 변환 스크립트의 무결성의 체크하고 버전 정보와 변환 정보를 데이터베이스에 저장한다.

단백질 서열 주석 데이터는 Reference Data Update Module을 통하여 주석 데이터의 버전을 저장하게 되며 하위 컴포넌트들은 다음과 같다. XML DOC Parsing은 XML 문서로부터 PIR ID와 주석 데이터를 추출한다. Mapping Table Comparing은 XML 데이터의 트리 노드와 데이터베이스 필드간의 매핑 정보를 이용하여 과거의 값과 새로운 노드 값이 변화가 있는지를 비교한다. Database Updater는 변화된 노드 값으로 DB의 필드를 갱신한다. 그리고 Update log Writer는 갱신한 내용을 로그 테이블에 저장한다.

여기서 주석 데이터 버전 관리시 트리거는 XML로부터 새롭게 입력되는 노드 값을 입력받아 해당 PIR ID의 매핑 되는 필드를 데이터베이스로부터 검색하여 갱신을 실행하고, 갱신한 내용을 로그 테이블에 저장하는 역할을 한다.



(그림 10) 전체 시스템 구성도

시스템의 구현환경은 데이터베이스 서버의 경우 CPU 1GHz의 인텔 펜티엄3과 384MB 램을 사용하였다. 플랫폼은 Windows 2000 Server를 사용하였고 DBMS로는 MS SQL 2000을 사용하였다. 프로그래밍 구현환경은 JDK 1.4.2와 JAVA SWING을 사용하였다.

5.2 데이터베이스 스키마

이 논문에서 사용되는 데이터베이스의 테이블은 Main 테이블을 중심으로 모델링 되어 있다. KEYWORDS, ORGANISMS, REFERENCE, FEATURES, SEQUENCES, LOGS 테이블의 속성 PIR ID는 MAIN 테이블의 PIR ID를 외래키로 참조하고 있다. 또한 REFERENCE 테이블의

REFID는 REF_JNLS 테이블의 REFID를 참조하고 있으며 VERSEQS의 PIR ID는 SEQUENCES 테이블의 PIR ID를 참조하고 있다.

MAIN 테이블은 PIR-PSD 식별자, 식별자 (Protein entry)가 생성된 날짜, 서열이 변경된 날짜, 주석데이터가 변경된 날짜와 코멘트로 이루어져 있다. KEYWORDS 테이블은 데이터베이스를 키워드를 이용하여 검색할 때 사용되는 키워드를 저장하는 테이블로 PIR-PSD 식별자, 키워드의 순서 정보, 키워드를 포함하고 있다. ORGANISMS은 생물체의 정보를 담고 있는 테이블로써 PIR-PSD 식별자, 단백질의 이름, 소스 정보, 널리 알려진 이름, 학명 정보를 가지고 있다. REFERENCE 테이블은 레퍼런스 정보 식별자인 refid를 가지고 있는 테이블이고 REF_JNS 테이블을 참조하고 있다. REF_JNLS 테이블은 해당 refid마다 가지고 있는 citation 정보, 볼륨 정보, 출판된 일시, 페이지 정보, 제목, 저자, PDB[25]와 MedLine[23] 데이터베이스와의 cross reference 정보를 가지고 있다. FEATURES 테이블은 단백질 내부의 여러 사이트와 region에 대한 명시적 정보를 담고 있으며, PIR-PSD 식별자, feature 정보의 순서, feature label, feature의 링크와 종류, 간략한 설명, 해당 residue의 번호, feature의 상태 정보를 담고 있다. LOG 테이블은 트리거를 사용하여 갱신한 정보를 가지고 있는 테이블로써 PIR-PSD 식별자, 갱신한 테이블에서의 식별자 순서 정보, 버전(릴리즈) 정보, 테이블명, 필드명, 변경 이전 값을 포함하고 있다.

〈표 1〉 데이터베이스 스키마

MAIN	(Pirid, Createdt, Seqrevdt, Txtrevdt, Comment)
KEYWORDS	(Pirid, Kyorder, Keyword)
ORGANISMS	(Pirid, Pname, Source, Common, Formal)
REFERENCE	(Pirid, Rforder, Refid)
REF_JNLS	(Refid, Citation, Vol, Mon, Year, Pages, Title, Authors, Muid, Pid)
FEATURES	(Pirid, Ftorder, Ftlabel, Ftlink, Fttype, Ftdesc, Ftspec, Ftstatus, Ftnote)
LOG	(Pirid, Torder, Release, Tname, Tfield, Value)
SEQUENCES	(Pirid, Release, Seqrevdt, Seq)
VERSEQS	(Pirid, Release, Seqrevdt, Matlen, Sidx, Eidx, Script, Seqf, SeqB, Similarity)
RELINFO	(Release, Rdate, NumSeq, NumRes)

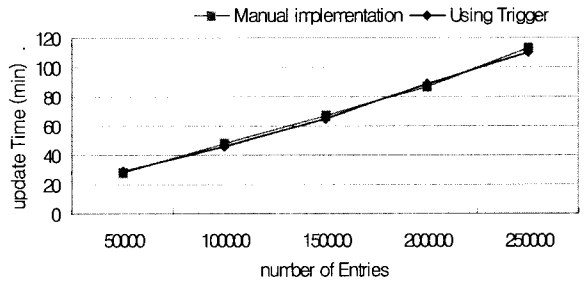
SEQUENCES 테이블은 단백질의 원본 서열을 가지고 있는 테이블로써 전체 시퀀스를 모두 저장한다. 제일 처음 버전의 XML 문서가 가지고 있는 서열 정보와 데이터베이스에 처음 입력되는 PIR-PSD 식별자의 서열은 모두 SEQUENCES 테이블에 저장되며, 기존에 존재하는 PIR-PSD 식별자의 서열이 데이터베이스에 삽입되는 경우 VERSEQS 테이블에 정보를 저장하게 된다. VERSEQS 테이블은 서열을 저장하는 것이 아니라, SEQUENCES 테이블에 있는 원본 서열을 원하는 시점의 서열로 변환하기 위해 필요한 정보를 담고 있는 테이블이다. 마지막으로 RELINFO 테이블은 과거 갱신을 수행한 적이 있는 XML 문서의 버전(릴리즈)정보를 담고 있는 테이블로써 XML 문서의 갱신 작업이 끝나면 RELINFO 테이블에 해당 릴리즈 정보를 삽입하게 된다. RELINFO 테이블을 이루고 있는 필드는 release 넘버, 릴리즈 한 날짜, 해당 릴리즈의 서열 갯수, 해당 릴리즈의 레지듀의 갯수로 이루어져 있다.

5.3 성능 평가

이 절에서는 우리가 제안한 갱신 시스템의 성능을 분석하기 위해 갱신 실행 시간과 제안된 갱신 시스템을 적용하여 얼마의 디스크 용량 감소가 있는지 측정하였다.

첫 번째 실험은 PIR-PSD 데이터베이스에서 릴리즈 된 '75.01' 버전의 갱신 실행 시간을 측정한 것이다. 트리거를 사용하여 버전관리를 수행하였을 때와 트리거를 사용하지 않고 응용 프로그램에서 직접 버전관리 메커니즘을 구현하였을 때의 갱신 시간을 측정하였다. '75.01' 버전의 총 엔트리 수는 283,269개이며, 단백질의 서열 데이터와 주석 데이터 모두를 포함한 갱신 시간이다. 그리고 매 50,000 엔트리마다 검색 시간을 측정하였다.

(그림 11)에 보는 바와 같이 단백질의 엔트리 숫자가 늘어남에 따라서 트리거를 사용하여 버전관리를 하였을 때와 응용 프로그램에서 직접 구현하였을 때 모두 갱신에 걸린 시간 또한 증가함을 알 수 있다. 그리고 두 경우 모두 증가하는 엔트리 숫자에 대하여 비슷한 갱신 시간을 보였다.

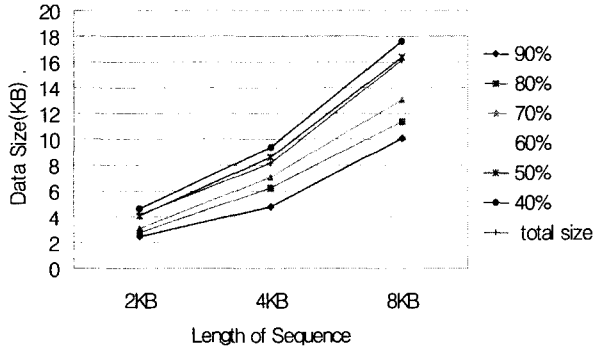


(그림 11) 엔트리별 갱신 수행 시간

갱신 시간은 비슷하지만, 트리거를 사용하였을 경우 데이터베이스가 능동적으로 새로운 데이터의 버전 가능성

여부를 검출함으로써 데이터베이스의 독립성을 높여 응용 프로그램에서의 원본 소스 수정에 따른 버전관리 메커니즘의 오류를 미연에 방지 할 수 있다. 또한 응용 프로그램과 데이터베이스 서버가 분리된 분산 환경의 경우 응용프로그램과 데이터베이스 서버간의 송·수신량을 줄일 수 있다.

두 번째 실험은 원본 서열과 버전 서열사이의 유사도에 따른 데이터 크기의 변화를 측정한 것이다. 실험에 측정된 서열의 크기는 평균 2KB, 4KB, 8KB에 해당하는 서열들의 유사도를 40%, 50%, 60%, 70%, 80%, 90%로 나누어 데이터의 크기를 측정하였다.



(그림 12) 유사도에 따른 데이터 크기

(그림 12)에서 total size는 비교 대상인 두 서열의 어떤 연산도 적용하지 않은 총 데이터 크기이다. 즉, NCBI의 경우로써 하나의 원본 샘플로부터 생성된 버전 서열을 아무런 연산 없이 저장할 때의 크기이다. 2KB의 서열 길이의 경우 50%, 4KB의 서열 길이의 경우 55%, 8KB의 서열 길이의 경우는 52% 이상의 서열 유사도를 가지고 있을 때, 전체 서열의 크기보다 작은 감소량을 보였다. 이 실험에서 50% 보다 낮은 유사도를 가지고 있을 때 총 서열 크기보다 데이터의 크기가 오히려 더 큰 수치를 나타내었고, 55% 이상의 유사도를 가지고 있을 때 제안된 기법을 사용하여 이력 데이터를 관리하였을 때 물리적 공간 감소량을 보였다.

공간 사용량을 측정한 실험에서 감소량을 보이기 시작한 유사도가 조금 다른 것은 버전 서열 정보를 추출할 때, Smith-Waterman 정렬의 결과에 따라 데이터베이스에 버전 정보로 저장되는 서열의 앞부분과 뒷부분에 대한 길이가 다르기 때문이다. 그리고 50% 이하의 유사도를 가진 경우 더 큰 공간사용량을 보인 이유는 하나의 아미노산 서열이 변경될 때, 이를 표현하기 위해 Edit Script는 3Byte의 용량이 차지하기 때문에 이 경우는 Edit Script의 길이가 버전 서열의 길이보다 오히려 길기 때문이다.

6. 결 론

기능이 밝혀진 단백질의 아미노산 서열을 유전 공학 기법을 이용하여 변화시키거나 새로운 잔기 서열을 만들어 유용한 단백질을 만드는 과정에서 하나의 원본 단백질 서열에 대하여 다른 서열 구성을 가지고 있는 여러 가지 단백질 서열이 생겨 날 수 있다. 이 논문에서는 이러한 버전 서열과 버전 서열의 주석정보를 저장할 때 중복성을 제거하고 유사도 측정 및 버전 관리 자동화에 대한 연구를 다루었다.

이 논문에서는 단백질 데이터를 두 가지로 분류하여 서로 다른 방법을 적용하였다. 첫째, 단백질 서열 데이터의 경우는 Swith-Waterman 알고리즘을 적용하여 변화된 정보만을 데이터베이스에 저장한다. 이로써, 단백질 샘플의 과거 버전 서열을 검색해 내는 것은 물론, 버전 관리 시 자동으로 유사도 측정이 가능하였다. 두 서열에서 55% 이상의 유사도를 가지고 있을 때, edit script를 사용하여 버전 서열의 저장 공간을 감소시킨 결과를 보였다. 둘째, 반 구조 데이터의 특성을 가지고 있는 단백질 주석 데이터를 트리 구조로 표현하고 트리거를 이용하여 변경 사항을 검출하여 데이터가 변경된 부분만 갱신 로그 테이블에 저장한다. 그리하여 단백질 주석 데이터의 변경 정보 추적과 버전 정보 저장이 자동으로 가능하게 하였다. 또한 트리거를 이용함으로써 단백질 주석 데이터의 갱신과 갱신로그 작성을 하기 위한 빈번한 응용 프로그램과 데이터베이스의 송수신 량을 줄여 분산 환경의 경우 네트워크의 과부하로 인한 시스템 오류를 미연에 방지하도록 하였다.

우리의 연구 결과를 통하여 단백질 데이터의 이력을 분석하여 특정한 환경에 노출된 유전체 샘플이 돌연변이를 일으키는 서열의 변화 과정을 데이터베이스에 저장함으로써 단백질 서열 돌연변이 분석에 이용 할 수 있다. 또한 정상인의 염기 서열과 암을 앓는 환자의 염기서열과 비교하면 유전자의 어떤 변화로 암이 발생하는 지를 규명할 수 있게 된다. 그 뿐만 아니라 방어항원의 돌연변이 서열 분석[2] 및 구조를 변경시켜 신약 및 백신 개발에도 유용하게 응용 될 수 있다.

참 고 문 헌

[1] 정광수, 이영화, 박성희, 류근호, "능동 데이터베이스 기반의 버전 서열 검출 메카니즘", 제7회 한국 과학기술 정보 인프라 워크샵 바이오인포메틱스 학술발표논문집, pp.249-259, 2002.

[2] 박영미아, 유천권, 성원근, 이화중, 오희복, "유전자 재조합 탄저 성분백신 개발 연구(III) : 면역횟수에 따른 항체가 변동 및 방어항원 유전자 변이 분석", 국립보건원보, 제38권

The Report of National Institute of Health 38, pp.92-107, 2001.

[3] Cathy, H., Wu, Lai-Su L. Yeh, Hongzhan Huang, Leslie Arminski, Jorge Castro-Alvear, Yongxing Chen, Zhang-Zhi Hu, Robert S. Ledley, Panagiotis Kourtesis Baris E. Suzek, C. R. Vinayaka, Jian Zhang, and Winona C. "The Protein Information Resource," Barker Nucleic Acids Research, Vol.31, pp.345-347, 2003.

[4] David L. Wheeler, Deanna M. Church, Alex E. Lash, Detlef D. Leipe, Thomas L. Madden, Joan U. Pontius, Gregory D. Schuler, Lynn M. Schriml, Tatiana A. Tatusova, Lukas Wagner, and Barbara A. "Rapp Database resources of the National Center for Biotechnology Information," 2002 update Nucl. Acids. Res, Vol.30, pp.13-16, 2002.

[5] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, Barbara A. Rapp, and David L. Wheeler "GenBank" Nucl. Acids. Res, Vol.30, pp.17-20, 2002.

[6] Kwang Su Jung, Sung-Hee Park, Keun Ho Ryu, Hyeon S. Son, "Sequence Version Management System based on Trigger," Korean Society for Bioinformatics Annual Meeting, Vol.1, pp.134-141, 2002.

[7] G. Stoesser, W. Baker, A. V.D Broek, E. Camon, M. Garcia-Pastor, C. Kanz, T. Kulikova, V. Lombard, R. Lopez, H. Parkinson, N. Redaschi, P. Sterk, P. Stoehr, M. Ann T., "The EMBL nucleotide sequence database," Nucl. Acids. Res, Vol.29, pp.17-21, 2001.

[8] Nabil R. Adam, Igg Adiwijaya, Terence Critchlow, Ron Musick, "Detecting Data and Schema Changes in Scientific Documents," ADL, pp.160-172, 2000.

[9] Norman W. Paton: "Active Rules in Database Systems" (Contents), Springer, New York, ISBN 0-387-98529-8, 1999.

[10] T. A. Tatusova, I. Karsch-Mizrachi, J. A. Ostell: "Complete genomes in WWW Entrez: data representation and analysis." Bioinformatics, Vol.15, No.7, pp.536-543, 1999.

[11] Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., Lipman, D. J. "Gapped BLAST and PSI-BLAST : a new generation of protein database search programs," Nucleic Acids Res, Vol.25, pp.3389-3402, 1997.

[12] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, Jennifer Widom, "Change Detection in Hierarchically Structured Information," SIGMOD Conference, pp.493-504, 1996.

[13] I-Min A. Chen, Victor M. Markowitz, Stanley Letovsky, Peter Li, Kenneth H. Fasman, "Version Management for Scientific Databases," EDB, pp.289-303, 1996.

[14] Douglis, F., Ball, T., "Tracking and Viewing Changes on the Web," In 1996 USENIX Technical Conference, 1996.

[15] Douglis, F., Ball, T., Chen, Y., "WebGUIDE: Querying and Navigating Changesin Web Repositories," In Fifth international World Wide Web Conference, 1996.

[16] Jennifer Widom, Stefano Ceri, "Introduction to Active Database Systems. Active Database Systems: Triggers and Rules For Advanced Database Processing", pp.1-41, 1996.

[17] Zhang, K., Wang, J., Sasha, D., "On the editing distance between undirected acyclic graphs," In International Journal of Foundations of Computer Science, 1995.

[18] Michael S. Paterson and Vlado Dancik, "Longest Common Subsequences," Mathematical Foundations of Computer Science, pp.127-142, 1994.

[19] Wang, J., Zhang, K., Jeong, K., Shasha. D., "A System for Approximate Tree Matching," In IEEE Transaction On Knowledge and Data Engineering, Vol.6, pp.559-570, 1994.

[20] Henikoff, S., and Henikoff, J. G., Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. USA 89, pp.10915-10919, 1992.

[21] Altschul, S. F. et al, "Basic local alignment tool," J. Mol, Biol., Vol.215, pp.403-10, 1990.

[22] Sasha, D., Zhang, K., "Fast algorithms for unit cost editing distance between trees," In Journal of Algorithms, Vol.11, 1990.

[23] Masys, D. R., New directions in bioinformatics, J. Res. Natl. Inst. Stand. Tech. 94, pp.69-63, 1989.

[24] Pearson, W. R., Lipman, D. J., "Improved tool for Biological sequence comparison," Proc Natl Acad Sci USA, Vol.85, pp.2444-2448, 1988.

[25] Abola, E. E., Bernstein, F. C., Bryant, S. H., Koetzle, T. F., and Weng, J., Protein data bank, In: Crystallographic databases - information content, software systems, scientific applications, Allen, F.H., Bergerhoff, G., and Sievers, R., eds., Data Commission of the International Union of Crystallography, Cambridge, 1987.

[26] Smith, T. F. and Waterman, M. S., Identification of common molecular sequences, J. Mol, Biol., pp.195-197, 1981.

[27] Hirschberg, D. S., "Algorithms for the longest common subsequence problem," In Journal of the ACM, pp.664-675, 1997.

[28] Wagner, R., "On the complexity of the extended string-to-string correction problem," In seventh ACM symposium on the Theory of Computation, 1975.

[29] Sellers, P. H., "An algorithm for the distance between two finite sequences," J. Comb. Th. A, Vol.16, pp.253-258, 1974.

[30] Needleman, S. B., Wunsch, C. D., "A general method applicable to the search for similarities in the amino acid sequence of two proteins," J. Mol. Biol, Vol.48, pp.443-453, 1970.

[31] Bellman, R., "Dynamic Programming," Princeton University Press, 1957.

정 광 수

e-mail : ksjung@dblab.chungbuk.ac.kr
2001년 충북대학교 화학공학부(공학사)
2004년 충북대학교 정보산업공학과 석사(공학석사)
2004년~현재 충북대학교 대학원 전자계산학과 박사과정
관심분야 : Bioinformatics, 단백질 서열 및 구조, 생명정보 데이터베이스

박 성 희

e-mail : shpark@dblab.chungbuk.ac.kr
1996년 충북대학교 도시공학과(공학사)
1998년 한국전자통신연구원 컴퓨터소프트웨어연구소위촉연구원
2001년 충북대학교 대학원 전자계산학과 석사(이학석사)
2003년~현재 충북대학교 대학원 전자계산학과 박사수료
관심분야 : Bioinformatics, 생명정보 데이터 통합, 단백질 구조 예측, XML 데이터베이스

류 근 호

e-mail : khryu@dblab.chungbuk.ac.kr
1976년 숭실대학교 전산학과(이학사)
1980년 연세대학교 공업대학원 전산전공(공학석사)
1988년 연세대학교 대학원 전산전공(공학박사)
1976년~1986년 육군 군수지원사 전산실(ROTC 장교), 한국전자통신연구소(연구원), 한국방송대학교 전산학과(조교수) 근무
1989년~1991년 University of Arizona, Research Staff (TempIS 연구원, Temporal DB)
1986년~현재 충북대학교 전기전자컴퓨터공학부 교수
관심분야 : 시간 데이터베이스, 시공간 데이터베이스, Temporal GIS, 객체 및 지식 데이터베이스 시스템, 지식기반 정보검색시스템, 데이터마이닝, 데이터베이스 보안 및 바이오인포메틱스 등