'샲' 등 한글 표현 문제 해결을 위한

공공 정보시스템
한글 처리 가이드라인

요약문

1. 제목

'샲' 등 한글 표현 문제 해결을 위한 "공공 정보시스템 한글 처리 가이드라인"

2. 가이드라인 제공 배경

최근 더 '샲' 아파트(주소), 모모이 '쬬'(귀화자 성명), '펦' 시콜라(상품명) 등 외국어 고유명사 표기를 위해 기존에 잘 쓰이지 않던 글자의 사용이 증가하고 있으나 현재 많은 공공 정보시스템은 이러한 글자를 정상적으로 처리하지 못해 민원서류 발급 및 접수시 해당 글자가 '?'로 표시되는 등의 문제가 발생하고 있다. 거의 모든 대민행정이 전자적으로 처리되고 있는 현 상황에서 현재 국민이 겪고 있는 불편을 조속히 해소하고 향후 이러한 문제가 발생하지 않도록 하기 위해서는 이 문제에 대한 종합적인 분석 및 근본적인 대책 마련이 필요하다.

3. 한글처리 현황 및 처리 방안

현재 공공 정보시스템 대부분은 '샲 등 일부 한글(이하 확장한글)을 처리하지 못하는 'EUC-KR' 방식을 채택하고 있다. 그러나 EUC-KR 방식은 전체 한글 11,172자 중 2,350자만 표현 가능한 문제가 있어 민간 정보시스템에서는 한글 뿐 아니라 다양한 국가의 문자를 동시에 표현 가능한 UTF-8 방식이 세계적으로 확산되는 추세이다. UTF-8 방식은 Google, Facebook, Twitter등 세계 주요 웹사이트 1만개 중 51%에서 사용되고 있으며, 여러가지 웹 관련 최신 기술은 UTF-8방식을 기본적으로 지원하고 있다. 그러므로 확장한글에 대한 처리 뿐 아니라 서비스의 고도화 및 글로벌화를 고려할

때 향후 구축되는 정보시스템은 UTF-8 방식을 우선 사용하는 것이 적합하다.
단, 기존에 운영중인 시스템은 UTF-8 전환시 응용프로그램 수정 뿐 아니라 축적된 모든 DB를 변환하여야 하므로 많은 시간과 비용이 소요될 수 있는데, 이러한 경우는 글자 순 정렬이나 색인처리 등 일부 기능적 제약은 있으나 기존 DB의 일괄 변환 없이 확장한글을 처리할 수 있는 NCR 등 변형 EUC-KR 방식을 사용하는 것이 효과적일 수 있다.

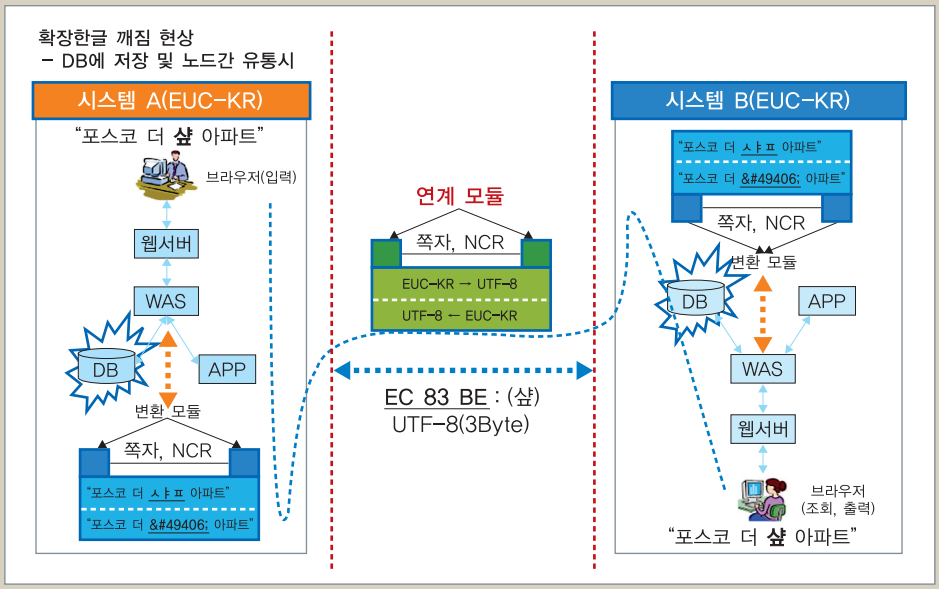
4. 가이드라인 주요 내용

공공기관 정보시스템에서 확장한글 문제를 해결하기 위한 본 가이드라인의 주요 내용은 첫 번째, 한글처리 관련한 문제 개요 설명 및 진단방법을 안내하고, 두 번째, 소스코드 등을 포함하여 상세한 기술적 해결방안 및 방안별 실제 구현방법을 제시하고, 마지막 세 번째, 방안별 소요비용 모의 산정식을 제공함으로써, 해당기관의 정보시스템에서 한글문자를 원활하게 처리하고 운영토록 지원한다.

〈공공 정보시스템 한글처리 문제 발생 사례 (인터넷 민원 신청 페이지)〉

아파트 이름에 포함된 '샾' 글자가 제대로 표시되지 못하고 '?'로 나타난 것을 볼 수 있다.

〈정보시스템 한글 처리(쪽자, NCR) 개념도〉



단일시스템내 변환모듈과 정보시스템 간의 연계모듈을 통해 확장한글을 처리할 수 있는 개념도를 볼 수 있다.

▣ '샲' 등 EUC-KR에 없는 한글의 처리 방법

① **쪽자쓰기** : KS X 1001 표준에 따라 한글을 채움기호와 초·중·종성으로 풀어쓰는 방식

② **NCR** : W3C 웹표준에 따라 확장한글을 숫자로 변환하여 표시하는 방식

③ **MS949** : EUC-KR을 기반으로 Microsoft사가 자체 개발한 인코딩 기술

④ **UTF-8** : 확장한글을 기본 포함하는 ISO 10646 표준(유니코드) 사용

※ (용어) 기본한글 : EUC-KR에 포함된 한글, 확장한글 : EUC-KR에 포함되지 않은 한글

〈확장한글 처리 기술 비교〉

구분	쪽자쓰기	NCR (Numeric Character Reference)	MS949	UTF-8
확장한글 처리 방법	초·중·종성으로 풀어쓰기 예) 샮 → (채움)ㅅㅑㅍ	숫자로 변환 예) 샮 → 샾	EUC-KR에 확장한글 별도 추가	확장한글을 기본적으로 포함
관련 표준	KS X 1001	W3C 웹표준 (SGML)	비표준	ISO 10646 (유니코드)
한글 1자당 필요용량	기본한글 2 Byte 확장한글 8 Byte	기본한글 2 Byte 확장한글 8 Byte	기본한글 2 Byte 확장한글 2 Byte	기본한글 3 Byte 확장한글 3 Byte
장점	– 용량 증가 미미 – 별도 처리 못하는 경우 풀어쓰기로 인식	– 용량 증가 미미 – 다양한 문자 표기 가능 – 출력시 자동변환(웹)	– DB 용량 증가 없음	– 국제표준에 따라 다국어 동시표현 가능
단점	– 출력시 변환모듈 필요 – 한글만 표기 가능 – 가나다순 정렬 복잡	– 가나다순 정렬 복잡	– 가나다순 정렬 복잡 – 웹 기반 환경에서 호환문제 유발 우려	– 기존 데이터 변환 필요 – 변환시 용량 증가

〈인코딩 방식별 확장한글 처리 사례 : '더샮APT102동'〉

EUC-KR	더		?		A	P	T	1	0	2	동	
코드 값	B4	F5	??	??	41	50	54	31	30	32	B5	BF

※ 처리불가로 오류발생

쪽자쓰기	더		□(채움)		ㅅ		ㅑ		ㅍ		A	P	T	1	0	2	동	
코드 값	B4	F5	A4	D4	A4	B5	A4	C1	A4	BD	41	50	54	31	30	32	B5	BF

NCR	더		&	#	4	9	4	0	6	;	A	P	T	1	0	2	동	
코드 값	B4	F5	26	23	34	39	34	30	36	3B	41	50	54	31	30	32	B5	BF

MS949	더		샮		A	P	T	1	0	2	동	
코드 값	B4	F5	98	DE	41	50	54	31	30	32	B5	BF

UTF-8	더			샮			A	P	T	1	0	2	동		
코드 값	EB	8D	94	EC	83	BE	41	50	54	31	30	32	EB	8F	99

▣ 기존 시스템 조치 소요비용

● UTF-8 전환

• 총 비용 = ① AP수정비용 + ② DB변환비용 + ③ 스토리지 증설

① AP(응용프로그램) 수정 비용

– 소요비용 = 작업량(Man-Day) × 중급기술자 1일 노임단가

– 작업량(Man-Day) = (DB의 한글 필드 수 × 1.5 + 13)

※ 작업량 = 문제이해(2MD) + 분석설계(10MD) + 개발 · 검증(1.5MD x 필드 수) + 반영(1MD)

※ 중급기술자 1일 노임단가 = 188,139원 ('10년 소프트웨어 사업 대가기준)

※ 단, 소요 기능점수를 산정 가능한 경우는 기능점수 기준으로 산정된 비용을 우선한다.

② 기존 EUC-KR로 된 DB를 UTF-8로 변환하는 비용

– 소요비용(원) = (10 + 2 × DB용량(TB)) × 중급기술자 1일 노임단가

③ DB 변환작업을 위해 필요한 스토리지 증설 비용

– 소요비용 = (기존용량(TB) × 2 + 실 데이터 증가 용량(TB)) × 20백만원

– 실 데이터 증가용량(TB) = 한글이 포함된 문자 데이터 용량 × 0.4

※ 백업용 사본 및 변환된 결과 DB를 저장하기 위해 기존 DB 용량의 2배 이상의 공간이 추가로 필요하다. 단, 작업 종료 후에는 불필요하므로 임차도 가능하다.

※ 실 데이터 증가용량은 첨부파일 등을 제외한 순수 문자 데이터 용량에 따라 변동된다.

※ 20백만원 = 스토리지 1TB 당 단가

〈산정사례〉

• 기존 데이터 중 문자데이터 300GB, 한글 필드 50개인 시스템 전환시
① AP 수정 : 16백만원 (약 4MM)
② DB 변환 : 1.5백만원
③ 스토리지 증설 : 15백만원 (600+120 = 720GB 증설)
⇒ 계 : 32.5백만원

● EUC-KR + NCR 적용

• NCR 적용 시에는 응용프로그램 수정 비용만 소요되며 산정 방법은 UTF-8 전환과 동일하다.

Contents

• 첨부 자료

Ⅰ. 개요

1. 목적
2. 용어 정의

본 단원에서는 공공 정보시스템에서 한글을 처리 시 참조되는 가이드라인의 제공 목적을 알리고, 한글 문자 처리와 관련된 문자셋 및 인코딩 등의 용어들을 설명함으로써 한글 처리 개념에 대한 이해를 돕는다.

1. 목적

가. 개요

초성, 중성, 종성의 조합으로 표현 가능한 모든 현대 한글 11,172자 중 많이 사용되는 2,350자만 표현 가능한 EUC-KR 인코딩 방식이 현재 국내의 다양한 정보시스템에서 널리 사용되고 있다.

그러나 사회적으로 글로벌화가 빠르게 진행되면서 외국어 상표, 귀화자 성명 등에 EUC-KR 인코딩 방식으로는 표현할 수 없는 글자들이 쓰이기 시작하였으며, 주요 국가 정보시스템에서 이 글자들을 정상적으로 처리하지 못하는 문제가 발생하였다.

예를 들어 아파트 이름 '더 샲' 이나 외국인 이름 '웬뚜엗한' 등에서 '샲', '엗' 등의 글자는 EUC-KR 방식으로는 처리할 수 없는 글자로 민원서류 발급 시 '?' 로 표시된다.

주민센터 등 현장에서는 이러한 글자들에 대하여 각 건별로 수작업으로 수정하거나 다른 글자로 대체하는 등의 방법을 사용하고 있으나 전자민원 처리, 시스템 간 정보연계 등 자동화된 업무에서도 정상적으로 처리하기 위해서는 시스템 차원의 대책이 필요하다.

근본적인 해결을 위해서는 정보시스템이 11,172자의 현대 한글을 모두 표현할 수 있도록 구현되어야 하며, 각 정보시스템들의 다양한 기술적, 업무적 특성을 고려한 구현 방안이 필요하다.

나. 목적

1) 현재 구축, 운영되고 있는 정보시스템들의 한글 입출력 문제 현황을 조사한다.
2) 정보시스템이 현대 한글 모두를 지원하도록 하는 방안을 마련한다.
3) 기존 정보시스템에서 한글 데이터를 손상 없이 저장하고 연계할 수 있는 개선 방안을 마련하고, 적용 가능한 가이드라인을 제시한다.
4) 향후 정보시스템 구축 시 참고할 수 있도록 개선된 지침을 제안한다.

2. 용어 정의

가. 문자셋 관련 용어

1) 문자셋 (Character Set)

정보를 표현하기 위한 글자들의 집합으로, 문자 집합이라고도 한다. 해당 문자체계 내의 각각의 글자들에게 나름대로의 규칙에 따라 번호를 부여한 목록이라고 볼 수 있다. 경우에 따라 문자 인코딩과 같은 뜻으로 쓰이기도 한다. 유니코드, KS X 1001 등이 문자셋에 해당한다.

2) 유니코드 (Unicode)

세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다루기 위해, 각각의 문자에 고유의 번호(코드 포인트, code point)를 부여하고 이를 운영체제나 플랫폼, 프로그래밍 언어 등의 환경에 관계없이 사용하도록 설계된 산업 표준이다.

3) UCS (Universal Multiple-Octet Coded Character Set)

유니코드 문자 표준을 따르는 ISO/IEC 10646 규격의 문자 집합이다. 일반적으로 유니코드와 같은 의미로도 쓰인다.

4) KS X 1001

정식 명칭은 '정보 교환용 부호계 (한글 및 한자)' 로서, 한국 산업 규격으로 지정된 한국어 문자 집합이다. 이전에는 KS C 5601이라고 불렸으나, KS 규격 번호 체계가 바뀜에 따라 KS X 1001로 변경되었다.

특수 문자와 로마자, 한글, 한자 등을 포함하고 있고, 현대 한글 11,172자 중에서 2,350자를 완성자로 표현하고 있다. 그리고 2,350자를 제외한 나머지 글자는 정보교환 시 채움쪽자 방식으로 교환하도록 규정하고 있다.
한자의 경우 4,888자가 포함되어 있다.

5) MS949 (Windows Codepage 949)

KS X 1001(KS C 5601)에서 지정한 영역 외에 확장한글(2,350자 외 현대 한글)을 배치한 문자셋이다. Microsoft에서 KS X 1001을 자체적으로 확장한 것으로 확장완성형 또는 통합완성형이라고 불리기도 한다.
표기 방식으로 CP949, Win949라고도 표현하며, Microsoft의 Outlook Express, Internet Explorer, Frontpage 등에서는 MS949를 ks_c_5601-1987로 표기하기도 한다.

나. 인코딩 관련 용어

1) 인코딩 (encoding)

문자셋을 컴퓨터에서 저장하거나 통신에 쓰기 위해 부호화(기호화)하는 방법을 가리킨다. 문자셋에서 정의된 글자들의 번호를, 컴퓨터에서 사용하기 적합하게 일정 범위 안의 코드값으로 변환하는 방법을 말한다. 문자셋과 인코딩은 흔히 혼동되기 쉬우나 엄밀히 구분하면 다른 개념으로, 예를 들면 유니코드는 문자셋이고, UTF-8은 유니코드의 여러 인코딩 방법 중 하나이다.

2) EUC-KR (Extended Unix Code - Korean)

KS X 1001 규격을 따르는 한글 인코딩이다. 현대 한글 11,172자 중에서 완성자 한글 2,350자만 표현한다.

3) EUC-KR KS

한글 2,350자만 표시해주는 EUC-KR 인코딩에 비해, 모든 현대 한글을 표현할 수 있도록 한 인코딩 방식이다. 2,350자 외 한글들을 KS채움쪽자 방식으로 인코딩하는 방법으로, KS X 1001 규격에 부합하는 방식을 이른다.
즉, 'EUC-KR' 에 'KS채움쪽자' 를 추가하여 확장한글도 표현하는 인코딩 방식을 'EUC-KR KS' 라고 지칭한다. (EUC-KR KS = EUC-KR + KS채움쪽자)

4) UTF-8 (8-bit Unicode Transformation Format)

데이터 저장과 교환에 있어서 일반적으로 쓰이는 유니코드 인코딩 방식이다. 가변 길이 문자 인코딩 방식으로, 모든 유니코드 문자를 처리하면서도 효율적으로 문자를 표현할 수 있다. 숫자와 알파벳은 ASCII와 호환되어 1바이트로, 그 밖의 문자들은 2~4바이트까지 저장되며, 현대 한글은 글자당 3바이트를 차지한다.

참 고

기타 유니코드 인코딩 방식

유니코드의 인코딩 방식에는 UTF-8이외에 UTF-16 및 UTF-32가 있다. UTF-16은 주로 쓰이는 대부분의 글자를 16비트로 처리하고, 그 범위를 넘어서는 글자는 두 개의 16비트 값을 쌍으로 묶어 처리한다. UTF-32는 모든 글자를 32비트로 처리한다.
UTF-16과 UTF-32는 하나의 글자가 차지하는 바이트 수가 일정하기 때문에 데이터를 일관성 있게 처리할 수 있는 장점이 있으나, 파일로 저장하기에는 용량을 많이 차지하는 단점이 있어서 프로그램 내부에서 문자열을 처리할 때에 주로 사용된다.

다. 기타

1) 확장한글

KS X 1001 완성자 2,350자 이외의 현대한글 8,822자의 한글을 가리킨다.
KS 완성자 (2,350자) + 확장한글 (8,822자) = 현대한글 (11,172자)

2) EUC-KR류 *)

EUC-KR 인코딩과, 그것에 뿌리를 둔 MS949, EUC-KR KS 인코딩 방식을 통틀어 'EUC-KR류' 의 인코딩이라고 부르기로 한다.

3) 채움기호

KS X 1001에서 확장한글을 표현하는 데에 쓰이는 기호이다. KS X 1001에서 0xA4D4 값(유니코드에서는 U+3164)을 가지고 있다.
확장한글을 시작할 때 사용하며, 받침이 없는 확장한글을 표시할 때 받침 대신 사용된다.
(채움기호는 화면이나 문서에 표시되지는 않지만, 편의상 이 문서에서는 '□'(점선 사각형) 기호로써 채움기호를 나타내기로 한다.)

4) KS채움쪽자 방식*)

KS 완성자 2,350자 이외의 한글(확장한글)을 표현하는 방법으로, KS X 1001 표준 규약을 따른다. 채움기호(□)를 앞에 쓰고 그 뒤에 초성, 중성, 종성 낱자가 따라오는 방식으로서 예를 들어 '샲' 이라는 글자는 '□ㅅㅑㅍ' 으로 표기한다. 하나의 한글을 표현하는 경우 채움기호를 포함한 네 글자의 낱자가 필요하고, 여덟 바이트의 메모리 공간을 차지한다. 받침이 없는 확장한글의 경우 받침 대신 채움기호를 한 번 더 사용한다.

5) NCR*) (Numeric Character Reference)

HTML Character Entity로서 "&#nnnn;" 또는 "&#xhhhh;" 의 형태로 10진수나 16진수로 Unicode codepoint 값을 써줌으로써 유니코드 문자 하나를 표현한다. 별도의 문자셋으로 볼 수는 없고, 기존의 문자셋으로는 표시할 수 없는 특정 글자들을 표시하는 용도로 XML 또는 HTML에서 사용되는 방법이다.

*) 해당 용어는 이 문서 제작을 위하여 자체적으로 정의한 용어이다.

6) 단일시스템

확장한글 문제를 다루는 데 있어서 기본 단위가 되는 시스템이다. 단일시스템은 그 구성 방식에 따라 웹 기반 시스템과 C/S 시스템으로 나눌 수 있다. 웹 기반 시스템은 Web 서버, WAS 등으로 구성되며, C/S 시스템은 클라이언트, 서버 등으로 구성된다.

7) 시스템 연계

하나의 단일시스템이 다른 단일시스템과 데이터를 주고받을 경우, 이를 가리켜 두 단일 시스템이 연계되어 있다고 말한다.

8) KS X 1026

정식 명칭은 '정보 교환용 한글 처리 지침'이며, 유니코드 한글 처리의 정규화에 대한 규격이다.
유니코드를 이용하여 한글을 표기할 때, 한글 하나를 표기하는 데에 여러 가지 방법을 사용할 수 있다. 실제로는 같은 글자인데도 글자를 표기하는 방법이 여러 가지라면 데이터를 검색하거나 비교할 때에 문제가 발생할 수 있다.
이에 따라 같은 글자를 표기하기 위한 방법을 단일화하여 정보처리 시에 문제가 발생하지 않도록 하고, 표준화를 통해 시스템 간 효율적이고 올바른 데이터 교환을 이루도록 하기 위해 KS X 1026-1이 마련되었다. (2007년 12월)
KS X 1026-1은 현대한글과 옛한글에 대한 지침을 함께 담고 있는데, 본 가이드라인과 관련하여서는 현대한글에 대한 내용만 참고하면 되겠다.
(참고로 KS X 1026-2는 KS X 1026-1에 대한 적합성 시험 기준이다.)

예를 들어, 한글 글자마디 '가'의 경우,

U+AC00 : 완성자 '가'
U+1100, U+1161 : 한글 자모 'ㄱ'과 'ㅏ'의 조합

두 가지로 나타낼 수 있는데, KS X 1026-1에서는 U+AC00만 쓸 것을 제시하고 있다.

Ⅱ. 조사·분석

1. 문제 상황
2. 현황 조사 및 분석
3. 시스템 유형 분류

본 단원에서는 확장한글 문제가 발생하게 된 상황에 대해서 그 원인을 알아보고, 현재 운영 중인 공공기관 시스템들의 문자 인코딩 사용 현황을 조사하고 분석하였다. 그리고 분석한 결과에 따라 시스템들을 적절한 유형으로 분류하였다.

1. 문제 상황

가. 문자 인코딩 문제의 이해

1) 컴퓨터와 문자 체계

가) 문자셋에 관한 최초의 표준이라고 할 수 있는 ASCII(American Standard Code for Information Interchange, 미국 정보 교환 표준 부호)는 영어 알파벳을 기초로 한 7비트 인코딩 방식으로서, 1967년에 표준이 되어 이후에 이것을 바탕으로 많은 문자 인코딩 방식이 등장하였다.

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
```

출력 가능한 아스키 문자 95개.
32번(space, 0x20)부터 126번('~', 0x7E)까지.

나) 영어 알파벳 외 다른 언어의 문자는 저마다 다른 체계로 확장, 발전하였는데, 문자셋이 글자 하나를 표현하는 데 있어서 몇 바이트의 메모리 공간을 차지하느냐에 따라서 아래와 같이 구분한다.

(1) SBCS (Single Byte Character Set)

알파벳과 일부 특수문자를 1Byte를 사용해서 표현하는 문자셋 종류

(2) MBCS (Multi Byte Character Set)

알파벳은 1Byte, 나머지 아시아권 국가의 문자 등을 2Byte 이상으로 표현하는 문자셋 종류. 그 중 2Byte로 표시하는 경우를 DBCS (Double Byte Character Set)라고 부른다.

2) 컴퓨터와 한글

가) 1987년 완성형 표준 제정 (KS C 5601-1987)

완성형 표준이 제정되면서 많은 시스템들이 KS 표준을 따르기 시작했다. 이 표준에 따라

한글 2,350자, 한자 4,888자로 표현 가능한 글자 수가 제한되었다.

(1) KS C 5601

한국 산업 규격으로 지정된 한글 문자 집합. 현재 KS X 1001로 명칭이 변경됨.

참 고

KS X 1001 완성자 한글 부분 표 (일부)

0x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3020		가	각	간	갇	갈	갉	갊	감	갑	값	갓	갔	강	갖	갗
3030	같	갚	갛	개	객	갠	갤	갬	갭	갯	갰	갱	갸	갹	갼	걀
3040	걋	걍	걔	걘	걜	거	걱	건	걷	걸	걺	검	겁	것	겄	겅
3050	겆	겉	겊	겋	게	겐	겔	겜	겝	겟	겠	겡	겨	격	겪	견
3060	겯	결	겸	겹	겻	겼	경	곁	계	곈	곌	곕	곗	고	곡	곤
3070	곧	골	곪	곬	곯	곰	곱	곳	공	곶	과	곽	관	괄	괆	

… (중략) …

0x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4820		혤	혭	호	혹	혼	홀	홅	홈	홉	홋	홍	홑	화	확	환
4830	활	홧	황	홰	홱	홴	횃	횅	회	획	횐	횔	횝	횟	횡	효
4840	횬	횰	횹	횻	후	훅	훈	훌	훑	훔	훗	훙	훠	훤	훨	훰
4850	훵	훼	훽	휀	휄	휑	휘	휙	휜	휠	휨	휩	휫	휭	휴	휵
4860	휸	휼	흄	흇	흉	흐	흑	흔	흖	흗	흘	흙	흠	흡	흣	흥
4870	흩	희	흰	흴	흼	흽	힁	히	힉	힌	힐	힘	힙	힛	힝	

(2) EUC-KR 인코딩

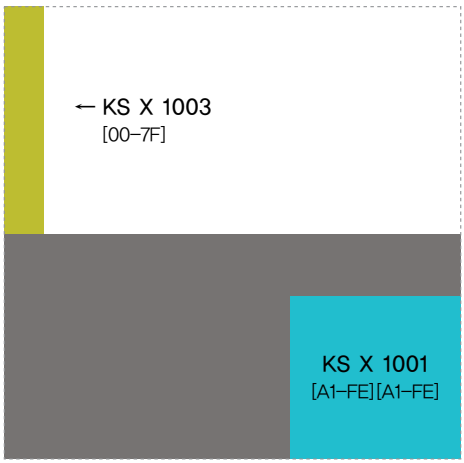
KS X 1001 (한글 및 한자) 와 KS X 1003 (로마자) 규격의 완성형 글자에 대한 8비트 문자 인코딩 방법이다.

- 로마자 문자 집합은 128 (0x80)보다 작은 값의 바이트에 배당. (KS X 1003)
- 한글 및 한자 집합은 KS X 1001 규격에 맞추어, 8번째 비트에 1을 추가(0x80) 하여 구현

예를 들어, KS X 1001의 30-2D 위치에 배당된 '강' 이라는 글자는 EUC-KR 인코딩에서 B0 AD라는 바이트 열로 표현된다.

KS X 1001		EUC-KR
0x30 0x2D	+ 0x80	0xB0 0xAD

EUC-KR 코드 배치도



(3) KS X 1001 표준과 EUC-KR 인코딩

KS X 1001 표준은 크게 완성자 2,350자에 대한 내용과 그 이외의 글자를 별도로 처리하는 방법에 대한 내용 두 가지로 구성되어 있으나 EUC-KR 인코딩은 KS X 1001 표준 중 완성자 2,350자 부분만 구현하고 나머지 8,822자에 대한 별도 처리는 구현하지 않는다.

참 고

한글 표현의 시도 (16비트 시스템에서 완성형과 조합형)

우리나라에 컴퓨터가 보급되기 시작하면서 컴퓨터에서 한글을 표현하기 위한 다양한 시도들이 있었다.

- 완성형 : 한글의 완성된 글자마다 하나하나에 코드를 배당하여 표현하는 방법.
- 조합형 : 한글의 초성, 중성, 종성에 각각 번호를 매기고, 5비트씩 메모리를 할당하여 기계적으로 조합하여 표현하는 방법.

  예 2바이트 조합형의 구성 :
  1 xxxxx yy (첫째 바이트) + yyy zzzzz (둘째 바이트) (x : 초성, y : 중성, z : 종성)

나) MS의 독자적인 한글 확장 (MS949)

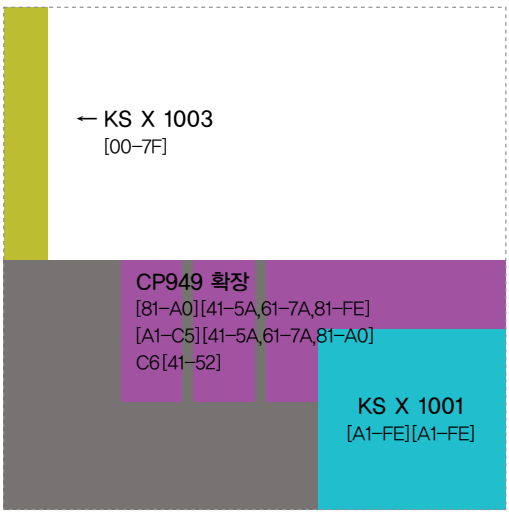
(1) MS는 한글 Windows 95부터 확장완성형 혹은 통합형 한글 코드(Unified Hangul Code)이라는 명칭으로, 이전에 사용하던 CodePage 949를 확장하여 모든 현대 한글을 수용하도록 만들었다.

(2) EUC-KR을 확장한 방식으로, EUC-KR 공간 외의 나머지 공간에 KS X 1001에 없는

8,822자의 현대 한글을 배당하였다.

(3) 한글 Windows에서 독자적으로 사용하던 방식으로, Windows XP 이후에서는 유니코드를 기본으로 사용하고 있으며, 현재는 MS에서도 호환용으로만 사용한다.

MS949 코드 배치도



다) 유니코드와 UTF-8

(1) 세계 각국의 문자를 통일된 방법으로 표현할 수 있는 문자 코드 규약이다. 'ISO/IEC 10646'으로 제정되어 있고, 일반적으로 유니코드라고 부른다.

(2) UTF-8은 유니코드를 위한 가변 길이 문자 인코딩 방식 중 하나로 모든 유니코드 문자를 표현할 수 있다.

(3) 일반적인 ASCII 문자열은 UTF-8 문자열에 속하며, 따라서 하위 호환성이 보장된다.

나. 확장한글 문제의 원인

1) EUC-KR의 확산 및 확장한글 문제 간과

가) 정보화가 본격적으로 시작된 1990년대의 한글판 운영체제 및 미들웨어는 대부분 기본 한글 인코딩 방식으로 EUC-KR 사용

나) 초기에 구축된 시스템과의 호환성 및 연계를 위해 EUC-KR을 지속적으로 채택하면서 발생 빈도가 낮은 확장한글 문제는 간과

2) MS949의 혼용으로 인한 호환성 문제 발생

가) 대부분의 단말 사용자 환경은 Microsoft사의 Windows 사용

나) Windows는 EUC-KR을 직접 지원하지 않는 대신 EUC-KR을 확장한 MS949를 사용하며 이 방식은 확장한글도 처리 가능

다) UNIX, Linux 등 EUC-KR만 지원하는 환경으로 구성된 서버는 Windows 기반 사용자 환경에서 입력된 한글 중 EUC-KR과 호환되는 글자는 처리 가능하나 MS949에서만 표현 가능한 확장한글은 정상적으로 처리하지 못할 수 있음

라) 응용프로그램 및 미들웨어의 한글처리 구현 방식에 따라 문제 발생 여부가 달라질 수 있어 오류 발생을 인지하기 어려움

다. 문제 발생 예

1) 인터넷 민원 신청 페이지

신청내역 홈 > 나의민원 > 나의민원처리결과 > 신청내역

신청인 정보

접수번호	20100124-	민원신청 확인증	민원신청 확인증보기
성명		주민등록번호	-*******

접수 목록

※ 처리 번호는 실제 접수처리 기관의 처리번호 입니다.

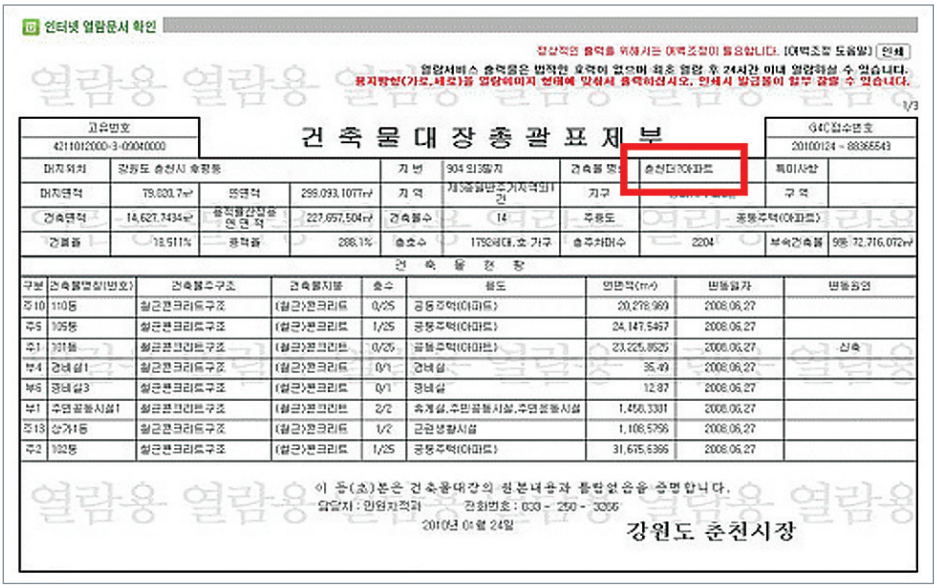
번호	민원명 (처리번호)	수령방법	부수	보완보정	유효기간	진행상태
001	주민등록표등본교부 (2010-3200160-)	일반보통우편	1			처리중

민원인 수령지주소

서울특별시 관악구 신림동 포스코 더 ? 아파트 101 동 101 호 우편번호(151-893)

아파트 이름에 포함된 '샾' 글자가 제대로 표시되지 못하고 '?'로 나타난 것을 볼 수 있다.

2) 인터넷 민원 열람 페이지


인터넷 열람문서 확인

건 축 물 대 장 총 괄 표 제 부

건축물 명 [샾]

건축물 대장의 '건축물 명'에 포함된 '샾' 글자가 제대로 표시되지 못하고 '?'로 나타난 것을 볼 수 있다.

참 고

확장한글 문제 발생 영역

확장한글 문제는 한글로 표기되는 데이터가 사용되는 곳에서는 어디에서든 발생할 수 있다. 그중 중요한 몇 가지를 예를 들면 다음과 같다.

- e-메일메일 전송 시 인코딩 타입(EUC-KR)에 따라 제목 및 본문에서 표현하지 못하는 문자가 발생할 수 있다.
- 웹사이트HTML 구성 시 헤더(meta tag) 부분에 기술된 인코딩 타입(EUC-KR)에 따라 지원하지 않는 문자가 발생할 수 있다.
- 클라이언트 프로그램C/S 시스템에서 응용프로그램을 유니코드 기반으로 만들지 않았을 경우, OS가 제공하는 환경(특정 코드페이지 또는 입력기)에 제약되어 입출력이 제한될 수 있다.
- 데이터베이스제품 종류, 설정 또는 버전에 따라 확장한글 데이터를 저장하지 못하는 데이터베이스가 있을 수 있다.

2. 현황 조사 및 분석

가. 정보시스템 현황 조사

정부 중앙부처와 지방자치단체 시스템들의 구성과 문자 인코딩 정보 및 현황 자료를 취합하여, 총 3,011개의 시스템을 조사/분석하였다.
각 시스템은 크게 '웹 기반' 시스템과 'C/S 기반' 시스템으로 구분된다.

1) 웹 기반 시스템

웹 기반 시스템은 DBMS 영역, WAS 영역, OS 영역, Web 영역으로 구성되어 있고, 각 영역에서 제품명/버전, 문자 인코딩을 조사하였다.

예

이름	구분	DBMS		WAS		OS		Web	
국토해양부/전자문서 시스템	웹 기반	oracle 9i	EUC-KR	tomcat	UTF-8	solaris 9	EUC-KR	apache	EUC-KR

참 고

이 조사 항목 중에서는 DBMS에서 데이터를 저장할 때 어떤 문자 인코딩을 사용하고 있는지, 그리고 웹 애플리케이션 영역인 WAS와 Web에서 어떤 문자 인코딩으로 데이터를 처리하고 있는지에 대한 정보를 바탕으로, 웹 기반 시스템의 유형별 분류 기준과 문제 해결 방안을 마련하였다.

2) C/S 시스템

클라이언트, 미들웨어, DBMS 영역으로 구성되어 있고, 각각의 제품명/버전과 DBMS 영역의 문자 인코딩 정보를 조사하였다.

예

이름	구분	클라이언트	미들웨어	DBMS	
경남 거제시/지리정보시스템	C/S	delphi 5.0	arcsde 9.2	oracle 9i	EUC-KR

참 고

C/S 시스템에 대한 조사 항목에서는 클라이언트와 미들웨어에서 어떤 문자 인코딩을 사용하고 있는지 명시적으로 나와있지는 않지만, DBMS에서 사용하는 문자 인코딩을 기준으로 클라이언트와 미들웨어가 운영되도록 애플리케이션이 만들어져있다고 볼 수 있다. 이를 바탕으로 C/S 시스템의 유형별 분류 기준과 문제 해결 방안을 마련하였다.

3) 정보시스템 현황 통계

국내 152개 공공기관의 정보시스템 3,011개에 대한 문자 인코딩 사용현황 통계.

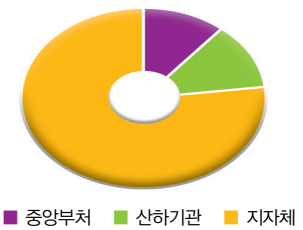
• **조사 대상**

(2010년 6월 기준)

기관 수 152
정보시스템 수 3,011

기관 구분	기관 수
중앙부처	18
산하기관	17
지자체	117
기관합계	152

조사 대상 기관



• **시스템 구분**

Web 기반	2,614	86.82%
C/S 기반	453	15.04%
Total	3,067	101.86%

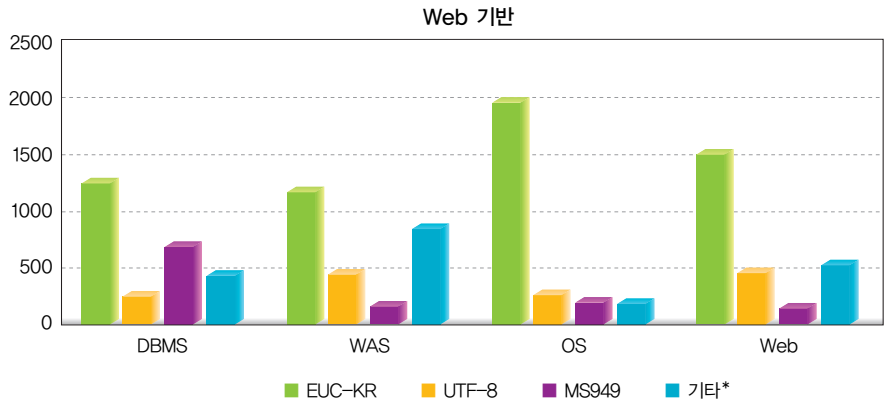
* 중복 해당되는 시스템 존재

• **시스템 구성**

Web 기반	EUC-KR	UTF-8	MS949	기타*
DBMS	1,240	244	697	433
WAS	1,167	445	171	831
OS	1,958	267	201	188
Web	1,505	455	152	502

C/S 기반	EUC-KR	UTF-8	MS949	기타*
DBMS	259	44	53	97

* 취합 시 표시가 없거나 분명하지 않은 결과들을 '기타'로 분류함('기타'로 표시된 것들 중 상당수는 EUC-KR로 예상)



나. 연계정보 현황 조사

1) 연계정보

외부의 타 시스템과 연계되어 데이터를 주고받는 시스템들에 대해서는 연계시 사용되는 인코딩 방식을 파악하기 위해 연계기관/시스템, 연계정보, 인코딩 방식, 연계방식과 같은 별도의 정보를 추가로 조사하였다. 여기에서는 외부와 연계할 때 어떤 문자 인코딩으로 데이터를 주고받느냐 하는 것이 중요한 정보이다.

예

기관/시스템명	연계기관/시스템	연계정보	송신	수신	인코딩	연계방식
충남 서산시/ 한국토지정보시스템	행정안전부/ 농촌행정시스템	토지정보	O	O	EUC-KR	웹서비스

2) 연계 현황 통계

• **조사 대상**

(2010년 6월 기준)

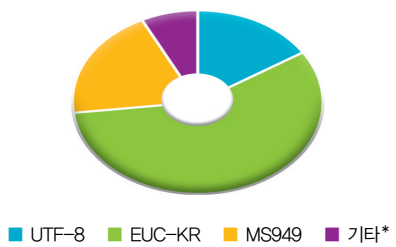
시스템 수 1,537

• **문자 인코딩 구분**

UTF-8	244	15.88%
EUC-KR	880	57.25%
MS949	300	19.52%
기타*	113	7.35%
Total	1,537	100.00%

* 표시가 없거나 불분명한 결과들을 '기타'로 분류

시스템간 연계 시 인코딩 사용 현황



다. 대표시스템 현황 분석

1) 대표시스템 선정

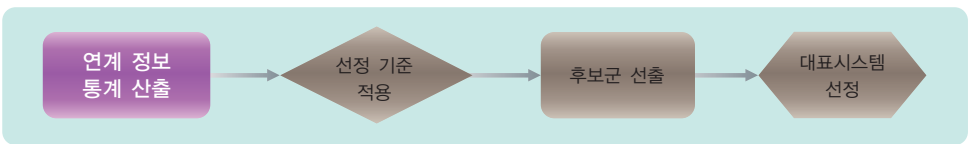
가) 대표시스템 선정 목적

상세한 원인 분석 및 구체적인 대안 제시를 위해 실제 운영 중인 시스템 중 대표시스템을 선정하여 분석

나) 대표시스템 선정 기준

(1) 확장한글 문제 발생이 예상되는 시스템 (EUC-KR 인코딩 사용)
(2) 타 시스템과의 연계가 많은 시스템
(3) 주민정보와 같이 사용자가 자주 접하는 시스템 (대민서비스)
(4) 유형 분류 중 많은 비중을 차지하는 유형에 해당하는 시스템

다) 대표시스템 선정 과정



(1) 국가 정보시스템 취합 자료를 기초로 하여, 연계 정보 통계 산출
(2) 연계 정보가 다수인 정보시스템 중 선정 기준에 적합한 시스템 후보군 선출

라) 대표시스템 선정 결과

(가) 다양한 시스템과 연계되며 전자민원 처리의 창구 역할을 하는 '민원24' 선정
(나) 16개 시도의 행정업무 처리 지원 시스템인 '시도행정시스템'과 230개 시군구에서 행정업무 처리 지원 시스템인 '새올행정시스템' 선정

2) 대표시스템 개요

민원24는 시간과 장소에 구애받지 않고 인터넷으로 필요한 민원을 처리할 수 있도록 하는 정부 민원포털이다. 이전에는 G4C라는 이름이었던 민원24는 행정안전부 주관으로 인터넷 상에서 행정기관이 보유한 5,000여 종의 민원사무에 대한 안내 및 신청, 20여 종의 민원에 대한 열람 서비스를 제공하고, 500여 종의 인터넷 민원발급 서비스를 제공하는 사이트이다.

24시간 언제 어디서나 제공되는 전자민원서비스를 의미하는 '민원24'는, 5,000여 종의 민원사무에 대한 안내 및 신청, 20여 종의 민원에 대한 열람 서비스를 제공하고, 500여 종의 인터넷 민원발급 서비스를 제공하는 정부민원포털 사이트이다.

민원24는 새올행정시스템, 출입국관리정보시스템, 국가화재정보시스템 등 각 기관의 정보시스템과 연계되어 민원 서비스를 제공하고 있다.

민원24의 민원 서비스 종류

분야별 민원	설명
개인과 가정 (208)	한 개인이 태어나서 일생을 살면서 겪게 되는 출생, 교육, 혼인, 여행, 사망 등 개인과 가정에 관련된 민원사무
부동산과 재산 (129)	토지, 농지, 수목, 주택, 아파트 등 부동산과 재산에 대해 등기, 임차 등의 권리를 행사하는 행위에 관한 민원사무
세금과 재정 (406)	국가가 부과하는 국제조세, 국세와 지방 공공 단체가 부과하는 각종 지방세 등 세금 및 재정에 관련된 민원사무
기업과 경제 (349)	기업이 합리적이고 효과적인 생산활동을 통해 경제가치를 창출해 낼 수 있도록 기업 설립에서부터 제반 경영에 관련된 민원사무
농림수산과 천연자원 (526)	농업, 축산업, 임업, 어업, 광업 등 자연으로부터 직접 천연자원을 얻거나 생산하는 활동과 관련된 민원사무
제조건설과 개발 (581)	농림수산물, 천연자원을 원료로 생활에 필요한 물건이나 에너지를 생산하는 제조업, 건설업, 에너지 사업 등에 관한 민원사무

분야별 민원	설명
서비스와 생활 (559)	자연에서 얻거나 기업에서 생산한 물품을 소비자에게 직접 판매하거나 각종 서비스를 통해 간접적으로 제공하는 사업에 대한 민원사무
사회보장과 복지 (947)	질병이나 빈곤, 실업, 산업재해 등으로 생기는 국민생활 문제를 국가가 제도적으로 보장하고, 제도화하는 사회보장과 복지에 관련된 민원사무
자연과 환경 (236)	우리를 둘러싸고 있는 자연 환경인 토지, 수질, 대기 등의 오염에 대한 환경정책 및 자연보호정책, 그리고 기상에 대한 민원사무
문화와 여가 (239)	개인의 삶을 윤택하게 해주는 영화, 공연, 문화시설, 문화재 등 문화생활과 스포츠, 관광 등 여가생활에 관련된 산업에 대한 민원사무
통일과 국방 (95)	남북간의 방문/교류/지원 등 통일을 위한 사업, 국방의 의무인 병역, 북한탈북자에 대한 북한이탈주민보호 등에 관한 민원사무
교통과 물류 (383)	생산된 물품을 자동차, 철도, 선박, 항공기 등을 통해 운송하고 유통, 보관하는 운수사업 및 물류사업에 관련된 민원사무

민원24의 민원 서비스 선택 화면

나) 시도/새올행정시스템

시도/시군구 행정정보시스템으로 전국 시도청 및 시군구청의 주요 행정업무를 종합정보화한 시스템이다. 시도/새올 행정시스템은 행정내부처리를 담당하는 영역과 대국민 행정서비스를 담당하는 영역으로 구성되어 있다. 시도행정정보시스템은 총 24개 시도행정 업무 중 자치행정 등 18개 업무를 시스템으로 구축하였고, 새올행정시스템은 시군구의 총 31개 행정 업무 중 내부행정 등 22개 업무를 시스템으로 구축하여 지방행정의 표준화, 전산화를 지원하고 있다.

시도/새올행정시스템 개요도

3) 대표시스템 구성

가) 대표시스템 구성

(1) 민원24는 민원서비스 제공을 위해, 다양한 민원처리기관의 정보시스템과 연계하여 동작하며, 이기종 정보시스템들과의 연계를 위해 가이스(GAIS)[1] 모듈을 사용하여 데이터를 주고받는다.

(2) 민원24 내에는 회원정보와 민원신청 내역을 기록/관리할 수 있는 회원DB를 보유하고 있으며, 민원 서비스에 필요한 각종 데이터는 해당 정보시스템과 연계하여 정보서비스를 제공한다.

예 주민등록등본 발급 시, 시군구민원행정시스템과 연계하여 해당 문서를 발급

나) 데이터 흐름도

(1) 민원24와 연계되는 정보시스템 간의 통신이 진행될 때, 두 시스템 각각 위치한 GAIS 모듈에서 정상적인 데이터 교환을 위한 교환관리 기능을 수행한다. (문자 인코딩 변환 등)

(2) GAIS는 공통모듈(각 시스템에서 동일한 모듈 사용) 형태로 사용되고 있으며, 각 시스템의 구성환경에 따라 별도의 방식으로 동작한다.

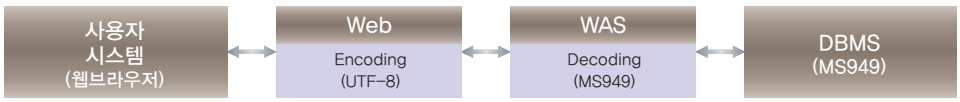
(3) GAIS는 소켓(socket) 통신으로 동작하며, 시스템 간 주고받는 데이터는 표준 XML을 따른다.

4) 시스템 현황 분석

확장한글 처리와 관련하여 민원24 및 민원24와 연계되는 정보시스템에 대한 현황 분석을 수행한다.

1) 가이스 (GAIS, Government Application Integration System)
정부기관 사이의 연계 시 데이터 교환관리를 수행하는 기능을 갖는 통합 모듈 (socket 통신)

가) 확장한글 처리 프로세스 ① : 민원24



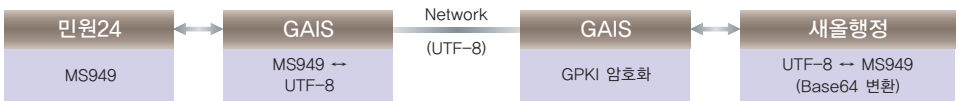
(1) 민원24 자체에서 운영 중인 DBMS(회원 DB)는 EUC-KR의 확장형인 MS949 문자 인코딩을 사용하고 있으며, 프로그램인 WAS와 Web은 EUC-KR 인코딩을 사용하고 있다.

(2) 단일시스템 내에서의 데이터 흐름을 살펴보면, 사용자가 입력한 데이터는 Web(서버)를 거치면서 인코딩(UTF-8)되며, 다시 WAS에서 디코딩(MS949) 과정을 거치면서 DBMS로의 저장이 이루어진다.

(3) 사용자 입력하는 데이터 중 확장한글에 해당하는 문자가 있더라도 인코딩과 디코딩 과정을 거치면서 MS949를 사용하는 DBMS에 저장된다.

(4) 과거(2010년 6월 이전)에는, Web과 WAS에서 별도의 인코딩/디코딩 과정 없이 모두 EUC- KR 인코딩으로 처리되었기에, DBMS가 MS949이더라도 확장한글 부분이 손상된 채로 저장되는 형태였다.

나) 확장한글 처리 프로세스 ② : 민원24 + 새올행정시스템



(1) 민원24와 새올행정시스템 간에 서식 민원을 처리하는 경우에는 양측의 GAIS 모듈이 UTF-8 데이터를 송수신한다. 이때 UTF-8 데이터는 Base64로 다시 한 번 인코딩하여 전송된다.

(2) 민원24와 새올행정시스템의 각각의 DBMS에서는 내부적으로 MS949 형태로 데이터를 저장/처리한다.

(3) 경우에 따라 UTF-8 변환이 아닌, 양측의 인코딩을 MS949로 통일하여 처리하고 있기도 하다.

예 주민등록등초본 발급

다) 확장한글 처리 프로세스 ③ : 시도/새올행정시스템

(1) 새올시스템 내 각각의 단일시스템은 서로 다른 인코딩 방식으로 데이터를 다루고 있다.

(가) 시도행정정보시스템 (웹 기반, Java) : 기본적으로 EUC-KR 인코딩을 사용한다. 확장한글은 NCR 방식으로 처리하여 데이터를 전송하기 때문에 별도 변환 없이 웹 브라우저에서 볼 수 있도록 하고 있다.

예 '샶' = '샾'

(나) 새올행정시스템 (웹 기반, Java) : MS949 인코딩으로 구축하여 내부적으로 확장한글 문제를 처리하고 있다.

(2) 시도/새올행정시스템 내에서 정보시스템들이 데이터를 주고받을 때, EUC-KR을 사용하는 Delphi 환경의 새올행정시스템(C/S)으로부터

① **새올행정시스템(웹)으로** (즉, EUC-KR in Delphi → MS949)

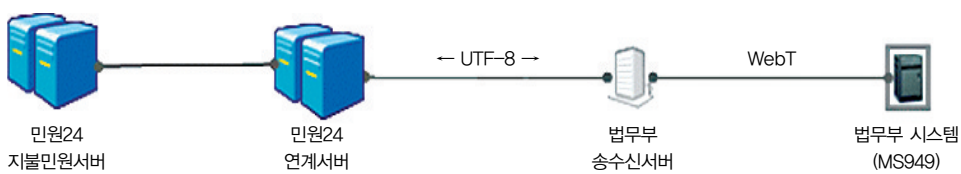
② **시도행정정보시스템(웹)으로** (즉, EUC-KR in Delphi → EUC-KR +NCR)

데이터를 보내게 되는데, 각각 다른 문자열 처리 방식을 사용하고 있기에 확장한글이 포함된 데이터가 손상되는 문제가 발생한다.

라. 기타 시스템 현황 분석

1) 시스템 현황 분석 ① – 출입국관리정보시스템

서울출입국관리사무소에서 관리하는 법무부 출입국관리시스템은 민원24 시스템과 연계되어 다섯 종의 증명서를 실시간 온라인 발급할 수 있도록 해준다. 이러한 민원 처리 시 확장한글 관련 문제가 발생하지 않도록 하기 위해서 민원24와의 연계 부분을 수정하는 작업 중에 있다. (2010년 8월 기준)



가) C/S 시스템으로서 DB는 MS949 데이터를 저장하도록 되어 있다.

나) 데이터의 입력은 클라이언트 단말에서 이루어지고, MS949 문자 인코딩을 사용하므로 확장한글이 깨지는 문제는 없다.

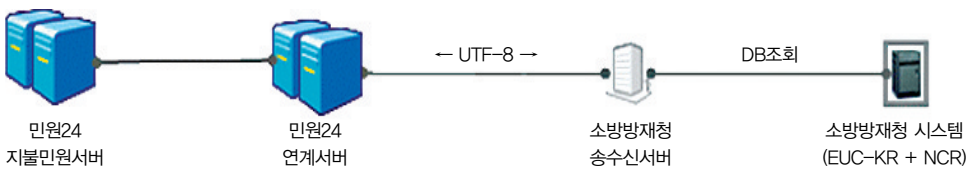
다) 외국인에 대한 정보를 입력하는 경우 한자가 포함되어 있을 수 있으나 필수 입력 사항이 아니어서 한자 데이터 입력은 드물고, 아직은 한자 데이터가 깨지는 경우가 발생하지 않았다.

라) 민원24의 송수신 모듈(GAIS)이 데이터 처리를 UTF-8로 일원화함에 따라, 향후 민원24 연계서버와 법무부 송수신 서버 사이에서는 UTF-8 데이터를 주고받게 될 것이다.

마) 민원24 외에 10여 개의 시스템과 연계되어 있고, 연계 방식은 각각 다르나 DB의 데이터(MS949 형식)를 그대로 전달해주고 있는 상황이다.

2) 시스템 현황 분석 ② – 국가화재정보시스템

소방방재청에서 관리하는 국가화재정보시스템은 민원24 시스템과 연계되어 화재증명 발급 서비스를 제공할 수 있도록 해준다. 이러한 민원 처리 시 확장한글 관련 문제가 발생하지 않도록 하기 위해서 민원24와의 연계 부분을 수정하는 작업 중에 있다. (2010년 8월 기준)



가) 웹 기반 시스템으로서, 내부적으로 NCR 방식으로 확장한글 처리를 한 상황이다.

나) 입력 데이터는 웹을 통해 들어오고, WAS에서 NCR 방식 인코딩을 한 후 DB에 저장한다. 확장한글 문제가 발생하지는 않는 상황이다.

다) 2008년부터 NCR 방식을 도입했으며, 그 이전 자료들에는 확장한글이 깨진 데이터가 들어있다. DB 클렌징을 하지 않은 상태이다.

라) 민원24 연계서버는 소방방재청 송수신서버를 사이에 두고 소방방재청 시스템과 데이터를 주고받고 있고, 송수신서버의 GAIS 모듈이 UTF-8 인코딩을 사용할 계획이므로 향후 민원24와는 UTF-8로 데이터를 송수신하게 된다.

마) 민원24 이외의 시스템들과의 연계에서는 현재 DB에 있는 데이터(NCR 방식)를 그대로 전달하고 있다.

3. 시스템 유형 분류

본 가이드라인에서는 단일시스템과 그들 간의 연계에 있어 확장한글 문제가 발생할 수 있는 구간을 대상으로 해결책을 찾고자 한다.
단일시스템의 경우에는 데이터 입력에서 DB 저장, 그리고 출력까지를 대상으로 하고, 연계의 경우에는 단일시스템 간의 정상적인 데이터 송/수신을 위한 API를 대상으로 한다.
각 정보시스템별로 환경이 모두 다르지만, 기준에 따라 몇 가지 문제 유형으로 분류하여 각 유형별로 해결 방안을 마련하고자 한다.

가. 유형 분류 기준

1) 입출력 응용프로그램을 수정할 필요가 있는가?

웹 기반 시스템에서의 Web과 WAS 부분 또는 C/S에서의 응용프로그램을 수정할 필요가 있는지 확인한다. 이는 프로그램이 텍스트 데이터를 입출력할 때에 유니코드(또는 UTF-8)가 아닌 다른 인코딩 방식으로 처리할 경우에 해당하는 것으로, 해당되면 'A' 기호로 표시하기로 한다. (웹 기반 시스템인 경우 'A1' 으로 표시하고, C/S 기반 시스템인 경우 'A2' 로 표시한다.)

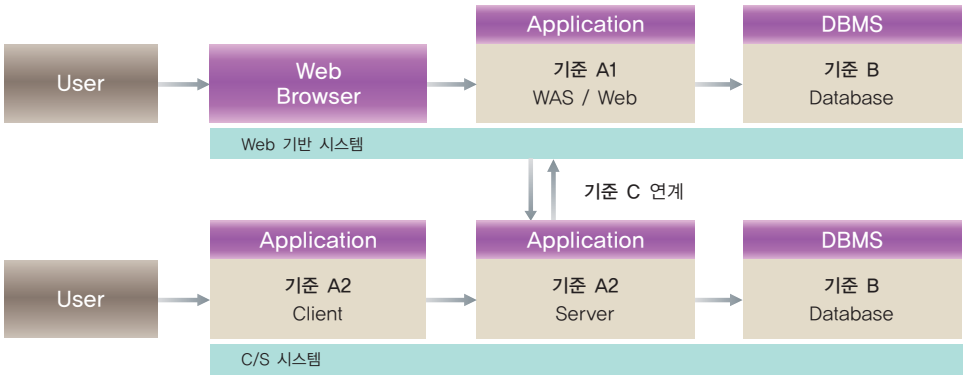
2) DB 데이터 또는 설정을 수정할 필요가 있는가?

DB에 저장되는 데이터의 인코딩 방식이 유니코드(또는 UTF-8)가 아닌 경우에 해당하는 것으로, 해당되면 'B' 기호로 표시하기로 한다.

3) 외부 송수신 방식을 수정할 필요가 있는가?

연계된 시스템과 송수신하는 데이터의 인코딩 방식이 UTF-8이 아닌 경우에 해당하는 것으로, 해당되면 'C' 기호로 표시하기로 한다.

나. 분류 기준 상세 설명



1) 기준 A : 입출력 프로그램 관련

응용프로그램 영역은 시스템 구성 방식에 따라 웹 기반 시스템에서의 프로그램 영역과 C/S 시스템에서의 프로그램 영역, 두 가지로 나눠볼 수 있다.

가) 웹 기반 시스템 – Web, WAS 관련

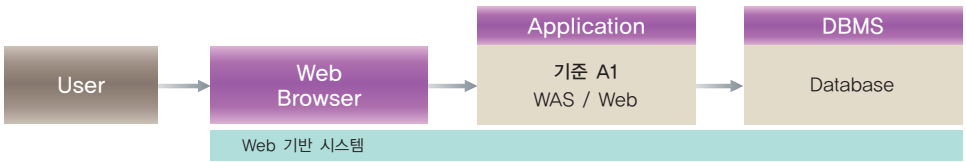
웹 기반 시스템에서는 Web 서버와 WAS가 응용프로그램으로서 동작한다. 사용자가 어떠한 데이터를 입력해서 그것이 시스템의 DB에 저장되기까지의 입력 과정에서, 먼저 사용자는 PC에서 웹브라우저를 통해 시스템에 접속하게 된다. 이때, 브라우저에 나타난 웹페이지를 통해 사용자의 데이터가 입력되는데, 데이터 문자열의 인코딩은 그 웹페이지의 인코딩 방식을 그대로 따르게 된다. 즉, EUC-KR 인코딩 방식이 지정된 웹페이지에서 입력한 데이터는 EUC-KR 인코딩 문자열이 되어 웹 서버로 전송되고, UTF-8 인코딩 방식이 지정된 웹페이지에서 입력한 데이터는 UTF-8 인코딩 문자열이 되어 웹 서버로 전송된다.

(웹페이지에 어떤 인코딩이 지정되었는지는 웹브라우저의 '소스 보기' 기능을 이용하면 확인할 수 있다. 웹페이지 소스의 시작 부분에서

〈meta http-equiv="content-type" content="text/html;charset=UTF-8" /〉

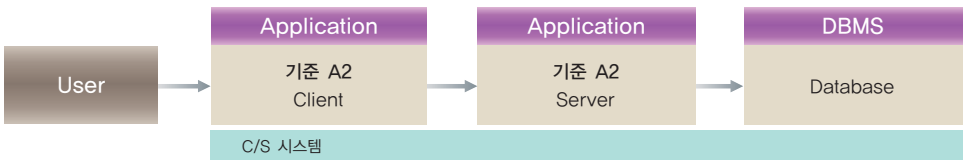
와 같은 문장을 찾을 수가 있는데 이 예에서는 'charset=UTF-8' 이라는 부분이 이 웹페이지에 UTF-8 인코딩이 쓰였다는 것을 알려 준다.)

따라서 데이터를 입력하는 웹페이지의 인코딩 방식이 EUC-KR로 되어 있다면, 사용자가 브라우저에서 입력한 확장한글은 Web과 WAS로 전송되는 과정에서 깨지는 경우가 발생할 수 있다. 그러므로 Web과 WAS 부분이 어떤 인코딩 방식으로 되어 있는지를 하나의 분류 기준으로 삼을 필요가 있다. (기준 A1)



나) C/S 기반 시스템 – 클라이언트 응용프로그램 관련

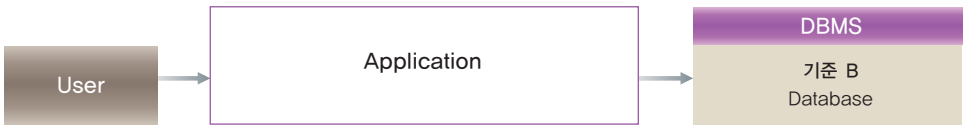
C/S 기반 시스템에서는 사용자가 입력하는 데이터가 독립적인(stand-alone) 클라이언트 프로그램을 통해 서버로 전달된다. 이 클라이언트와 서버 프로그램들이 어떤 인코딩 방식으로 문자열 데이터를 전송하느냐에 따라서 확장한글이 깨지는 경우가 발생할 수 있으므로, 이것을 분류 기준으로 삼을 필요가 있다. (기준 A2)



2) 기준 B : DB 관련

DB 문자 인코딩이 데이터 저장 과정에서 확장한글이 깨질 우려가 있는지가 하나의 기준이 된다. DB 인코딩이 UTF-8인 경우 수정이 필요 없지만, EUC-KR이나 MS949로 되어 있는 경우 별도의 수정이 필요할 수도 있다.

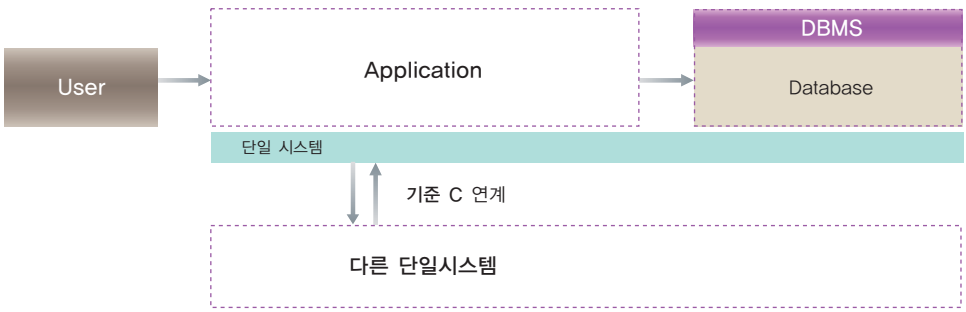
시스템 각 부분을 UTF-8 인코딩으로 전환하는 데에 있어서, 기존에 저장된 데이터들이 어떻게 인코딩되어 있는지도 중요한 문제이므로, DB에서 어떤 인코딩 방식을 쓰고 있는지가 문제 유형 분류의 중요한 기준이라고 할 수 있다.



3) 기준 C : 외부 송수신 관련

각각의 단일시스템이 독자적으로 운영되고 그 내부에서만 데이터가 사용된다면 확장한글은 큰 문제가 아닐 수 있다. 하지만 실제로는 시스템들끼리 복잡하게 얽혀있고 데이터를 주고받는 방식들도 다르기 때문에, 데이터의 호환성이 깨지면서 확장한글 문제가 크게 문제가 된다.

따라서 시스템 간 송수신에 있어서 하나의 표준 인코딩 방법을 정하고 데이터를 주고받도록 해야 문제를 해결할 수 있다. 그러므로 송수신 인코딩 방법이 UTF-8이 아닌 경우가 문제 유형 분류의 기준이 될 수 있다.



참 고

OS의 영향 관계

확장한글 문제 해결을 위한 유형 분류 기준으로서 시스템 OS의 문자 인코딩 설정이 무엇인지를 크게 고려할 필요는 없다. 시스템 구성의 가장 외부에 위치하는 웹 브라우저(또는 클라이언트 프로그램)로부터 가장 내부의 DB에 이르기까지 문자열 데이터가 전달되어 가공/처리되는 과정에서는 OS의 영향을 직접 받지는 않는다. 즉, OS 위에서 동작하는 응용프로그램들이 데이터를 처리하고 주고받는 과정에는 응용프로그램의 설정과 인코딩 방식이 데이터 처리에 영향을 주는 요소이며, 글자가 깨지는 이유는 데이터를 직접 다루는 응용프로그램들이 그것을 제대로 변환하지 못하기 때문이다. (2000년 이후의 대부분의 OS는 유니코드 입출력 처리 가능)

예를 들어 유닉스와 리눅스 운영체제에는 오래 전부터 iconv라는 이름의 프로그램 및 표준 API 라이브러리가 포함되어 있다. 이 API는 EUC-KR, MS949, UTF-8을 비롯한 많은 인코딩 타입 간의 데이터 변환을 지원하고 있어서, 유닉스와 리눅스 운영체제 기반의 응용프로그램들은 이 API를 통해 원하는 인코딩 방식으로 문자열을 변환하여 처리할 수 있다. 즉, OS는 어떤 인코딩의 문자열이든 모두 사용할 수 있는 기반을 제공하고 있고, 그것을 이용해서 어떤 식으로 동작할 것인가는 응용프로그램을 어떻게 만드느냐에 따라 다르다고 할 수 있다.

또한 C/S 시스템에서의 응용프로그램의 경우, windows 9x 계열에서 동작하도록 하기 위해 MS949(또는 MBCS) 기반으로 컴파일 되어 있을 수 있다 (Widows 2000 이전 버전). 그러나 이 또한 OS 문제는 아니며 응용프로그램을 수정해야 될 문제이므로 OS의 영향이 있는 것은 아니다.

▶ **OS와의 영향 관계가 존재하는 경우의 예시**

대부분의 경우 위와 같이 OS의 영향을 받지 않겠지만, 응용프로그램들의 처리 과정에서 입력된 데이터를 파일로 입/출력하는 경우, 파일명에 한글이 포함되면 해당 글자가 인식되지 않아, 파일 입/출력에 문제가 발생하는 경우가 있을 수 있다.

예를 들어 사용자가 게시판 등에 한글이름으로 된 파일을 첨부할 경우, 해당 파일을 응용프로그램 서버에 업로드 및 다운로드하는 과정에서 OS의 문자 인코딩 영향을 받아 파일명이 복원되지 않을 수 있다. (OS의 인코딩이 확장한글 미지원 또는 한글 미지원)

참 고

개발 언어의 영향 관계

C/S 시스템에서 응용프로그램을 작성할 때, OS가 제공하는 입출력 API와 콤포넌트를 사용하기 때문에 어떤 언어로 개발하는가에 따라 차이가 나타나지는 않는다. 다만 어떤 API를 사용하여 어떤 문자 인코딩 기반으로 응용프로그램을 작성하는지에 따라 확장한글 문제가 발생할 수도 있는 것이다.

참고로 Java는 모든 문자열 처리를 유니코드 기반으로 수행하고, 입출력 시에 필요한 경우에 특정 인코딩으로 변환/처리하는 과정을 거친다. 이때 입출력 단계에서 어떤 인코딩을 사용하도록 했는가에 따라서, 내부적으로는 유니코드로 문자열 처리를 하는 Java 응용프로그램이라도 확장한글 문제가 발생할 수 있다.

요컨대 응용프로그램에서 확장한글 문제 발생 여부는 어떤 개발 언어와 어떤 OS를 사용했는가보다는, 응용프로그램이 어떤 문자 인코딩으로 데이터를 입출력하고 처리하도록 제작되었는가에 따라 달라진다고 할 수 있다.

다. 시스템 유형 분류

1) 유형 분류표

위의 기준에 따라 유형을 나누어보면 아래와 같이 분류할 수 있다.
(유형 분류 방법은 '# 첨부11 유형 판별 차트'를 참고한다.)

유형	기준 A	기준 B	기준 C	비고
Type A	X			응용프로그램 수정 필요
Type B		X		DB 수정 필요
Type C			X	송수신 수정 필요
Type AB	X	X		응용프로그램, DB 수정 필요
Type AC	X		X	응용프로그램, 송수신 수정 필요
Type BC		X	X	DB, 송수신 수정 필요
Type ABC	X	X	X	응용프로그램, DB, 송수신 수정 필요
Type 0				수정 불필요
Type X	?	?	?	유형 분류 불가

2) 유형 설명

아래는 각 유형들에 대한 설명이다.

가) Type 0

내부/외부의 모든 문자셋이 유니코드(UTF-8) 문자셋을 쓰고 있는 바람직한 시스템.
시스템 내부적으로 확장한글 문제가 발생하지 않고, 연계된 다른 시스템들과 통신할 때에도 데이터를 UTF-8로 보낼 수 있으므로 별도의 추가 작업이 필요 없는 시스템 유형이다.

나) Type A – Type A1 또는 A2

WAS/Web 응용프로그램 또는 C/S 기반의 응용프로그램 수정이 필요한 시스템.
넓게 보아 응용프로그램 영역에서 UTF-8이 아닌 다른 인코딩 방식을 사용하고 있어서,

문자열 데이터를 다룰 때 확장한글이 깨질 우려가 있는 시스템 유형이다.
웹 기반과 C/S 기반의 차이에 따라 다음 두 가지의 세부 타입으로 구분한다.

※ A1 : WAS, Web을 수정해야 하는 웹 기반 시스템
※ A2 : 클라이언트-서버 프로그램을 수정해야 하는 C/S 시스템

다) Type B

DB 데이터 처리에 있어 수정이 필요한 시스템.
DB에서 데이터를 저장할 때 유니코드(또는 UTF-8)를 사용하지 않아서, 확장한글 데이터가 처리되지 않는 시스템 유형이다.

라) Type C

연계되는 시스템과의 송수신 방식 변경이 필요한 시스템.
단일시스템 내의 각 부분들이 UTF-8 인코딩으로 처리되어, 내부에서 데이터를 처리할 때는 확장한글 문제가 발생하지 않으나, 연계되는 시스템으로 데이터를 전송할 때는 다른 방식을 사용하고 있어 확장한글 문제가 발생하는 시스템 유형이다.

마) Type AB (A1B / A2B)

응용프로그램 입출력과 DB의 데이터 처리의 수정이 필요한 시스템.
Type A와 Type B에 모두 해당하는 유형으로, DB와 응용프로그램에서 확장한글 문제가 발생할 수 있는 시스템 유형이다.

※ A1B : 웹 기반 시스템
※ A2B : C/S 기반 시스템

바) Type AC (A1C / A2C)

응용프로그램 입출력과 연계되는 시스템과의 송수신 방식에서 수정이 필요한 시스템.
DB는 UTF-8 인코딩을 쓰고 있어서 확장한글이 문제되지 않지만, 응용프로그램 영역에서 데이터를 다루거나 시스템 연계 시 다른 인코딩 방식을 사용하고 있어 확장한글이 깨질 우려가 있는 시스템 유형이다.

※ A1C : 웹 기반 시스템
※ A2C : C/S 기반 시스템

사) Type BC

연계되는 시스템과의 송수신 방식과 DB 데이터 처리의 수정이 필요한 시스템.
응용프로그램 영역은 UTF-8 인코딩을 사용하고 있어 확장한글 데이터가 전달되지만, DB에 데이터를 저장할 때와 시스템 연계 시, UTF-8 인코딩을 사용하지 않아 확장한글 데이터가 깨질 위험이 있는 시스템 유형이다.

아) Type ABC – Type A1BC 또는 A2BC

응용프로그램과 DB, 시스템 연계의 모든 부분에 수정이 필요한 시스템.
DB 영역과 WAS, Web 등의 응용프로그램 영역, 그리고 연계되는 시스템과의 송수신 방식 등 모든 부분에서 확장한글 문제가 발생할 우려가 있는 시스템 유형이다.

※ A1BC : 웹 기반 시스템
※ A2BC : C/S 기반 시스템

자) Type X

특정 유형으로 분류하기 어려운 시스템.
시스템에 대한 정보가 부족하여 적절히 분류하기가 어려운 시스템 유형이다. 이 유형의 시스템들은 개별 문제점을 파악하고 그에 따라 적절한 조치를 취해야 한다.

라. 분석 및 통계

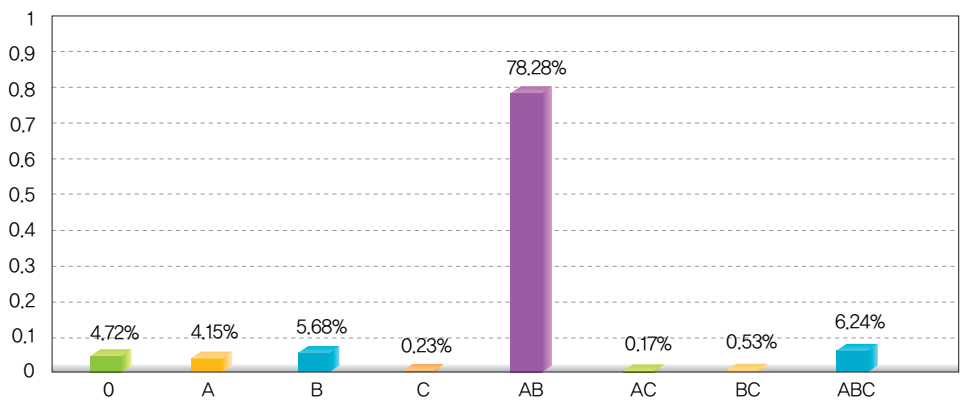
기초 조사 자료를 가지고 위의 분류 기준에 따라 전체 3,011개의 단일시스템들을 유형별로 분류한 결과는 다음과 같다.

(2010년 6월 기준)

유형		시스템 수	비율
Type 0		142	4.72%
Type A	(A1)	125	4.15%
Type B		171	5.68%
Type C		7	0.23%
Type AB		2,357	78.28%
	(A1B)	1,992	66.16%
	(A2B)	326	10.83%
	(A1A2B)	39	1.30%
Type AC	(A1C)	5	0.17%
Type BC		16	0.53%
Type ABC		188	6.24%
	(A1BC)	145	4.82%
	(A2BC)	34	1.13%
	(A1A2BC)	9	0.30%
total		3,011	100.00%

* 유형 분류 기준 A에 해당하는 시스템 중, 웹 기반 시스템에서 프로그램 수정이 필요한 경우는 A1, C/S 시스템에서 프로그램 수정이 필요한 경우는 A2로 표기하였다.

** 간혹 웹과 C/S 방식이 같이 적용된 시스템이 있는데 그 경우 둘 다 수정이 필요한 때에는 A1A2로 표기하였다.



위의 결과를 보면 상당한 비율을 차지하는 것이 AB 유형(78.28%)인 것을 알 수 있다. 즉, AB 유형은 데이터가 입출력되는 응용프로그램 부분과 DB에 저장되는 부분을 수정해야 하는 유형이다. 이는 다시 웹 기반 시스템인 A1B 유형(66.16%)과 C/S 시스템인 A2B 유형(10.83%)으로 구분할 수 있으며, 웹 기반인 A1B 유형이 대부분을 차지하고 있다는 것을 확인할 수 있다.

Ⅲ. 해결방안

1. 문자 인코딩 대안 조사/분석
2. 문자 인코딩 변환 모듈
3. 단일시스템 해결 방안
4. 시스템 연계 해결 방안

앞서 살펴본 바와 같이, EUC-KR로 구현된 시스템에서 표현할 수 없는 글자들을 표현하기 위해서는 시스템을 개선할 필요가 있다. EUC-KR의 대안을 조사/분석하고, 이를 시스템에 적용하여 확장한글 문제를 해결할 수 있는 방안들을 알아본다. 이와 더불어 EUC-KR을 개선하여 확장한글을 표현할 수 있도록 지원하는 변환 모듈의 구성과 기능에 대해서도 다루어본다.

1. 문자 인코딩 대안 조사/분석

기존의 EUC-KR 인코딩 방식에서 처리하는 2,350자의 글자 외의 한글을 사용하기 위한 방법으로는 다음과 같은 방법들이 있다. 아래의 예시에서 2,350자 범위 밖의 글자인 '샲'을 각 방식에서 어떻게 처리하는지 보여주고 있다.

가. UTF-8 인코딩 / 유니코드

UTF-8 인코딩은 유니코드의 가변 길이 문자 인코딩 방식으로, 모든 유니코드 문자를 표현할 수 있어서 널리 쓰이는 문자 처리 방식이다. 유니코드에서 한글 표현의 경우 KS X 1026-1으로 표준화되어 있고, 기술표준원 산하 문자코드위원회에서 이를 권고하고 있다.

UTF-8	더			공백	샲			공백	A	P	T	공백	1	0	2	동		
코드 값	EB	8D	94	20	EC	83	BE	20	41	50	54	20	31	30	32	EB	8F	99

1) 특징

가) 문자의 종류에 따라 1~4byte의 용량을 차지한다. (한글은 3byte 차지)
나) ASCII 문자셋 및 인코딩과 호환 가능하며 바이트 단위의 처리가 용이하다.

2) 장점

가) 모든 유니코드(UCS) 글자를 처리할 수 있다.
나) 실질적인 국제 표준으로서, 다중 플랫폼, 언어 및 국가 간에 통용할 수 있는 인코딩 방법이다.
다) 현대 한글 11,172자가 가나다 순으로 순차적으로 배열되어 정렬과 검색이 용이하다.

3) 단점

가) EUC-KR 인코딩과 호환되지 않아, 기존 DB 데이터를 전면적으로 변환해야 한다.

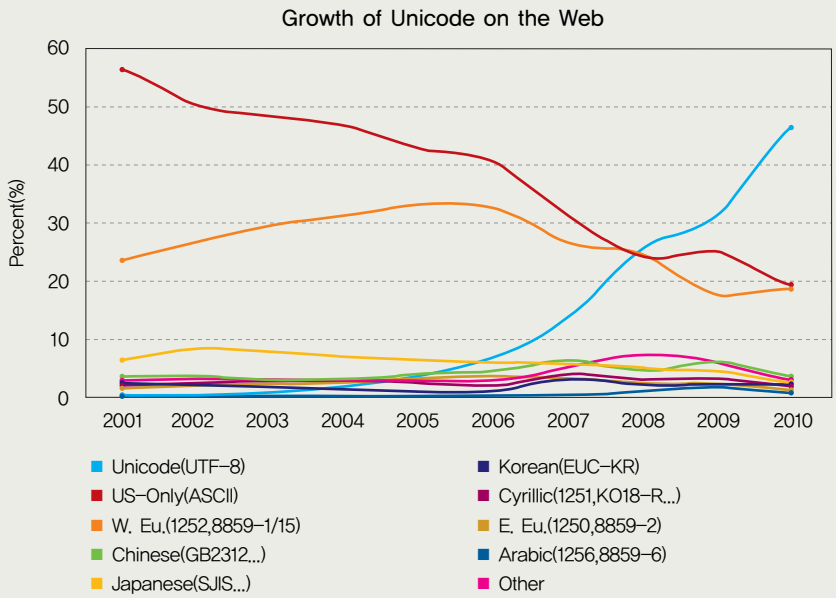
나) 한글의 경우 글자당 3byte 씩의 용량을 차지하므로, 글자당 2byte를 차지하는 EUC-KR에 비해 데이터의 용량이 커질 수 있다. 단, 한글 문자열 데이터의 전체 규모를 고려할 때 장비 추가 구매가 필요한 경우는 많지 않을 것으로 보인다.

예 레코드당 한글 데이터가 2,000자 포함된 데이터가 100만 건 있는 경우 증가용량
약 2GB(4GB→6GB) / 서버용 하드디스크 최소 단위 용량 146GB

참 고

유니코드(Unicode)

최근 십여 년 간 대부분의 H/W, S/W에서 지원하는 코드 시스템으로, 현재까지 사용하던 대부분의 문자셋을 대치하는 중이며, 향후 장기간 사용 가능한 코드 시스템이다.
현대 한글 11,172자가 순서대로 배열되어 있어, 현대 한글 처리에 가장 효율적이다.
아래 그래프는 유니코드의 인코딩 방식인 UTF-8이 2008년 이후 다른 모든 인코딩 방식에 비해 웹에서의 사용 비율이 높아지고 있음을 보여주고 있다.



* 출처 : Unicode nearing 50% of the web – The Official Google Blog(posted by Mark Davis, Senior International Software Architect) – 2010년 1월

참 고

KS X 1026-1

정식 명칭은 '정보 교환용 한글 처리 지침'이며, 유니코드 한글 처리의 정규화에 대한 규격이다.
유니코드를 이용하여 한글을 표기할 때, 한글 하나를 표기하는 데에 여러 가지 방법을 사용할 수 있다. 실제로는 같은 글자인데도 글자를 표기하는 방법이 여러 가지라면 데이터를 검색하거나 비교할 때에 문제가 발생할 수 있다.
이에 따라 같은 글자를 표기하기 위한 방법을 단일화하여 정보처리 시에 문제가 발생하지 않도록 하고, 표준화를 통해 시스템 간 효율적이고 올바른 데이터 교환을 이루도록 하기 위해 KS X 1026-1이 마련되었다. (2007년 12월)

예를 들어, 한글 글자마디 '가'의 경우,

U+AC00 : 완성자 '가'
U+1100, U+1161 : 한글 자모 'ㄱ'과 'ㅏ'의 조합

두 가지로 나타낼 수 있는데, KS X 1026-1에서는 U+AC00만 쓸 것을 제시하고 있다.

나. EUC-KR (+NCR) 방식

EUC-KR 인코딩으로 표현 가능한 2,350자는 기존 방식으로 표시하고, 그 범위를 벗어난 글자에 대해서는 NCR (Numeric Character Reference) 방법으로 표시하는 방식이다. '샲'은 '샾' 또는 '&#xC0FE'로 표시되는데, NCR 방법은 '&'와 '#' 기호 뒤에 해당 글자의 유니코드 코드 포인트를 10진수 또는 16진수로 적고 ';' 기호로 닫는 방식이다.

NCR (10진수)	더		공백	&	#	4	9	4	0	6	;	공백	A	P	T	공백	1	0	2	동	
코드 값	B4	F5	20	26	23	34	39	34	30	36	3B	20	41	50	54	20	31	30	32	B5	BF

또는

NCR (16진수)	더		공백	&	#	x	C	0	F	E	;	공백	A	P	T	공백	1	0	2	동	
코드 값	B4	F5	20	26	23	78	43	30	46	45	3B	20	41	50	54	20	31	30	32	B5	BF

1) 특징

가) 확장한글은 8byte를 차지하고, 일반 한글은 EUC-KR에서와 같이 2byte를 차지한다.[2]

나) EUC-KR (+NCR) 방식에서 글자 하나당 byte 수는 다음과 같다.

문자 종류	숫자	알파벳	한글	확장한글	한자	확장한자
byte 수	1	1	2	8	2	8~10

2) 장점

가) HTML, XML과 같은 마크업 언어(Markup Language)를 표현하는 웹 브라우저에서는 원래의 유니코드 글자로 복원하여 출력해준다.

나) 기존 EUC-KR 인코딩과 같이 사용할 수 있어, 데이터 송수신 때나 DB 저장 시 데이터를 온전히 전달할 수 있다. (EUC-KR 범위 안의 문자로 표현)

다) 유니코드 값을 사용하므로 한글 외의 다른 문자도 모두 나타낼 수 있다. (확장한자 등에도 대응)

3) 단점

가) 웹 브라우저 이외의 경우에는, 별도의 변환 과정을 거쳐야 해당 글자를 복원하거나 출력할 수 있다.

나) 복원하지 않은 상태로는 어떤 글자인지 알아볼 수 없다. 그대로 화면에 표시할 경우, 기호와 문자 번호가 드러나게 된다.

다) 웹 브라우저의 경우에도 입력 시에는 별도의 처리가 필요하다.

2) 유니코드에서 현대한글 코드의 범위 '가'에서부터 '힣'까지의 10진수의 값은 44032에서 55203으로, 다섯 자리 숫자이다.

다. EUC-KR KS : EUC-KR + KS채움쪽자 방식 적용

확장한글에 대해 채움 쪽자 방식을 적용하여 KS 표준을 정확히 구현한 것으로, KS 완성자 2,350자는 기존 방식으로 표시하고, 확장한글은 채움코드(0xA4D4) 뒤에 초/중/종 낱자를 붙여 표시하는 방법이다.

"더 샲 APT 102동" 이라는 문자열을 이 방식으로 쓰면 다음과 같이 된다.

EUC-KR KS	더		공백	□ 채움		ㅅ		ㅑ		ㅍ		공백	A	P	T	공백	1	0	2	동	
코드 값	B4	F5	20	A4	D4	A4	B5	A4	C1	A4	BD	20	41	50	54	20	31	30	32	B5	BF

1) 특징

가) 확장한글은 8 바이트를 차지하고, 일반 한글은 EUC-KR에서와 같이 2 바이트를 차지한다.

2) 장점

가) KS X 1001 규격을 준수하는 표준적 방법이다.

나) 기존 EUC-KR 인코딩과 호환성이 있어, 데이터 송수신 때나 DB 저장 시 데이터를 온전히 전달할 수 있다.

다) 별도의 변환 작업을 하여 완성자 처리를 하지 않더라도 출력 시 어떤 글자인지 확인할 수 있다. (가독성)

3) 단점

가) 입출력 시 별도의 변환 과정이 필요하다.

나) 한글 이외에 한자 등 다른 종류의 문자 확장에는 사용할 수 없다.

라. MS949 (Codepage 949)

KS X 1001에서 지정한 영역 외에 확장한글을 배치하여 사용하는 방식이다. MS 자체적으로 확장하여 만든 방식으로서 확장완성형 또는 통합완성형이라고 불리기도 한다.

MS949	더		공백	샲		공백	A	P	T	공백	1	0	2	동	
코드 값	B4	F5	20	98	DE	20	41	50	54	20	31	30	32	B5	BF

1) 특징

가) 현대 한글을 2byte로 처리한다.

2) 장점

가) EUC-KR을 확장하였으므로 기존 EUC-KR과 호환성이 있다.
나) 확장한글에 대해서도 추가 데이터 용량을 차지하지 않는다. (2byte)

3) 단점

가) 한글이 가나다 순으로 순차적으로 배치되어 있지 않아, 정렬 시 별도의 작업이 필요하다.
나) Windows 용으로 개발된 비표준 인코딩 방식으로 타 OS에서는 호환성 문제가 있다.

예 글로벌 리눅스 배포판인 Ubuntu 8.10에서 압축파일(*.zip)을 해제할 때, 그 압축파일 속에 한글(MS949)로 된 파일이름이 있으면 글자가 깨져서 나타난다.

참 고

문자 인코딩 해결 방안별 장단점 비교

방안	장점	단점
UTF-8 인코딩	• 모든 UCS 글자 수용 • 다국어 처리 가능	• 기존 DB의 전면적 변환 필요
EUC-KR (+NCR)	• 웹브라우저 운영 시 효율적 대안 • 유니코드 표현 가능	• 입력 시 별도 처리 필요 • C/S 환경에서 변환 필요
EUC-KR + KS채움쪽자	• KS 규격을 만족하는 표준적 방법 • 비교적 높은 가독성	• 입출력 시 변환 필요 • 확장한자 미지원
MS949 (CodePage 949)	• 한국어 Windows에서 통용 • 비교적 적은 데이터 용량	• Windows 외 시스템 대비 안됨 • 타 시스템 연계 시 깨질 수 있음

2. 문자 인코딩 변환 모듈

가. 개요

확장한글 변환 모듈은 확장한글을 포함한 문자열을 UTF-8 문자열과 서로 변환하기 위해 사용한다. 확장한글이 포함된 문자열을 저장하거나 다른 시스템으로 전달할 때 확장한글이 깨지지 않고 유지되도록 하는 기능을 한다.

나. 변환 모듈의 구성

변환 모듈은 Java와 C++ 소스 코드 형태이다. 특정 시스템에 이 모듈을 적용할 때에 각각의 환경에 맞게 수정하거나 최적화하여 사용할 수 있도록 소스 코드 상태로 제공한다.
변환 모듈은 다음과 같은 세 가지의 함수로 구성된다.

1) EUC-KR류[3] → UTF-8로 변환 함수
2) UTF-8 → EUC-KR류로 변환 함수
3) 지정한 문자 인코딩 범위 밖의 글자가 있는지 확인하는 함수

다. 변환 모듈의 기능

변환 모듈에 포함된 세 함수는 아래와 같은 기능을 한다. (Java 소스코드를 기준으로 설명)

3) EUC-KR류 : EUC-KR과, 그것에 기반을 둔 EUC-KR KS, MS949 및 EUC-KR (+NCR)을 통틀어 이르는 용어이다.

1) EUC-KR류 → UTF-8로의 변환 함수

```
public String
convertEuckrToUTF8
(byte[ ] euckrStream, CharsetType charType, boolean decodeNCR);
```

주어진 EUC-KR류의 입력 스트림(즉, EUC-KR 또는 MS949로 인코딩된 문자열의 바이트 스트림)을 유니코드(UTF-8) 문자열로 변환하는 메소드이다.
KS채움쪽자 방식으로 기록된 한글은 유니코드 한글로 변환하고, NCR 방식으로 기록된 글자는 decodeNCR 파라미터 값에 따라 유니코드 한글로 변환하거나(true일 때) 그대로 둔다(false일 때).

converEuckrToUTF8( ) 함수의 입력 파라미터와 출력

구분	입력 파라미터	내용
입력	euckrStream	변환하려는 입력 스트림
입력	charType	주어진 입력 스트림의 문자 집합 (EUC-KR 또는 MS949)
입력	decodeNCR	주어진 입력 스트림 내 NCR 형식의 문자가 있는 경우, 일반 유니코드 문자로 변환할지 여부
출력		변환된 유니코드(UTF-8) 문자열

converEuckrToUTF8( ) 함수의 출력 결과 예시

euckrStream (EUC-KR류 문자열)	더 ㅁㅅㅑㅍ 아파트 옆에 있는 헤어 샾
⇩	

charType	decodeNCR	변환 결과 (UTF-8 문자열)	비고
EUC-KR	false	더 샲 아파트 옆에 있는 헤어 샾	
EUC-KR	true	더 샲 아파트 옆에 있는 헤어 샲	
MS949	false	더 샲 아파트 옆에 있는 헤어 샾	EUC-KR과 변환 결과 동일
MS949	true	더 샲 아파트 옆에 있는 헤어 샲	EUC-KR (+NCR)과 변환 결과 동일

* UTF-8로 결과 저장시 모든 KS채움쪽자는 UCS 완성자로 변환된다. ('샲')

2) UTF-8 → EUC-KR류로 변환 함수

```
public byte[ ]
convertUTF8ToEuckr
(String unicodeStr, CharsetType charType, EncodeMethod encodeMethod);
```

주어진 유니코드(UTF-8) 문자열을 EUC-KR류의 스트림(즉, EUC-KR 또는 MS949로 인코딩된 문자열의 바이트 스트림)으로 변환하는 함수이다. encodeMethod에 따라 확장한글과 확장한자를 처리하는 방법이 달라진다.

converEuckrToUTF8( ) 함수의 입력 파라미터와 출력

입출력 구분	입력 파라미터	내용
입력	unicodeStr	변환하려는 유니코드 문자열
입력	charType	결과 스트림의 문자 집합 (EUC-KR 또는 MS949)
입력	encodeMethod	입력 문자열 내에, 지정된 문자 집합 범위 외의 문자가 포함된 경우 이를 어떤 방식으로 처리할지 지정하는 enum 타입의 값. • ALL_NCR : 깨지는 글자를 모두 NCR 방식으로 변환 • NCR_EUCKR : 깨지는 글자 중 한글은 KS채움쪽자 방식으로, 나머지 글자는 NCR 방식으로 변환 • EUCKR : 깨지는 글자 중 한글은 KS채움쪽자 방식으로, 나머지 글자는 무시 • IGNORE : 깨지는 모든 글자를 무시.
출력		변환된 EUC-KR류의 문자열 스트림

converEuckrToUTF8( ) 함수의 출력 결과 예시

unicodeString (UTF-8 문자열)	일본식 한자 '会'은 '會'와 같다. '뒃'은 EUC-KR 완성자가 아니다.	
⇩		
charType	encode-Method	변환 결과
EUC-KR	ALL_NCR	일본식 한자 '会'은 '會'와 같다. '텇'은 EUC-KR 완성자가 아니다.
EUC-KR	NCR_EUCKR	일본식 한자 '会'은 '會'와 같다. '□ㅌㅓㅊ'은 EUC-KR 완성자가 아니다.
EUC-KR	EUCKR	일본식 한자 '?'은 '會'와 같다. '□ㅌㅓㅊ'은 EUC-KR 완성자가 아니다.
EUC-KR	IGNORE	일본식 한자 '?'은 '會'와 같다. '?'은 EUC-KR 완성자가 아니다.
MS949	ALL_NCR	일본식 한자 '会'은 '會'와 같다. '뒃'은 EUC-KR 완성자가 아니다.	'뒃'은 MS949 코드
MS949	NCR_EUCKR	일본식 한자 '会'은 '會'와 같다. '뒃'은 EUC-KR 완성자가 아니다.	'뒃'은 MS949 코드
MS949	EUCKR	일본식 한자 '?'은 '會'와 같다. '뒃'은 EUC-KR 완성자가 아니다.	'뒃'은 MS949 코드
MS949	IGNORE	일본식 한자 '?'은 '會'와 같다. '뒃'은 EUC-KR 완성자가 아니다.	'뒃'은 MS949 코드

3) 지정한 문자 인코딩 범위 밖의 글자가 있는지 확인하는 함수

```
public boolean
detectCharOutOfRange
(String unicodeStr, CharsetType charType);
```

주어진 유니코드(UTF-8) 문자열 내에, 지정한 문자 집합 범위 외의 문자가 포함되어 있는지 확인하는 함수이다. 즉, 입력 문자열을 그 문자 인코딩으로 인코딩할 때 깨지는 글자가 포함되어 있으면 true, 그렇지 않으면 false가 출력된다.

detectCharOutOfRange( ) 함수의 입력 피라미터와 출력

입출력 구분	입력 파라미터	내용
입력	unicodeStr	확인하려는 UTF-8 문자열
입력	charType	목표 문자 집합
출력		• true : 깨질 문자가 포함되어 있을 때 • false : 깨질 문자가 없을 때

detectCharOutOfRange( ) 함수의 출력 결과 예시

unicodeStr \ charType	EUC-KR	MS949
'𠮷'은 일본식 한자이다.	true	true
'덏'은 EUC-KR에는 없다.	true	false

* 범위 밖의 글자가 있는 경우에 출력 값이 true.

라. 변환 모듈의 용도

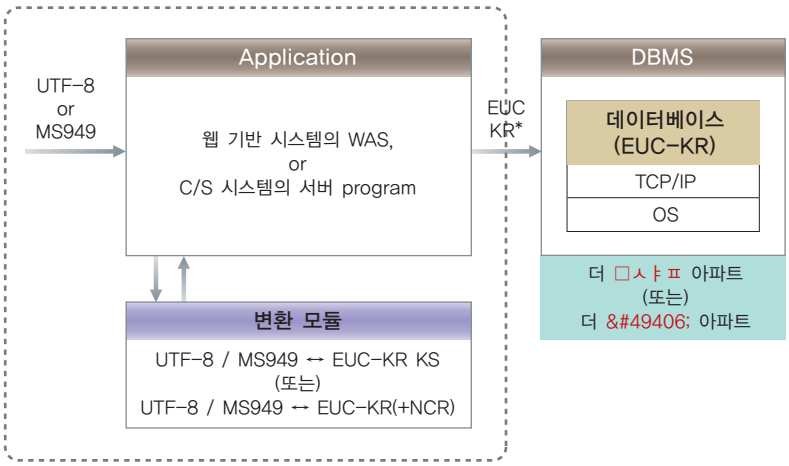
기존 시스템에서 사용하고 있는 인코딩으로 입출력 데이터를 상호 변환한다.

1) 용도 1 : 입력되는 데이터를 변환할 때

입력되는 데이터의 인코딩과 프로그램 단계에서의 인코딩이 상이할 경우, 정상적인 데이터 처리를 위해 입력 단계에서 변환 모듈을 활용하여 확장한글 문제가 발생하지 않도록 할 수 있다.

가) 기존 응용프로그램이 EUC-KR 인코딩으로 동작하는 경우, 확장한글이 포함된 데이터가 입력된다면 응용프로그램 처리 과정에서 해당 문자가 소실될 수 있다. 이럴 경우 convertUTF8ToEuckr() 함수를 통해 EUC-KR과 호환하는 방법으로 확장한글을 변환하여, EUC-KR을 사용하는 응용프로그램에서 데이터를 그대로 처리할 수 있게 한다.

변환 모듈의 용도 1 : 입력 데이터 변환



2) 용도 2 : DB에 데이터를 저장하거나 읽을 때

EUC-KR 인코딩을 사용해온 DB의 기존 데이터와 호환성을 유지하면서, 새로 입력되는 데이터에 대해서는 확장한글 문제가 발생하지 않도록 할 수 있다.

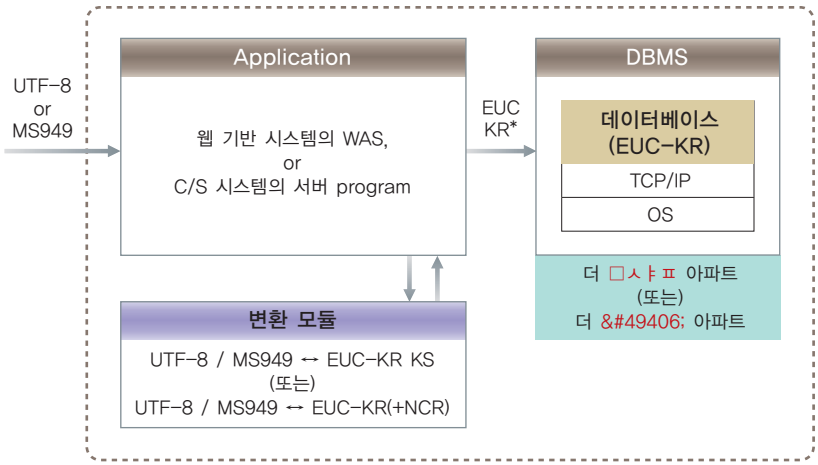
가) 기존 DB가 EUC-KR로 설정되어 있을 경우, 확장한글이 포함된 데이터를 DB에 저장하면 확장한글은 손상되어 DB에 저장되지 않는다. 이럴 경우, DB 저장 직전에 convert UTF8ToEuckr() 함수를 통해 EUC-KR과 호환하는 방법으로 확장한글을 변환하여, EUC-KR을 사용하는 DB에 데이터를 저장할 수 있다.

나) 응용프로그램에서 DB로 데이터가 들어가는 경로는 기존의 EUC-KR 인코딩 데이터가 전달되던 통로인데, 변환 모듈을 통해 만들어진 EUC-KR KS 데이터나 EUC-KR (+NCR) 데이터도 EUC-KR에서 사용하는 코드값을 쓰기 때문에 같은 경로로 DB에 입력될 수 있다.

다) DB에서 데이터를 출력할 때도 변환 모듈의 convertEuckrToUTF8() 함수를 이용

하여, KS채움쪽자(또는 NCR) 방식으로 저장되었던 확장한글을 복원해서 출력할 수 있다.

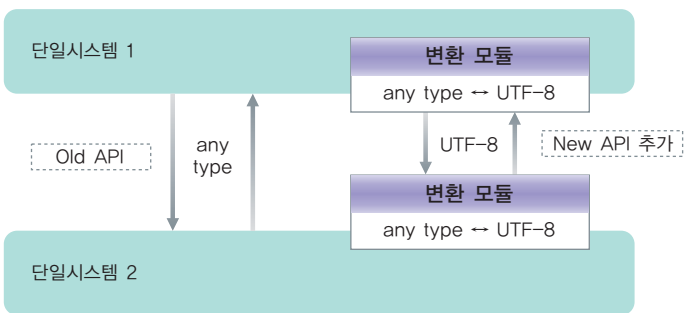
변환 모듈의 용도 2 : DB 저장 데이터의 변환



3) 용도 3 : 연계된 다른 시스템과 데이터를 주고받을 때

내부에서 어떤 타입의 인코딩을 사용하든지 외부의 시스템과 연계될 때는 언제나 UTF-8로 송수신해야 한다. 이때 변환 모듈을 사용하면 확장한글을 유지하면서 데이터를 송수신할 수 있다.

변환 모듈의 용도 3 : 연계 데이터 변환



가) 연계된 다른 단일시스템과 데이터를 주고받을 때에는 UTF-8을 사용하고, 현재의 단일시스템 내부에서는 EUC-KR류의 인코딩을 사용하는 상황이라면, 데이터를 외부에서

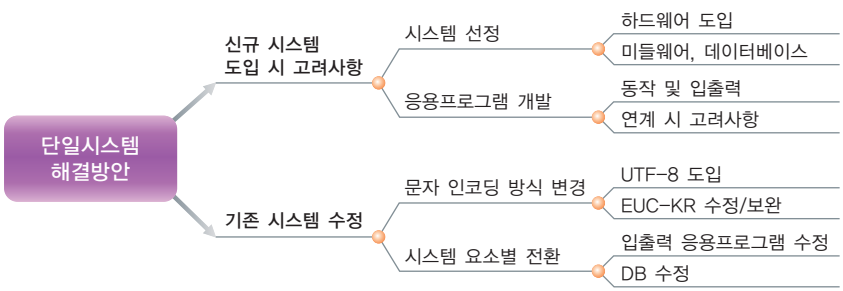
받은 직후에, 변환 모듈의 convertUTF8ToEuckr() 함수를 이용하여 EUC-KR류의 문자열로 변환하면 운영 중인 기존 시스템에 적용할 수 있다.

나) 위의 경우에서 반대로 상대방 시스템에 데이터를 보내는 경우, 내부적으로 사용하던 EUC-KR류 방식의 확장한글 데이터를 유니코드(UTF-8) 문자열 형태로 외부에 보내주어야 한다. 이때 변환 모듈의 convertEuckrToUTF8() 함수를 이용하면 정상적으로 UTF-8 문자열로 바꾸어 전송할 수 있게 된다.

3. 단일시스템 해결 방안

가. 해결 방안 개요

정보시스템에서 확장한글 문제를 해결하려고 할 때에 취할 수 있는 방안들이다. 모든 현대 한글을 표현할 수 있고, 여러 플랫폼과 언어 환경을 지원할 수 있는 유니코드 기반 시스템으로 개편하는 것이 가장 바람직하다. 즉, 시스템의 모든 부분에서 유니코드 데이터를 처리하고 UTF-8 인코딩을 사용하는 것이 가장 바람직한 표준 모델이라고 할 수 있다.그러나 전면적으로 개편할 경우 DB 데이터 이전과 프로그램 수정에 드는 비용과 시간이 상당하기 때문에, 부분적으로 수정하여 확장한글 문제를 해결하는 방법을 추구할 수 있다. 단, 이때에도 어떤 형태로든 원 데이터가 손상되지 않고 저장될 수 있는 문자 인코딩 방식을 적용하여 최소한 현대한글 11,172자를 모두 표현할 수 있는 시스템으로 만들어야 하며, 그래야만 데이터가 다른 시스템과 연계되어 유통되는 과정에서도 손상되지 않을 수 있다.



나. 신규 시스템 도입

1) 시스템 선정 시 고려 사항

가) 하드웨어 도입

하드웨어 도입 시 OS 또는 펌웨어는 유니코드 기반 시스템이어야 한다.

나) 미들웨어(WAS), 데이터베이스

미들웨어 및 데이터베이스는 유니코드(또는 UTF-8)를 지원해야 하며, 설치 시 유니코드(또는 UTF-8)로 기본 설정해야 한다.

2) 응용프로그램 개발 시 고려 사항

가) 동작 및 입출력

응용프로그램 입출력 시 유니코드 기반으로 개발한다.
웹 기반 응용프로그램의 경우 UTF-8 인코딩을 적용해야 한다.

나) 연계 고려 사항

타 시스템과 연계 시 상대 시스템과 UTF-8 데이터를 송수신할 수 있도록 API 마련

다. 기존 시스템 수정

1) 문자 인코딩 방식 변경

가) UTF-8 도입

(1) 시스템의 모든 부분에서 UTF-8을 사용하도록 하여 확장한글 문제를 해결할 수 있다.
(2) 기존 시스템의 경우 EUC-KR로 인코딩한 데이터를 DB에 가지고 있다면 UTF-8 인코딩으로 적용할 경우 데이터 마이그레이션이 필요하다.

나) EUC-KR 수정/보완

시스템을 모두 UTF-8로 전환하기 어려운 경우, EUC-KR을 확장한 EUC-KR류의 인코딩 방식을 사용할 수 있다. (단, 이 경우에도 시스템의 데이터 입출력 부분에는 UTF-8을 도입해야 한다.)

(1) EUC-KR 인코딩을 확장하여 확장한글을 담을 수 있는 EUC-KR류의 방법으로는 EUC-KR (+NCR), EUC-KR KS, MS949 등이 있고, 기존 EUC-KR 데이터와 호환되므로 단기적인 대안으로 사용할 수 있다.

(2) EUC-KR KS와 MS949의 경우 확장한자에 대해 NCR 표기 방법을 병행 사용할 수 있다.

(3) EUC-KR류의 인코딩을 사용하기 위해서는 변환모듈을 각 시스템의 필요한 부분에 적용해야 한다.

2) 시스템 요소별 전환

가) 입출력 응용프로그램 수정

웹 기반 시스템에서는 웹 브라우저를 통해, C/S 시스템에서는 별도의 클라이언트 프로그램을 통해서 사용자가 데이터를 입력하게 된다. 입력 단계에서부터 데이터가 손상되지 않도록 해야 한다.

(1) 웹 기반 시스템 :

(가) 사용자가 데이터를 입력하는 웹 페이지가 EUC-KR로 설정되어 있는 경우 확장한글 데이터가 서버로 전달될 때 손상된다.

(나) 손상 없이 전달되려면 웹 페이지 설정이 UTF-8 이어야 한다. 즉, 웹 페이지의 시작 부분에 메타 태그 내용이 UTF-8로 설정되어야 한다.〈meta http-equiv="content-type" content="text/html;charset=UTF-8" /〉

(다) 서버 프로그램의 웹 페이지 생성 부분을 수정하여 위와 같이 웹 페이지의 메타 태그 내용이 표시될 수 있도록 해야 하고, 입력된 UTF-8 데이터를 처리할 수 있도록 수정해야 된다.

(2) C/S 시스템 :

(가) 클라이언트 프로그램 작성 도구에 따라 확장한글 구현 능력에 차이가 날 수 있다.

(나) 일반적인 경우 클라이언트에서 데이터를 가공하여 서버로 전송하고, 그대로 DB에 저장하는 경우가 많다. 사용자가 입력한 데이터가 EUC-KR에 의존할 경우 입출력에 제한이 있을 수 있다.

(다) 확장한글 데이터가 깨지지 않게 하려면 클라이언트 프로그램을 유니코드 기반으로 재작성해야 한다.

나) DB 수정

데이터가 실제 저장되는 곳이 DB이므로, DB가 확장한글을 지원할 수 있는지가 중요하다.

(1) EUC-KR류의 인코딩 유지

EUC-KR 인코딩을 사용하던 DB를 한시적으로 그대로 유지하려 하는 경우, EUC-KR류의 인코딩(EUC-KR + NCR, EUC-KR KS, MS949)을 적용하여 새로 저장되는 데이터부터는 확장한글이 지원되도록 하는 방안을 사용할 수 있다.

(2) UTF-8 인코딩 도입

UTF-8 인코딩을 DB에 도입하면, 향후에도 문자 인코딩과 관련된 문제는 발생하지 않을 것으로 기대할 수 있다. 단, 기존 데이터의 마이그레이션(migration) 작업이 필요하다. DB에서 사용하는 문자 인코딩 방식을 변경하는 것은, 새로 입력되는 데이터뿐만 아니라 기존에 DB에 들어있던 데이터들의 문자 인코딩 방식 또한 변경하는 것을 뜻하므로, 기존 데이터를 전부 이전하는 과정이 필요하다.

(가) EUC-KR 또는 MS949로부터 UTF-8로 전환

문자 인코딩 방식으로 EUC-KR이나 MS949를 사용하던 DB의 데이터는 UTF-8과 호환되지 않으므로, 모든 데이터를 새롭게 인코딩하여 이전해야 한다. 이 과정에서 기존 인코딩 방식에서 글자당 2바이트를 차지하던 한글 데이터가 UTF-8에서는 글자당 3바이트씩을 차지하게 되므로 DB에서 데이터가 차지하는 용량은 1.5배까지 늘어날 수 있다.

(나) EUC-KR로부터 MS949로 전환

EUC-KR 인코딩을 사용하던 DB를 MS949로 전환하려 할 때에는, MS949 문자 인코딩이 EUC-KR과 호환되므로 경우에 따라 DB의 설정만 변경하는 것으로 이전 과정이 끝날 수도 있다. 단, 이때 DBMS의 종류와 버전에 따라 MS949를 지원하지 않을 수도 있고, 설정 변경만이 아니라 데이터를 다시 이전해야 하는 경우도 있을 수 있으므로 정확한 마이그레이션 방법에 대해서는 현재 사용하는 DBMS의 기술 지원 정보를 참고하도록 해야 한다.

참 고

DB 마이그레이션 절차

다른 문자 인코딩 방식을 사용하도록 DB를 마이그레이션하는 것은 주의 깊게 수행해야 할 중요한 작업이다. 마이그레이션 과정에서 수행하는 절차에는 다음과 같은 것들이 있다. (Oracle의 예)

1. 데이터 백업
2. 깨질 데이터가 있는지 확인 : Database Character Set Scanner 이용
3. 데이터 필드 크기 조정 : 새 인코딩 사용 시 필요한 크기로 늘림
4. 데이터 이전 공간 확보 : 충분한 디스크 공간 확인
5. 예행연습 : 문제 상황 체크와 실제 다운시간(downtime) 줄이기 위함
6. 데이터 이전 : CSALTER 스크립트 또는 Export/Import 유틸리티 이용
7. 데이터 확인

참고 :

- Oracle Globalization Support
  http://www.oracle.com/technetwork/database/ features/globalization
- Character Set Migration Best Practices for Oracle Database 10g
  http://www.oracle.com/technetwork/database/features/globalization/twp-character-set-migration-best-pr-128766.pdf
- Character Set Migration Best Practices for Oracle9i
  http://www.oracle.com/technetwork/database/features/globalization/mwp-129407.pdf

(3) 데이터 클렌징

기존 인코딩 방식을 사용하고 있던 DB가 새로운 문자 인코딩 방식을 적용하거나 DB 이전을 하고자 할 때, 기존 인코딩 방식으로는 표기할 수 없어 이미 손상된 데이터(예를 들어 '샾'이 '?'로 바뀌어 저장된 것) 또는 달리 표현하였던 데이터는 클렌징(cleansing) 과정을 거쳐 원래대로 되살려야 한다.이때 데이터에 손상이 있는 경우는 사실상 원본 확인이 불가능하기 때문에 일괄 수정은 어렵고 개별 작업이 필요할 수 있다. 그러나 다르게 표현된 데이터의 경우는 그 처리방식에 따라 일괄 수정이 가능하다.

(가) 데이터 유형 파악

먼저 손상되거나 다르게 표현된 글자가 어떤 형태를 띠고 있는지 알아야 한다. 다음과 같은 유형들이 있을 수 있다.

① **글자 깨짐**

- 해당 문자 인코딩 범위 밖의 글자 입력으로, 실제 데이터가 저장되지 않고 손상된 경우

  예 물음표(?), 공백( ), 사각형(□)

- 내용상 물음표('?')가 있을 자리가 아닌 곳에 있는 물음표 : 사람 이름, 주소 등의 데이터에 포함된 물음표 (반각 또는 전각 물음표 – KS코드 A3BF)

  예 더 ? 아파트

② **유사 글자 처리**

- 기존 인코딩 방식에서 표현되지 않는 문자를 그와 유사한 표현 가능한 문자로 대체 입력한 경우

  예 더 샵 아파트

③ **임시 처리** : 신규 인코딩 방식과 다른 방법으로 처리된 확장한글 표현 방식

- 기존의 EUC-KR 인코딩에서 확장한글 표현을 위해 KS채움쪽자 방식을 적용한 경우

  예 더 □ㅅㅑㅍ 아파트

- 기존의 EUC-KR 인코딩에서 확장한글 표현을 위해 NCR 방식을 적용한 경우

  예 더 샾 아파트

(나) 데이터 클렌징

손상된 글자를 복원하려 할 때에 어떤 방법으로 작업을 할 것인가를 고려할 필요가 있다.

① **손상된 데이터를 수정**

- 특정 패턴을 파악하여 일괄 수정하는 방안DB에 들어있는 데이터의 종류가 대체로 일정하고, 그에 따라 손상된 데이터의 패턴도 충분히 예측 가능하고 예외가 없을 시, 일괄 수정이 가능하다.
- 문제 사항 발견 시 별도 수정하는 방안클렌징 작업을 위해 별도의 계획과 일정을 할당하지 않고, 시스템 사용 중에 손상된 데이터가 발견되면 그때그때 처리하는 방식이다.예를 들어 주소를 열람할 때에 손상된 글자가 주소에 포함되어 있으면, 운영자가 확인하여 올바른 글자로 다시 수정/입력하는 방식이다.

② 다르게 표현된 데이터를 수정

- 기존 시스템에서 임시 처리한 확장한글 처리 방식을 신규 시스템의 인코딩 방식에 맞도록 변환 모듈을 사용하여 일괄 처리를 한다.
- EUC-KR KS 또는 EUC-KR (+NCR)로 임시 처리된 경우, 시스템에서 제공하는 변환 방식으로는 처리할 수 없기에, 이를 지원하는 별도의 변환모듈을 사용하여야 한다.

(다) 참고

① 수정 내역 보관

- 수정된 내용에 대한 로그(log)를 저장/보관하여, 수정 오류 발생 시 확인하여 되돌릴 수 있도록 한다.

② 수정의 권한

- 원본을 통해 데이터를 작성/입력하는 담당자가 해당 오류 사항 발견 시, 관리 책임자의 확인을 거쳐 데이터를 수정한다.

4. 시스템 연계 해결 방안

가. 해결 방안 개요

단일시스템들이 자체적으로는 확장한글 문제를 해결한다고 해도, 시스템 간에 데이터가 호환되지 않으면 문제가 해결된 것이라고 할 수 없다. 한 가지 통일된 방식으로 시스템 사이에 데이터를 송수신하는 것이 중요하다.

나. 연계 문자 인코딩 통일

1) 여러 시스템들 사이에 주고받는 데이터의 형식이 제각각이면 혼란과 유지보수 비용이 커질 수 있으므로, 외부 시스템과 데이터를 교환할 때에는 UTF-8 인코딩으로 통일하는 것이 효과적이다.

2) 외부와 연계 방식을 UTF-8로 한다고 해서, 당장에 단일시스템 내부를 UTF-8로 전부 바꾸어야 하는 것은 아니다. 일단 외부와의 연계 부분에서는 UTF-8로 변환해서 주고받는 것으로 하고, 시스템 내부는 상황에 따라서 순차적으로 수정하는 것으로 하면 된다.

3) 연계 문자 인코딩을 통일해 놓으면, 연계된 다른 시스템들과 무관하게 자체 시스템 변경이나 업그레이드를 할 수 있다. 내부적으로는 어떤 인코딩으로 데이터를 처리한다 해도, 외부와 송수신할 때 정해진 인코딩(유니코드)을 사용하면 된다.

다. 변환 모듈 사용

1) 단일시스템 내부가 모두 유니코드 기반으로 전환되어 있다면, 외부와 연계 시 변환 모듈을 사용할 필요가 없다.

2) 단일시스템 내부에서 여전히 EUC-KR 데이터를 사용하고 있는 상황이라면, 연계된 상대 시스템과 데이터를 교환할 때 변환 모듈을 사용해서 확장한글 데이터를 유지해야 한다.

라. 신규 인터페이스 추가

1) 신규 인터페이스(API)를 구현할 때 기존 인터페이스(API)는 그대로 두고, UTF-8 인코딩을 사용하는 인터페이스를 추가로 구현한다.

2) 신규 인터페이스가 UTF-8 인코딩을 사용하고 있음을 명확히 알 수 있도록 관련 기관 및 개발자에게 고지한다.

3) 기존 인터페이스(API)를 제거하면 그와 연계된 시스템들은 제대로 동작할 수 없게 되므로, 연계된 시스템들이 신규 인터페이스(API)로 모두 전환할 때까지 기존 인터페이스를 유지한다.

Ⅳ. 전환계획

1. 수준별 전환계획
2. 유형별 전환계획

각 기관에서 정보시스템 개선 시 참고할 수 있는 최소/부분적/전면적으로 전환하는 수준별 전환 방법과 응용프로그램/DB 처리 등 각각의 유형별 한글 처리 방법을 모델별로 소개한다.

1. 수준별 전환계획

웹 브라우저를 통한 간단한 형태의 전환에서부터 부분적인 수정을 통한 해결(Web, C/S, 연계 환경 등), 그리고 전면적인 해결에 이르기까지, 일반적인 관점에서의 수준별 전환계획을 소개한다.

구분	수준			내용	비고
	1	2	3		
웹 브라우저를 통한 해결				웹 브라우저에서 지원하는 기능으로 임시 사용	최소
기존 시스템 수정/보완 (웹)				웹 기반 시스템에서의 일부 수정	부분
기존 시스템 수정/보완 (C/S)				C/S 기반 시스템에서의 일부 수정	부분
기존 시스템 수정/보완 (연계)				시스템 간 연계에 있어 일부 수정	부분
전면적 전환				시스템 내 모든 부분의 전면적 수정	최대

가. 웹 브라우저를 통한 해결

정보시스템 중 확장한글 문제가 일어나는 시스템들의 상당수는 웹 기반 시스템이다. 즉, 이 시스템들에서는 웹 브라우저가 주요 데이터 입출력 통로인데, 브라우저가 확장한글을 어떻게 처리하느냐에 따라서 시스템들을 수정하지 않고 그대로 운영할 수도 있다.

1) 브라우저에 EUC-KR (+NCR) 이용 : 모든 브라우저

가) NCR 출력 지원

모든 브라우저는 출력 시 'HTML Character Entity (NCR)'를 지원한다. 즉, EUC-KR 인코딩에서 지원하지 않는 글자인 경우 NCR 타입으로 저장된 글자를 브라우저에서 유니코드로 변환하여 출력을 해준다.

예 NCR 타입 데이터(샾) → 브라우저 출력(샾)

나) 입력 시 수동 입력 필요

NCR 타입은 브라우저에서 출력 시에만 지원하고 있으므로, 입력 시 별도 수작업으로 NCR 코드를 입력하여 확장한글 문자를 유지할 수 있다. 즉, 입력 단계에서 해당 글자의 유니코드 값을 확인하고 NCR 형식에 맞춰서 입력하는 과정이 필요하다.

예 주소 입력 시 "더 샲 아파트"를 입력할 경우,

(1) '샲' 자가 확장한글에 해당하므로, 해당 글자의 유니코드 값을 확인한다.[4]('샲' 자의 유니코드 값은 10진수로 '49406' 이며 16진수는 '0xC0FE' 이다.)

(2) 주소 입력 창에 확장한글을 NCR 형식(&#____;)에 맞추어 입력한다.("더 샲 아파트" 대신 "더 샾 아파트" 또는 "더 샾 아파트"로 입력)

(3) 이렇게 입력된 데이터를 서버로 전송/저장한다.

다) 장단점

(1) 장점

(가) 확장한글 문제와 관련해서 기존의 EUC-KR 시스템들을 수정하지 않고도 사용할 수 있다.

(나) 확장한글 이외에도 모든 유니코드 문자를 이용할 수 있다. (확장한자 등)

(2) 단점

(가) 입력 시 별도 수작업으로 코드 값을 입력해야 한다.

2) 브라우저에 EUC-KR KS 이용 : Firefox 브라우저

가) EUC-KR KS 지원 (KS채움쪽자)

KS X 1001 표준에서 규정한 KS채움쪽자 방식을 브라우저가 이미 지원하고 있다면, 사용자가 입력한 데이터를 EUC-KR로 인코딩해서 전달할 때 확장한글을 포함한 모든

4) 유니코드 값 확인 방법 : HWP에서 해당 글자(샲) 입력 후, 블록으로 잡고, 'Ctrl+F10' 단축키를 넣으면, 문자표 대화상자 안에서 유니코드의 16진수 값을 확인할 수 있다.

현대 한글을 깨지지 않고 전달할 수 있다.

이 데이터는 기존 시스템에서 사용하는 EUC-KR 규약에 부합하기에 Web/WAS/DB 등을 수정하지 않고도 기존 시스템에서 그대로 사용할 수 있다.

예를 들어 Firefox 브라우저는 EUC-KR 인코딩에서 KS채움쪽자 방식을 지원하기에, 사용자가 "더 샾 아파트"를 입력하면 이 문자열을 "더 ?ㅅㅑㅍ 아파트"로 변환하여 서버로 전달한다.

출력 시에도 KS채움쪽자 방식으로 쓰여진 확장한글이 HTML 문서에 포함되어 있다면, 그 글자를 원래의 글자로 복원해서 화면에 출력한다. ('?ㅅㅑㅍ' → '샾')

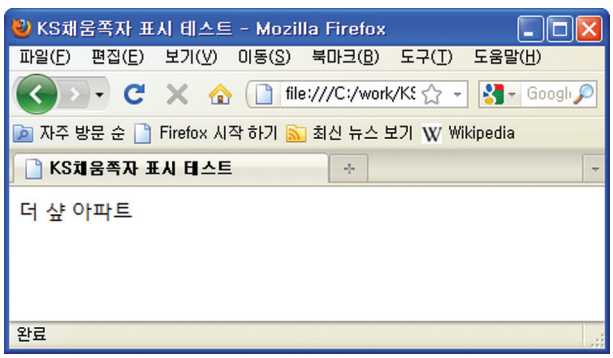
다음 예는 간단한 HTML 파일을 Firefox 브라우저에서 열어본 화면이다.

웹 브라우저의 KS채움쪽자 방식 지원 여부를 확인하기 위한 HTML 문서

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
        <title>KS채움쪽자 표시 테스트</title>
        <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR" />
</head>
<body>
        더 ㅅ ㅑ ㅍ 아파트
</body>
</html>
```

풀어쓴 낱글자들(ㅅ, ㅑ, ㅍ) 앞에는 보이지는 않지만 채움 기호(KS 코드 A4D4)가 들어가 있다.

Firefox 3.6에서의 결과



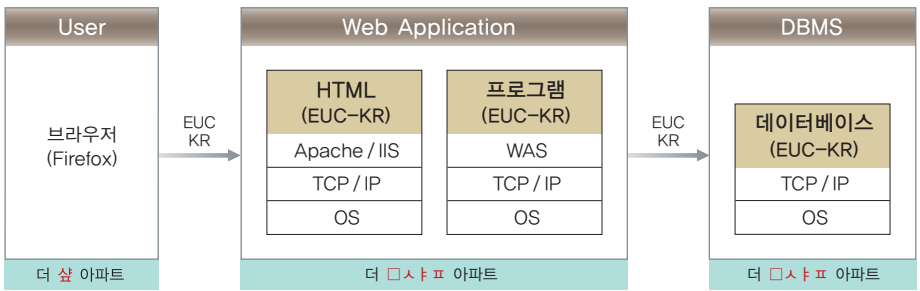
KS채움쪽자 방식의 글자 '샾'이 제대로 표시되었다.

나) 데이터 입출력 방안

브라우저에서 KS채움쪽자 방식의 인코딩/디코딩이 모두 가능하다면, 기존의 시스템들을 수정하지 않고도 문제가 해결될 수 있다.

Firefox의 경우 EUC-KR 인코딩에서 KS채움쪽자 방식을 정확히 구현하고 있어 이와 같이 동작할 수 있다. (단, 웹 접근성 표준을 준수하는 사이트가 아니면 Firefox가 정상 동작하지 않을 수 있다.)

EUC-KR KS 인코딩을 지원하는 브라우저에서 데이터를 입력한 경우



위 그림은 EUC-KR 인코딩 기반의 단일시스템에 브라우저로 데이터를 입력하는 경우의 예이다. 브라우저를 통해 들어온 확장한글 '샾'은 KS채움쪽자 방식으로 다음 단계로 전달된다. 이것은 '□ㅅㅏㅍ'이라는 문자열의 각각의 낱자도 모두 EUC-KR 범위 내의 코드값을 갖기 때문으로, 이 문자열은 기존의 EUC-KR 데이터를 주고받는 모든 인터페이스에서 문제없이 사용 가능하고, EUC-KR로 설정된 DB에서도 그대로 저장 가능하다.

다) 장/단점

(1) 장점

(가) 확장한글 문제와 관련해서 기존의 EUC-KR 시스템을 수정하지 않고도 사용할 수 있다.

(나) 입력과 출력 단계에서 모두 사용할 수 있다.

(2) 단점

(가) 웹 표준을 준수하는 시스템에서는 Firefox를 정상 이용할 수 있으나, 국내의 많은 웹 기반 시스템이 'ActiveX 기술'을 사용하는 등 MS IE 기준으로 만들어져

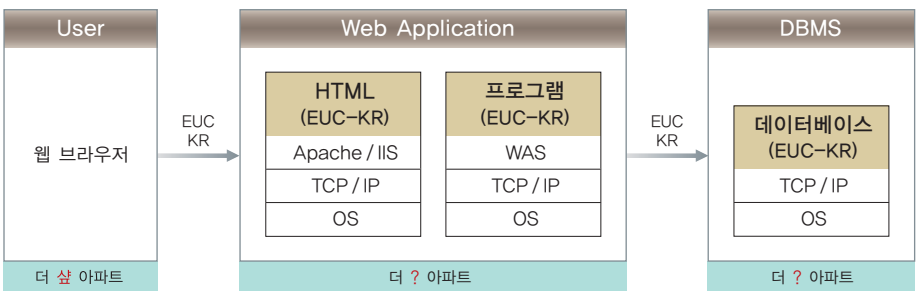
있어, Firefox에서 구동 시 문제가 발생할 수 있다. 모든 시스템들을 웹 표준에 맞게 수정하는 문제와도 연결될 수 있다.

※ 현재(2010년 8월) 국내에서 가장 많이 사용하는 IE를 비롯한 다른 브라우저에서는 KS채움쪽자 방식을 지원하지 않는다.

※ Firefox 외에 KS채움쪽자 방식을 지원하는 프로그램으로는 개발도구로 널리 쓰이고 있는 Eclipse가 있다.

나. 기존 시스템의 수정/보완 ① – 웹 기반 시스템

시스템을 순차적으로 수정, 보완해나가면서 최종적으로 전면적인 전환에까지 이르는 방법이다. 이 과정에서 중간 단계의 필요한 경우에 변환 모듈이 쓰이게 된다. 아래와 같은 웹 기반 시스템의 예를 들어 단계별 전환 방법을 살펴본다.



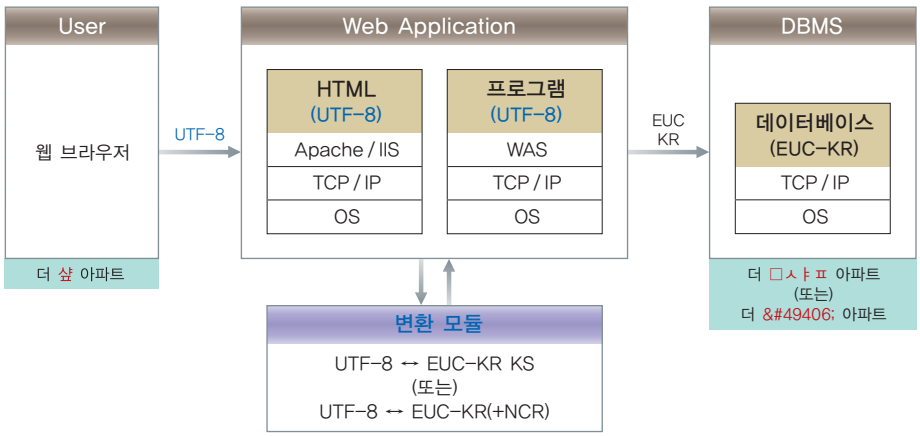
1) 입력 단계에서의 수정

가) 확장한글 데이터가 손상되지 않도록 하기 위해서는, 사용자가 입력하는 시작 단계에서부터 데이터를 안전하게 지켜야 한다.즉, 웹 기반 시스템의 경우 웹 응용프로그램이 사용자 입력을 받을 때부터, C/S 시스템의 경우 클라이언트 프로그램이 입력을 받을 때부터 확장한글이 깨지지 않고 입력되어야 한다.

나) 웹 응용프로그램에서 한글 데이터를 입력받을 때 EUC-KR 또는 UTF-8 인코딩이 사용된다. 이때 만약 EUC-KR 인코딩으로 데이터를 입력받으면, 일반적으로 확장한글은 깨진 상태로 서버로 전달되게 된다. 따라서 UTF-8 인코딩으로 사용자 입력을 받아들일 수 있도록 Web과 WAS 부분을 수정해야 한다.

다) 이렇게 일단 사용자 데이터를 UTF-8 인코딩으로 입력받고 나서, 기존의 DB가 EUC-

KR 데이터를 저장하고 있다면 그에 맞게 다시 인코딩을 변환한다. 단, 이때 변환 모듈을 이용하여 KS채움쪽자 방식 또는 NCR 방식을 적용해야 확장한글이 깨지지 않는다.



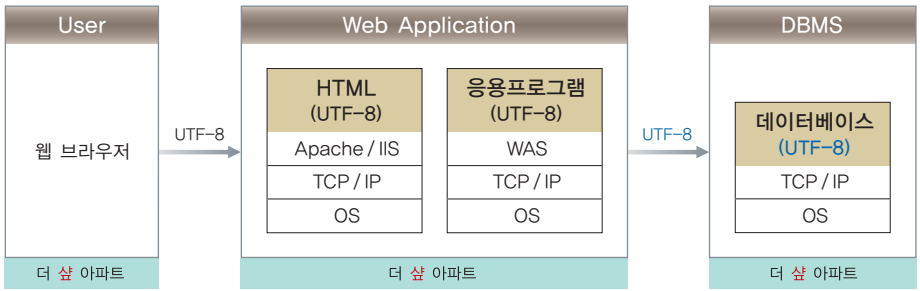
2) 저장 단계에서의 수정

가) 데이터가 저장되는 DB에서 UTF-8로 인코딩하여 저장하도록 전환하는 것이 중요하다.

나) 기존 DB가 MS949 데이터를 갖고 있는 경우, 확장한글 데이터는 깨지지 않으므로 일단은 전환하지 않고 한시적으로 사용 가능하나, 한글 외의 문자 대응 문제나 외부 시스템과의 연계 효율성 등을 생각해볼 때 최종적으로는 UTF-8로 전환이 필요하다. 그리고 한시적으로 사용하는 동안에도 외부로는 MS949 형식의 데이터가 전달되지 않도록 주의해야 한다.

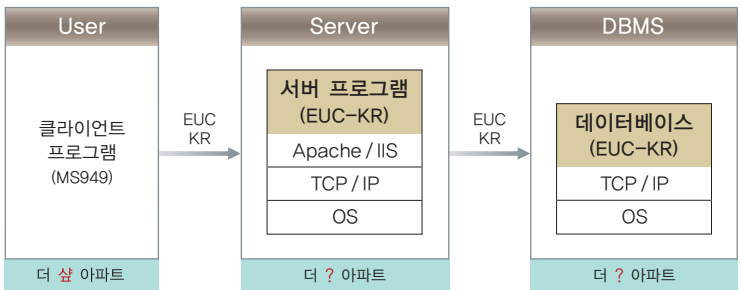
다) DB를 UTF-8 인코딩으로 전환할 때 기존 데이터의 마이그레이션(migration)이 필요할 것이다. 경우에 따라, 기존 데이터 중에 이미 깨진 글자가 들어있는 경우에는 데이터 클렌징(cleansing)도 필요할 수 있다.

라) 이제 DB가 UTF-8 데이터를 저장할 수 있게 되었으므로, 웹 응용프로그램에서 UTF-8 데이터를 전달해주면 그것을 변환하지 않고 바로 저장할 수 있다. 따라서, 입력 단계에서의 수정에 필요했던 변환 모듈은 이제 필요 없게 되므로 제거해도 된다.



다. 기존 시스템의 수정/보완 ② – C/S 시스템

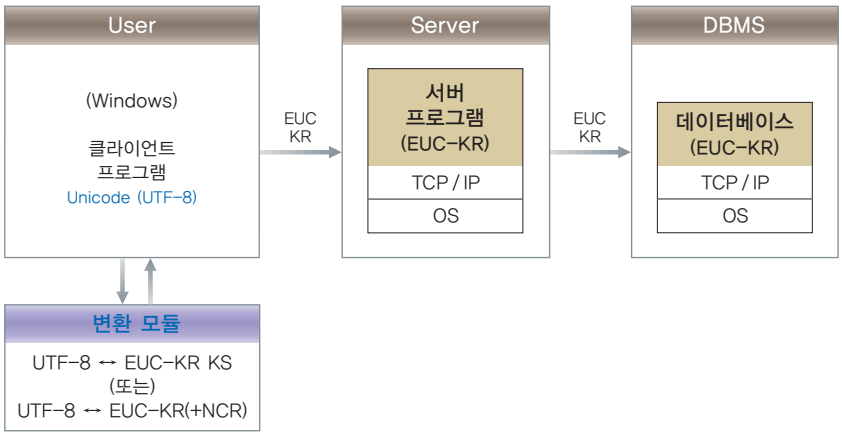
C/S 시스템의 경우도 사용자 입력을 받는 부분이 클라이언트 프로그램이라는 점 외에는 웹 기반 시스템의 경우와 유사하다고 볼 수 있다. 아래 C/S 시스템의 단계별 전환 과정을 살펴본다.



1) 입력 단계에서의 수정

가) 대부분 클라이언트 프로그램은 Windows에서 동작하고, 유니코드 또는 MS949 문자 인코딩으로 사용자 입력을 처리한다. 따라서 클라이언트 프로그램 자체는 확장한글을 처리할 수 있지만, 그 다음 단계에서 데이터가 깨질 수 있다.

나) 따라서 클라이언트 프로그램을 유니코드(또는 UTF-8) 기반으로 수정하고, 변환 모듈을 이용해 EUC-KR KS 또는 EUC-KR (+NCR) 데이터를 만들어 다음 단계의 서버 프로그램이나 DB에 전달해줄 수 있도록 해야 한다.

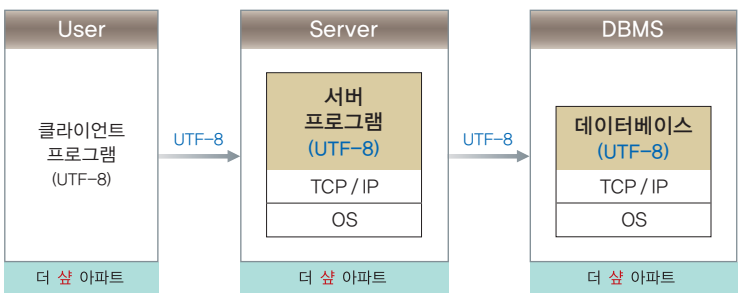


2) 저장 단계에서의 수정

가) 웹 기반 시스템에서의 경우와 같이, DB를 UTF-8로 전환하는 단계이다. 데이터의 마이그레이션과 클렌징이 필요한 것도 웹 기반 시스템에서와 같다.

나) 여기서, 경우에 따라 서버 프로그램을 수정해야 할 수도 있고, 수정하지 않을 수도 있다. 만약 서버가 데이터를 어떤 식으로든 조작/가공하는 역할을 하고 있는 경우라면 서버도 UTF-8 인코딩으로 문자열을 다루도록 수정해야 한다.그렇지 않고, 서버가 데이터는 건드리지 않고 단지 DB 쪽에 전달만 해주는 역할만 한다면 서버는 수정하지 않고 그대로 두어도 된다.

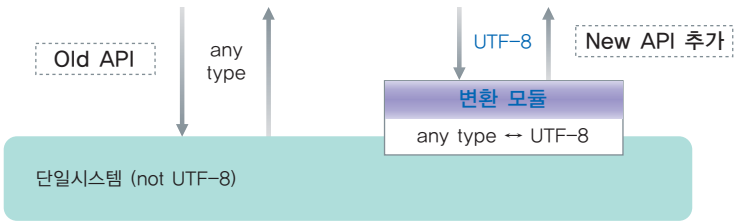
다) 모든 부분이 UTF-8을 쓰게 되었으므로 변환 모듈은 이제 사용하지 않아도 된다.



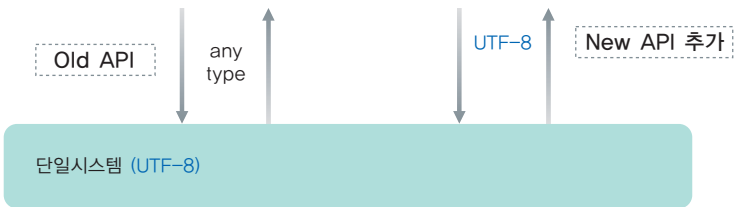
라. 기존 시스템의 수정/보완 ③ – 연계 API의 수정

타 시스템과 연계되는 시스템인 경우, 데이터 교환 방식의 차이로 발생하는 혼란을 방지하기 위해 인코딩 타입을 UTF-8로 고정해야 한다. 다만, 시스템에 따라 UTF-8로 전환되거나 그렇지 않은 시스템들이 혼재되어 있으므로 기존의 API도 그대로 유지한다.

1) UTF-8을 사용하는 API를 추가한다. 그리고 단일시스템 내부에 UTF-8이 적용되지 않은 상황이라면 연계 API를 위해 변환 모듈이 필요하다.



2) 단일시스템이 내부적으로 UTF-8로 전환된 후에는, 연계 API 사용 시 변환 모듈이 필요 없다.



3) 추후 이 시스템과 연계된 상대 시스템들이 모두 신규 API(UTF-8)로 연계된다면, 기존의 연계 방식(Old API)은 더 이상 사용되지 않으므로 제거해도 된다.

마. 전면적 전환

시스템의 모든 부분을 유니코드(또는 UTF-8) 기반으로 만들어 확장한글 문제가 발생하지 않도록 하는 방법이다. 새 시스템을 구축하거나, 이전 시스템을 전면적으로 개편할 때에 적용할 수 있다.

1) 입력부

가) 웹 기반 시스템 : Web과 WAS에서의 데이터 처리가 UTF-8로 이루어지도록 설정한다. 사용자에게 보여지는 HTML 페이지들과 사용자가 입력하는 데이터 모두 UTF-8 인코딩이 사용되도록 한다.

나) C/S 시스템 : 클라이언트 프로그램을 유니코드 기반으로 만들고, 프로그램의 입출력 데이터는 UTF-8 인코딩을 사용하도록 한다.

2) 처리부

가) 웹 기반 시스템 : WAS에서 DB로 데이터가 전달될 때 데이터를 가공/처리하는 부분이 있다면 UTF-8 인코딩으로 문자열 데이터를 처리하도록 만든다.

나) C/S 시스템 : 서버 프로그램에서 DB로 데이터가 전달될 때 데이터를 가공/처리하는 부분이 있다면 UTF-8 인코딩으로 문자열 데이터를 처리하도록 만든다.

3) 저장부

가) DB에 저장되는 문자열 데이터는 UTF-8 인코딩을 사용한다.

나) 기존의 DB에서 데이터를 가져와야 하는 경우라면, 일괄적으로 인코딩 변환 작업을 거친다.

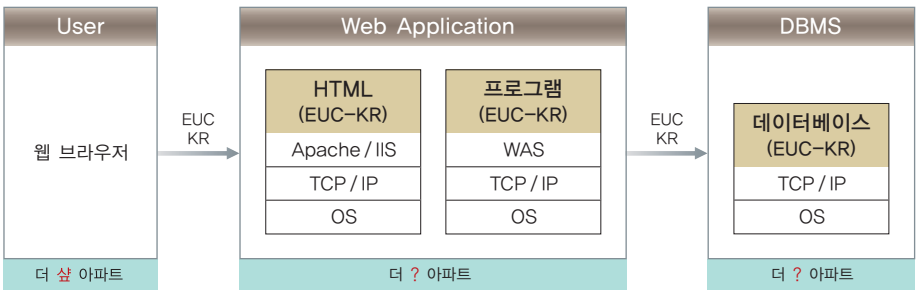
다) 기존의 DB에서 데이터를 가져오는 경우에, 기존의 데이터 중에서 확장한글이 깨진 것이 있다면, 원래의 글자로 복원하기 위한 클렌징(cleansing) 작업이 필요하다.

2. 유형별 전환계획

주요 유형별 확장한글 해결을 위한 전면적/부분적 해결 방안을 제시한다.

가. A1B 유형

웹 기반 시스템으로 응용프로그램과 DB 데이터 처리를 모두 수정해야 하는 유형이다.



1) 유형 소개

가) 웹기반 시스템에서 웹 응용프로그램의 인코딩 타입이 확장한글을 지원하는 형태가 아니고, 데이터베이스의 인코딩 타입 역시 확장한글을 고려하지 않은 경우

나) 외부 시스템과의 송수신은 고려하지 않음

다) 사용자가 웹 브라우저를 통해 입력한 확장한글 데이터가 DBMS에 정상적으로 저장되기까지의 중간 단계(웹 응용프로그램, 데이터베이스)를 수정해야 하는 유형

2) 유형 분석

가) 사용자로부터 입력되는 데이터는 웹 응용프로그램의 인코딩 설정(HTML)에 따라 웹브라우저에서 EUC-KR 타입으로 인코딩되어 전달이 된다. EUC-KR 타입은 확장

한글을 지원하지 않으므로, 입력되는 원천 데이터에서부터 이미 확장한글은 지원되지 않는 형태로 전달된다.

나) 이후 웹 응용프로그램에서 가공 및 처리를 거쳐 데이터베이스로 저장되는 과정까지 EUC-KR 타입을 사용한다.

다) 데이터가 이동되는 전 과정이 EUC-KR로 통일되어 있으나, 입력 데이터에서부터 확장한글이 처리되지 못함으로써, 이후 단계에서도 그대로 확장한글 문제를 승계하게 된다.

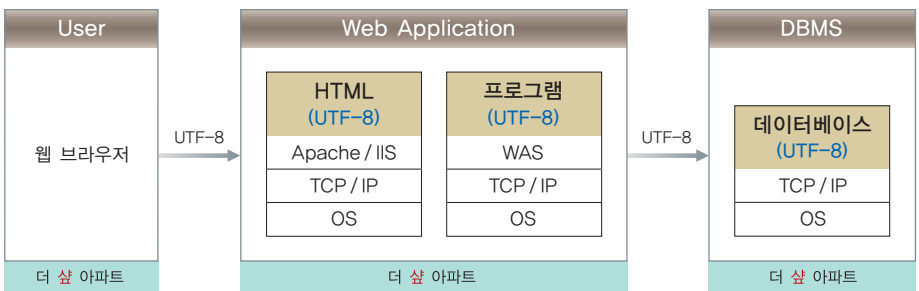
3) 해결 과제

가) 사용자로부터 입력되는 확장한글 원천 데이터에 문제가 없어야 한다.

나) 정상적으로 입력된 확장한글 데이터가 데이터베이스에 정상 저장되어야 한다.

4) 전면적 해결 방안

전체 구간을 확장한글 처리가 가능한 유니코드(UTF-8)로 변경함으로써, "입력→처리→저장"의 모든 과정에서 확장한글이 정상 처리되도록 구현한다.



가) 해결 단계

(1) 웹 응용프로그램 내의 인코딩 타입 선언을 UTF-8로 변경한다.

웹페이지에서의 유니코드(UTF-8) 인코딩 타입 정의 예제

구분	종류	내용
브라우저 인코딩	HTML	〈meta http-equiv="content-type" content="text/html;charset=UTF-8" /〉
서버 App 인코딩	Java	〈%@page contentType="text/html; charset=UTF-8"%〉
	PHP	header("Content-Type: text/html; charset=UTF-8");

(2) 입력된 데이터가 저장되는 데이터베이스의 인코딩 타입 역시 UTF-8로 변경한다.

(가) DB 마이그레이션 (Migration) 필요 : 기존 인코딩 타입(EUC-KR)에서 UTF-8로 변경하기 위해서 저장된 데이터들의 일괄 변환작업을 거쳐야 한다.

(다) DB 클렌징 (Cleansing) 필요 : 기존 DB에 잘못 저장된 확장한글 데이터가 있다면, 해당 데이터를 정상 데이터로 수정하는 클렌징 작업을 진행해야 한다.

나) 장점

(1) 데이터의 입력에서부터 저장에 이르기까지 모든 과정이 유니코드(UTF-8) 타입으로 통일되어 데이터 전송 시에 별도의 변환 과정이 필요 없다.

(2) 유니코드(UTF-8)는 확장한글을 정상적으로 표현/저장할 수 있는 인코딩 타입으로, 한글 표현에 제한이 없으며, 한글 뿐만 아니라 전 세계 어느 문자로도 표현이 가능하므로 정보시스템의 국제화에 필수 요소이다.

(3) 추후 다른 시스템과의 연계 시에도 유니코드로 통일된 데이터 송수신을 처리할 수 있다.

다) 단점

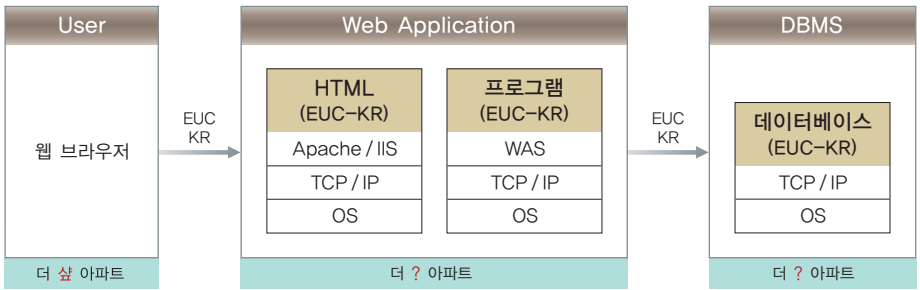
(1) 현재 운영중인 시스템인 경우, 전면적인 개편이 필요하므로, 개편에 따르는 비용을 감안해야 한다. (웹 응용프로그램 인코딩 타입 수정, DB 마이그레이션 및 클렌징)

5) 부분적 해결 방안 (1) – 변환 모듈 적용

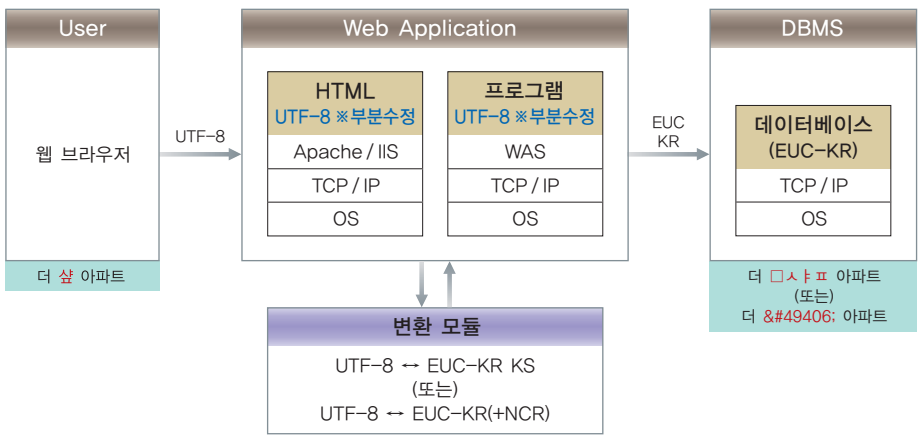
기존 웹 응용프로그램과 데이터베이스의 인코딩 환경(EUC-KR)을 그대로 유지하고, 확장한글 입력이 예상되는 페이지만 유니코드(UTF-8) 타입으로 입력받은 뒤, 변환 모듈을 사용하여 'EUC-KR KS' 또는 'EUC-KR (+NCR)' 형태로 DB에 저장한다.

예 이름, 주소 필드가 포함된 페이지

〈기존 입력 페이지 – 유지〉



〈확장한글 입력 수정페이지〉



가) 해결 단계

(1) 확장한글 입력이 예상되는 페이지의 인코딩 타입을 유니코드(UTF-8)로 변경한다.

웹페이지에서의 유니코드(UTF-8) 인코딩 타입 정의 예제

구분	종류	내용
브라우저 인코딩	HTML	〈meta http-equiv="content-type" content="text/html;charset=UTF-8" /〉
서버 App 인코딩	Java	〈%@page contentType="text/html; charset=UTF-8"%〉
	PHP	header("Content-Type : text/html; charset=UTF-8")

(2) UTF-8로 입력된 데이터를 변환 모듈을 사용하여 확장한글이 가능한 형태로 변환하는 것으로, EUC-KR 타입의 데이터베이스를 그대로 사용하여 저장한다.

(가) KS채움쪽자 방식 (EUC-KR KS) : 확장한글에 해당하는 문자를 KS규격에 따라 변환하여 지원한다. (UTF-8 → EUC-KR KS)

(나) NCR 방식 (EUC-KR + NCR) : 확장한글을 HTML Character Entity로 표현하는 것으로 "샾"와 같은 형태로 변환한다.

(3) 변환 모듈을 사용하여 데이터베이스에 저장된 데이터는, 데이터 출력 시에도 변환 모듈을 사용하여 적절한 인코딩으로 변환하여 출력한다.

예 EUC-KR KS → UTF-8)

(4) 이후 시스템 전환 시점이 도래하면, "전면적 해결 방안"을 참고하여 유니코드 시스템으로 전환한다.

나) 장점

(1) 기존 웹 응용프로그램의 인코딩 설정을 그대로 두고, 확장한글 이슈가 있는 페이지만을 수정하므로, 전체 인코딩 변경에 대한 부담을 줄일 수 있다.

(2) 데이터베이스를 기존 EUC-KR 환경 그대로 사용할 수 있으므로, DB 전환에 대한 비용 부담을 줄일 수 있다.

다) 단점

(1) 데이터베이스에서 '이름'과 같이 DB 필드의 자릿수 제한이 있는 경우, 이를 초과할 수 있다.

확장한글 1글자 2byte → 확장한글 모듈 변환 처리 후 8byte

최근 4자리 이상의 이름(우리말 이름)이나 외국 귀화자 이름 표기를 고려하여 자릿수가 확장되어 있어 대부분 문제가 없을 것으로 판단. (작업 전 '이름' 필드의 자릿수 제한을 확인할 필요 있음)

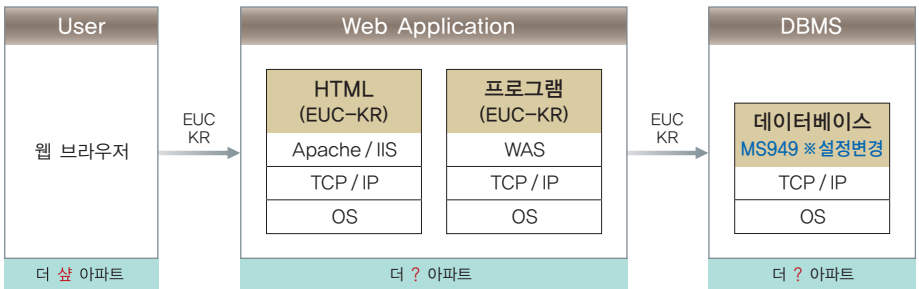
(2) 확장한글 입력이 예상되는 페이지만 UTF-8로 전환하여 처리하는 것은 임시방편으로, 추후 전면적 해결방안을 다시 고려해야 한다.

6) 부분적 해결 방안 (2) – MS949 인코딩

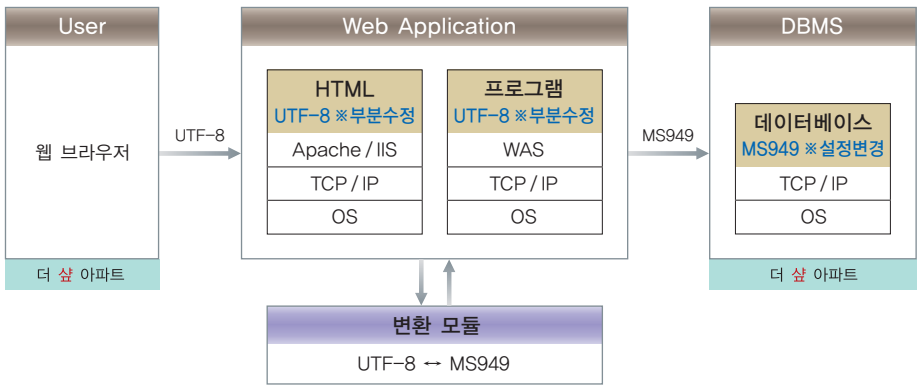
기존 웹페이지들은 EUC-KR 인코딩 환경을 그대로 유지하고, 데이터베이스는 MS949로 설정을 변경한다.확장한글 입력이 예상되는 페이지만, 유니코드(UTF-8) 타입으로 입력받은 뒤 변환모듈을 사용하여 'MS949'로 변환 처리를 한다.

예 이름, 주소 필드가 포함된 페이지

〈기존 입력 페이지 – DB 설정 변경〉



〈확장한글 입력 수정페이지 – DB 설정 변경〉



가) 해결 단계

(1) 데이터베이스의 인코딩 타입을 EUC-KR에서 MS949로 변경한다.

(가) MS949는 EUC-KR의 확장 형태이므로, 데이터베이스에서 인코딩 설정 변경으로 적용할 수 있다. (DB 마이그레이션 불필요)

※ 데이터베이스 종류별 설정 변경 방법 or MS949 지원 유무 별첨 자료 참조

(나) 설정 변경 이후 입력되는 확장한글은 정상으로 저장되겠으나, 그 이전에 입력

하였던 확장한글은 비정상 입력되어 있을 것이므로, 이를 복구하기 위해서 별도의 DB 클렌징 과정이 필요할 수 있다.

(2) 확장한글 입력이 예상되는 페이지의 인코딩 타입 선언을 UTF-8로 변경한다.

웹페이지에서의 유니코드(UTF-8) 인코딩 타입 정의 예제

구분	종류	내용
브라우저 인코딩	HTML	〈meta http-equiv="content-type" content="text/html;charset=UTF-8" /〉
서버 App 인코딩	Java	〈%@page contentType="text/html; charset=UTF-8"%〉
	PHP	header("Content-Type : text/html; charset=UTF-8")

(3) UTF-8로 입력된 데이터를 변환 모듈을 사용하여 MS949 타입으로 변환하고 데이터베이스에 저장한다.

(가) 입력된 데이터를 변환 모듈을 사용하여 MS949 타입으로 변환하고, 앞서 MS949로 설정 변경한 데이터베이스에 저장하는 방식 (UTF-8 → MS949)

(나) 변환 모듈을 사용하여 데이터베이스에 저장된 데이터는, 데이터 출력 시에도 변환 모듈을 사용하여 적절한 인코딩으로 변환하여 출력한다. (MS949 → UTF-8)

(4) 이후 시스템 전환 시점이 도래하면, "전면적 해결 방안"을 참고하여 유니코드 시스템으로 전환한다.

나) 장점

(1) 기존 웹 응용프로그램의 인코딩 설정을 그대로 두고, 확장한글 이슈가 있는 페이지만을 수정하므로, 인코딩 변경에 대한 부담을 줄일 수 있다.

(2) 기존 데이터베이스의 인코딩(EUC-KR)에서 설정 변경만으로 MS949 인코딩을 지원할 수 있으므로, DB 마이그레이션에 대한 비용 부담을 줄일 수 있다.

다) 단점

(1) MS949는 국내에서만 국한되어 사용되는 확장 코드 방식으로 글로벌 환경 또는 모바일 환경 고려 시, 제한이 발생할 수 있다.

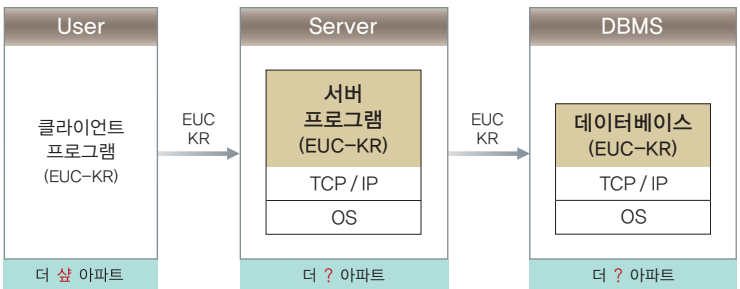
※ MSDN의 MS949 (CodePage949) 소개 내용 발췌
(http://msdn.microsoft.com/ko-kr/goglobal/cc305154%28en-us%29.aspx)

> Using Unicode is recommended in preference to any code page because it has better language support and is less ambiguous than any of the code pages.

(2) 확장한글 입력이 예상되는 페이지만 UTF-8로 전환하여 처리하는 것은 임시방편으로, 추후 전면적 해결방안을 다시 고려해야 한다.

나. A2B 유형

C/S 기반 시스템으로 응용프로그램과 DB 데이터 처리를 모두 수정해야 하는 유형이다.



1) 유형 소개

가) C/S기반 시스템에서 클라이언트, 서버, 데이터베이스의 인코딩 타입이 모두 확장한글을 고려하지 않는 경우 (EUC-KR)

나) 외부 시스템과의 송수신은 고려하지 않음

다) 사용자가 클라이언트 프로그램을 통해 입력한 확장한글 데이터가 데이터베이스에 정상적으로 저장되기까지의 모든 단계를 수정해야 하는 유형

2) 유형 분석

가) 사용자로부터 입력되는 데이터는 클라이언트 프로그램에서 지정된 인코딩 타입(EUC-KR)으로 서버 프로그램으로 전달된다. 이때 확장한글은 지원되지 않는 문자로 처리된다.

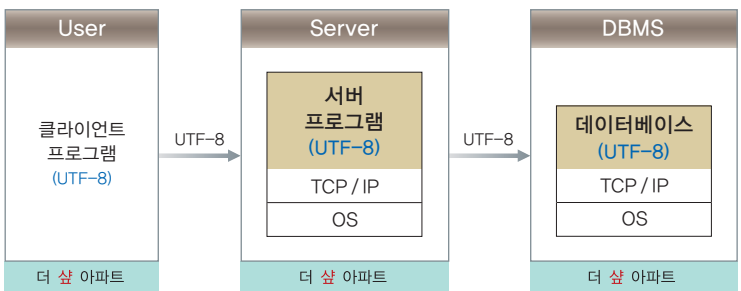
나) 클라이언트 프로그램 단계에서부터 지원되지 않는 확장한글은 이후 모든 단계에서도 정상 저장되지 않고 출력되지 않는다.

3) 해결 과제

가) 클라이언트와 서버가 확장한글 데이터를 주고받는 데 문제가 없어야 한다.
나) 데이터베이스에 확장한글이 정상 저장되어야 한다.

4) 전면적 해결 방안

전체 구간을 확장한글 처리가 가능한 유니코드(UTF-8)로 변경함으로써,"입력→처리→저장"의 모든 과정에서 확장한글이 정상 처리되도록 구현한다.



가) 해결 단계

(1) 클라이언트 와 서버 프로그램에서 유니코드(UTF-8)를 지원하도록 프로그램을 수정한다.

(2) 입력된 데이터가 저장되는 데이터베이스의 인코딩 타입 역시 UTF-8로 변경한다.

(가) DB 마이그레이션(migration) 필요 : 기존 인코딩 타입(EUC-KR)에서 UTF-8로 변경하기 위해서 저장된 데이터들의 일괄 변환작업을 거쳐야 한다.
(나) DB 클렌징(cleansing) 필요 : 기존 DB에 잘못 저장된 확장한글 데이터가 있다면, 해당 데이터를 정상 데이터로 수정하는 클렌징 작업을 진행해야 한다.

나) 장점

(1) 데이터의 입력에서부터 저장에 이르기까지 모든 과정이 유니코드(UTF-8) 타입으로 통일되어 데이터 전송 시에 별도의 변환 과정이 필요 없다.

(2) 유니코드(UTF-8)는 확장한글을 정상적으로 표현/저장할 수 있는 인코딩 타입으로, 한글 표현에 제한이 없으며, 한글 뿐만 아니라 전 세계 어느 문자로도 표현이 가능하므로 정보시스템의 국제화에 필수 요소이다.

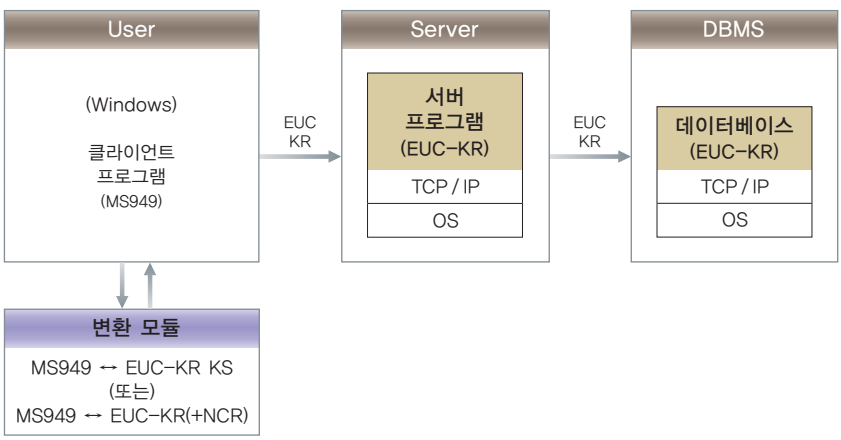
(3) 추후 다른 시스템과의 연계 시에도 유니코드로 통일된 데이터 송수신을 처리할 수 있다.

다) 단점

(1) 현재 운영중인 시스템인 경우, 전면적인 개편이 필요하므로, 개편에 따르는 비용을 감안해야 한다.(클라이언트/서버 프로그램 수정, DB 마이그레이션 및 클렌징)

5) 부분적 해결 방안 (1) – 변환 모듈 적용

기존 서버 프로그램과 데이터베이스의 인코딩 환경(EUC-KR)을 그대로 유지하고, 확장한글을 입력하는 클라이언트 프로그램에서 변환 모듈을 통해 'EUC-KR KS' 또는 'EUC-KR (+NCR)' 형태로 변환하도록 수정한다.



가) 해결 단계

(1) 입력된 데이터를 변환 모듈을 사용하여 EUC-KR에서 사용할 수 있는 확장한글 형태로 변환한다.

(가) KS채움쪽자 방식 (EUC-KR KS) : 확장한글에 해당하는 문자를 KS규격에 따라 변환하여 지원한다. (MS949 → EUC-KR KS)

(나) NCR 방식 (EUC-KR + NCR) : 확장한글을 HTML Character Entity로 표현하는 것으로 "샾"와 같은 형태로 변환한다.

(2) 입력 단에서 데이터가 EUC-KR로 전달 가능한 타입으로 변환되었으므로, 이후 서버 프로그램 및 데이터베이스 단에서는 별도의 수정 없이 사용할 수 있다.

(3) 변환 모듈을 사용하여 서버단에 전달된 데이터는, 데이터 출력 시에도 변환 모듈을 사용하여 적절한 타입으로 변환하여 출력한다. (예 : EUC-KR KS → MS949)

(4) 이후 시스템 전환 시점이 도래하면, "전면적 해결 방안"을 참고하여 유니코드 시스템으로 전환한다.

나) 장점

(1) 클라이언트에 적용된 변환 모듈을 사용하여 확장한글을 변환/처리하므로, 각 구간별 인코딩 타입을 변경하지 않고도 사용할 수 있다.

다) 단점

(1) Windows 클라이언트 프로그램에만 해당한다. (MS949를 지원하는 OS)

(2) 데이터베이스에서 '이름'과 같이 DB 필드의 자릿수 제한이 있는 경우, 이를 초과할 수 있다.

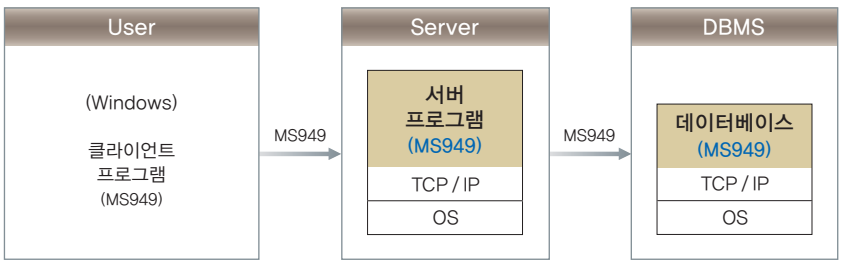
확장한글 1글자 2byte → 확장한글 모듈 변환 처리 후 8byte

최근 4자리 이상의 이름(우리말 이름)이나 외국 귀화자 이름 표기를 고려하여 자릿수가 확장되어 있어 대부분 문제가 없을 것으로 판단. (작업 전 '이름' 필드의 자릿수 제한을 확인할 필요 있음)

(3) 변환 모듈을 사용하여 처리하는 것은 임시방편으로, 추후 전면적 해결방안을 다시 고려해야 한다.

6) 부분적 해결 방안 (2) – MS949 사용

클라이언트 프로그램에서 입력되는 확장한글을 정상 저장하기 위해 서버 프로그램과 데이터베이스 단을 MS949 타입으로 수정한다.



가) 해결 단계

(1) 서버에서 EUC-KR 인코딩의 영향을 받을 경우, 프로그램의 문자 인코딩 타입을 MS949로 수정한다.

(2) 데이터베이스의 인코딩 타입을 EUC-KR에서 MS949로 변경한다.

(가) MS949는 EUC-KR의 확장 형태이므로, 데이터베이스에서 인코딩 설정 변경으로 적용할 수 있다. (DB 마이그레이션 불필요)※ 데이터베이스 종류별 설정 변경 방법 or MS949 지원 유무 별첨 자료 참조

(나) 설정 변경 이후 입력되는 확장한글은 정상으로 저장되겠으나, 그 이전에 입력하였던 확장한글은 비정상 입력되어 있을 것이므로, 이를 복구하기 위해서 별도의 DB 클렌징 과정이 필요할 수 있다.

(3) 이후 시스템 전환 시점이 도래하면, "전면적 해결 방안"을 참고하여 유니코드 시스템으로 전환한다.

나) 장점

(1) 클라이언트와 서버 프로그램의 수정을 최소화할 수 있다.

(2) 기존 데이터베이스의 인코딩(EUC-KR)에서 설정 변경만으로 MS949 인코딩을 지원할 수 있으므로, DB 마이그레이션에 대한 비용 부담을 줄일 수 있다.

다) 단점

(1) Windows 클라이언트 프로그램에만 해당한다. (MS949를 지원하는 OS)

(2) Windows 이외의 환경에서 클라이언트를 신규 개발할 시, 제약을 받을 수 있다.

(3) MS949는 국내에서만 국한되어 사용되는 확장 코드 방식으로 글로벌 환경 또는 모바일 환경 고려 시, 제한이 발생할 수 있다.

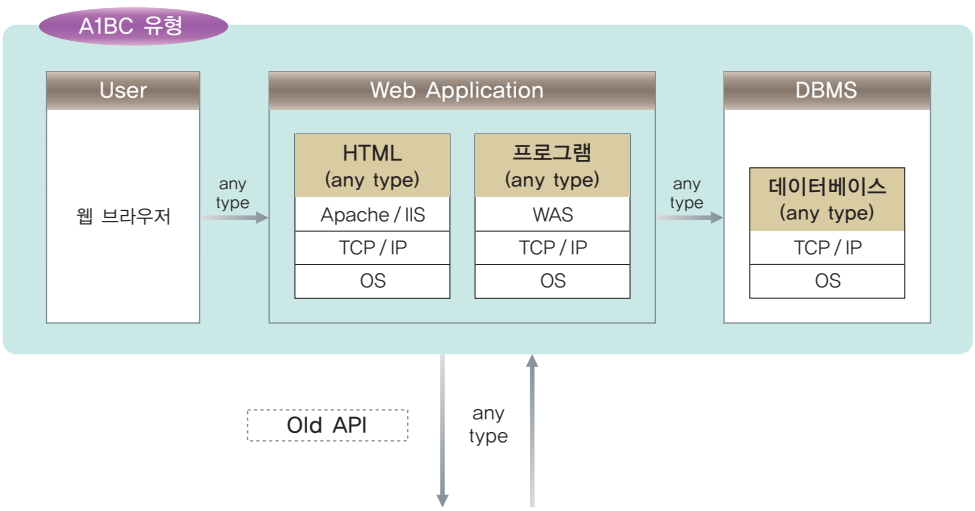
※ MSDN의 MS949 (CodePage949) 소개 내용 발췌

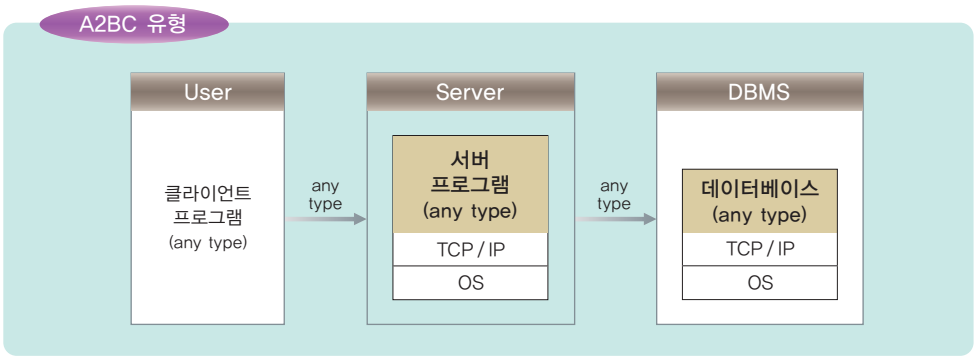
(http://msdn.microsoft.com/ko-kr/goglobal/cc305154%28en-us%29.aspx)

> Using Unicode is recommended in preference to any code page because it has better language support and is less ambiguous than any of the code pages.

다. ABC 유형

연계되는 시스템에 대한 구성으로, 각 시스템은 응용프로그램과 DB의 수정이 필요하고, 이들 시스템 간에 연계 관계가 이루어지는 유형이다.





1) 유형 소개

가) 연계 구조에서 서로 상이한 두 시스템 간의 연계 (예 : 웹 기반 ↔ C/S 기반)

나) 두 시스템 간의 송수신 데이터가 유니코드(UTF-8)인 경우, 별도 변환 작업에 대한 고려 없이 운영 가능

다) 두 시스템 중 송수신 데이터가 어느 한 곳이라도 EUC-KR 또는 MS949 인코딩 타입일 경우, 별도 변환 작업에 대한 고려 필요

라) 각 시스템 단에서의 확장한글 해결방안은, 각각의 해결방안 참조 (A1B, A2B 등)

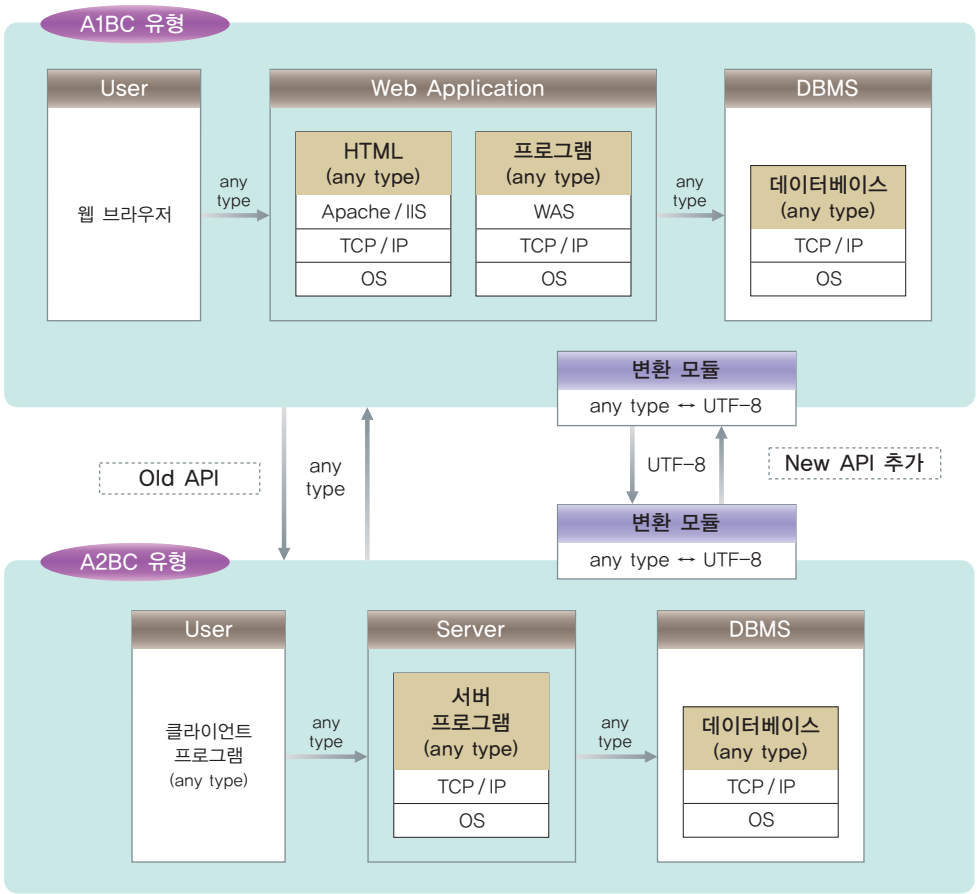
2) 해결 과제

가) 두 시스템 간의 정상적인 데이터 전송을 위해 연계 단에서의 인코딩 타입은 유니코드(UTF-8)를 기준으로 한다.

나) 각 시스템은 어떠한 인코딩 타입을 사용하더라도 확장한글 데이터는 처리 가능해야 한다.

3) 해결 방안

연계 입출력 구간을 확장한글 처리가 가능한 유니코드(UTF-8)로 변경함으로써,각 단일 시스템에서 사용할 확장한글이 정상 전송되도록 구현한다.



가) 해결 단계

(1) 각 시스템에서 제공하는 기존 API가 UTF-8인 경우, 별도 수정 없이 연계할 수 있다.

(2) 각 시스템에서 제공하는 기존 API가 UTF-8이 아닌 경우, 기존 API를 그대로 두고 동일한 역할을 수행하는 UTF-8 신규 API를 추가한다. ※ 기존 API를 제거하지 않고 유지하는 이유는, 연계되는 모든 시스템을 한 번에 변경할 수 없고, 순차적으로 변경할 시 신규 API와 기존 API가 같이 필요하기 때문이다.

(3) 시스템 간 연계를 할 때, 상대 시스템에게 추가된 신규 API를 사용하도록 고지하고 유도한다.

나) 장점

(1) UTF-8 인코딩 타입으로 연계를 하면 확장한글 송수신이 보장되고, 단일시스템이 어떠한 인코딩 타입을 사용하더라도 연계되는 시스템에 영향을 주지 않는다.

(2) 연계 API가 UTF-8로 구현되면 각 단일시스템의 변경이 비교적 자유로우며, 이와 연계된 다른 단일시스템에 미치는 영향도 미약하다.

다) 단점

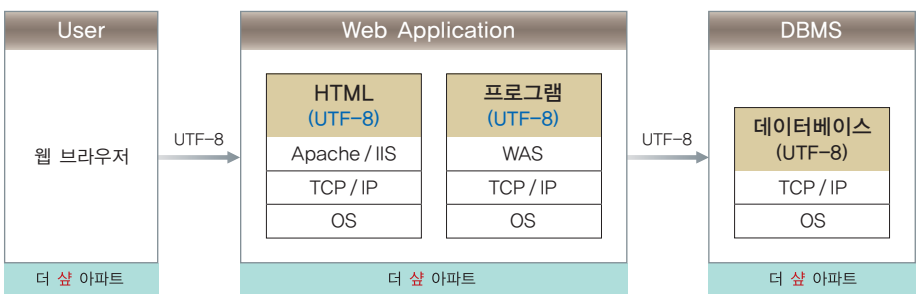
(1) 연계 API 추가 작업 시, 연계되는 상대 시스템까지 같이 작업이 진행되지 않으면 확장한글 개선의 효과가 나타나지 않는다.

라. 기타 유형

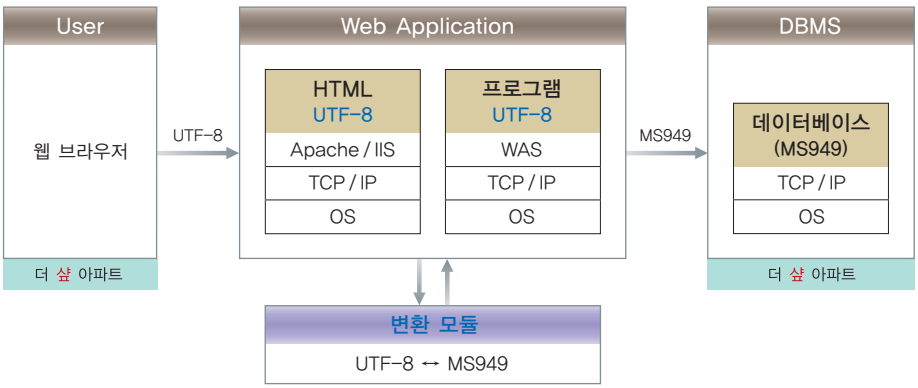
1) A 유형

응용프로그램의 수정만 필요한 유형으로, DB의 종류에 따라 A1B 유형을 참조한다. (C/S 환경인 경우는 A2B 참조)

가) DB가 UTF-8인 경우 : 'A1B의 전면적 해결방안' 참조



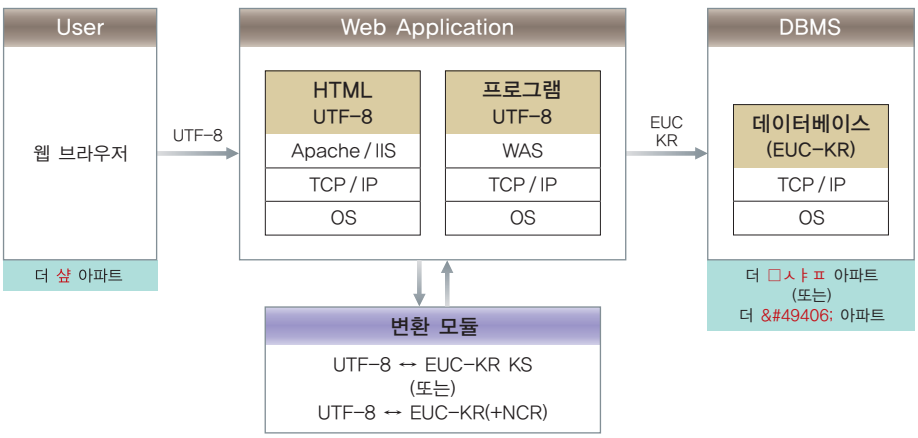
나) DB가 MS949인 경우 : 'A1B의 부분적 해결방안 (2) – MS949' 인코딩 참조



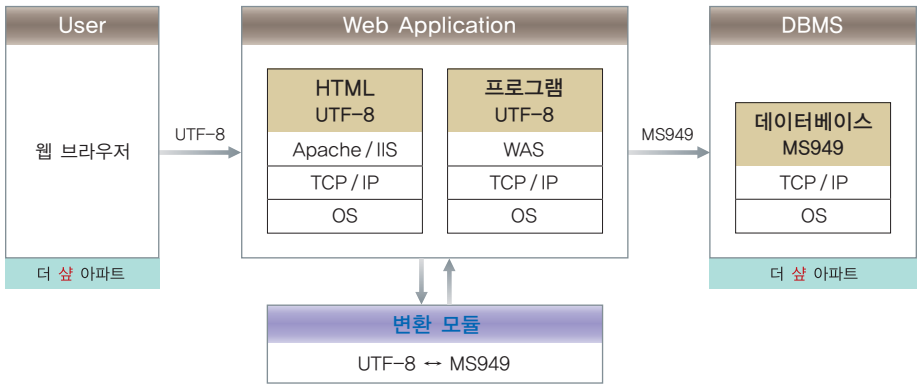
2) B 유형

응용프로그램은 유니코드(또는 UTF-8)로 구성되어 있고, DB의 수정만 필요한 유형으로, A1B 유형을 참조한다. (C/S 환경인 경우는 A2B 참조)

가) DB가 EUC-KR인 경우 : '부분적 해결 방안 (1) – 변환 모듈 적용' 참조

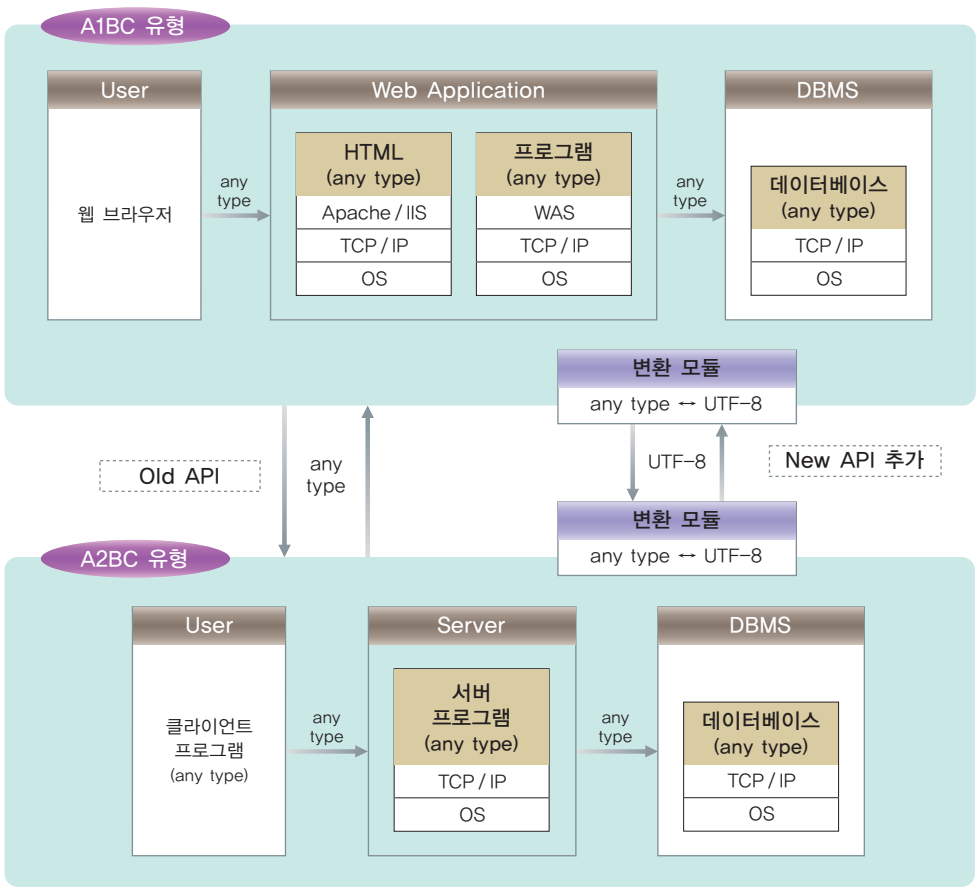


나) DB가 MS949인 경우 : '부분적 해결 방안 (2) – MS949 인코딩' 참조



3) C 유형

연계되는 구간에 대한 수정만 필요한 유형으로, ABC 유형을 참조한다.



4) AC 유형

응용프로그램과 연계 구간에 대한 수정이 필요한 유형으로, A 유형과 C 유형을 각각 참조한다.

5) BC 유형

DB와 연계 구간에 대한 수정이 필요한 유형으로, B 유형과 C 유형을 각각 참조한다.

6) O 유형

응용프로그램, DB, 연계 구간의 모든 문자 인코딩이 유니코드(또는 UTF-?8)로 처리되어 있어, 별다른 수정이 필요 없는 유형이다.

7) X 유형

특정 유형으로 분류하기 어려운 유형으로, 개별 문제점을 파악하고 그에 따라 적절한 조치를 취하여야 한다.

V. 전환 비용 산출

1. 모의운영 실시
2. 전환 비용 분석

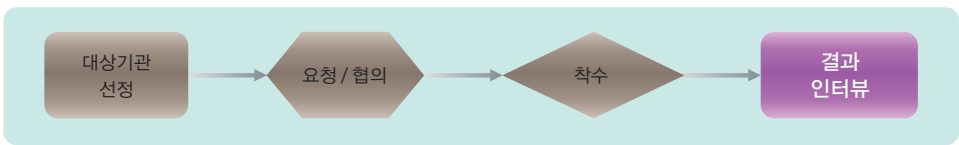
확장한글 문제 해결 시 시스템 개선에 필요한 예산을 추정할 수 있는 방안을 마련한다. 현실성 있는 비용 산정을 위해, 공공기관의 정보시스템을 대상으로 모의운영을 실시하고, 그 결과를 바탕으로 전환 비용을 산출할 수 있는 기준을 마련한다.

1. 모의운영 실시

현재 운영 중인 정보시스템 중 확장한글 문제가 발생되었거나 또는 발생 예상되는 정보시스템을 대상으로 하여, 관계 기관의 협조와 지원으로 모의운영을 실시하게 되었다.

가. 모의운영 절차

모의운영을 수행을 위한 각 단계별 진행 절차



1) 모의운영 대상 기관 선정

정보시스템 분석을 통해 선정된 대표시스템을 모의운영 대상으로 하였다.

2) 모의운영 요청 및 협의

모의운영 진행 여부에 대해 기관 관계자와 협의 및 협조 요청 수행

3) 모의운영 착수

가) 모의운영을 위한 테스트 시스템 구성

실제 운영되는 시스템과 동일한 구성을 가지는 모의운영용 테스트 시스템을 구성하여 확장한글 개선 여부를 검증한다.

(테스트 시스템이 이미 구성되어 있는 경우 해당 시스템을 활용한다.)

나) 개선 전 확장한글 입출력 확인

모의운영 대상 시스템에서 확장한글 문제 발생이 예상되는 '주민 이름' 및 '주소' 등을 대상으로, 확장한글이 포함된 단어의 입출력 테스트를 수행하여 문제 발생 여부를 확인한다.

다) 변환 모듈 적용

확장한글 문제 해결을 위한 여러 방법 중, 변환 모듈을 사용하는 부분적 해결 방안을 적용한다. 데이터를 처리하는 응용프로그램에서 변환 모듈을 호출하여 사용할 수 있도록 수정 적용한다.

라) 피드백 및 수정

변환 모듈 적용 시 오류 및 피드백이 발생하면, 해당 오류를 검증/수정/반영하여 재적용을 시도한다.

마) 개선 후 확장한글 입출력 확인

변환 모듈 적용이 완료된 시스템에서 확장한글이 정상 처리되는지를 확인한다.

4) 모의운영 결과 인터뷰

모의운영 수행에 대한 결과를 리포팅하고, 심층 인터뷰를 통해 투입 리소스 확인 및 비용 산출 데이터를 수집한다.

나. 모의운영 진행 일정

모의운영 대상에 대해, '모의운영 절차'에 따라 협의 및 작업 착수가 진행되었으며, 결과에 대한 리포팅(인터뷰)를 수행하였다.

1) '가' 시스템

일시	작업
2010-08-11	관계자 미팅 및 현황 조사
2010-08-19	모의운영 진행 여부 협의
2010-08-25	모의운영 협조 요청
2010-08-27	모의운영 실무 협의 미팅 (1차)
2010-09-06	모의운영 실무 협의 미팅 (2차)
2010-09-10	모의운영 착수
2010-09-13	1차 피드백
2010-09-14	2차 피드백
2010-09-16	모의운영 완료
2010-09-17	모의운영 결과 인터뷰

2) '나' 시스템

일시	작업
2010-07-23	관계자 미팅 및 현황 조사
2010-07-26	모의운영 진행 여부 협의
2010-08-10	모의운영 협조 요청 및 연계 기관 조사 협의
2010-08-25	연계 기관 조사 (시스템 '다')
2010-09-02	연계 기관 조사 (시스템 '라')
2010-09-13	모의운영 착수 ('나' ↔ '라')
2010-09-27	모의운영 계획 수립
2010-10-05	모의운영 결과 인터뷰 (예상 데이터 산출)

다. 모의운영 결과

본 프로젝트 수행 산출물로 개발된 변환 모듈과 적용 가이드를 전달하고, 이에 따라 변환 모듈을 시스템에 적용한 결과 아래와 같은 투입 리소스 및 시간을 확인할 수 있었다.

1) '가' 시스템

가) 시스템 유형

(1) A1B 유형에 해당하는 시스템을 기준으로 모의운영 수행

(2) '부분적 해결 방안 (1) – 변환모듈 사용' 과 '부분적 해결 방안 (2) – MS949 인코딩' 방식 적용[5)]

나) 적용 과정

(1) 내부 시스템 1

입력되는 확장한글 데이터를 MS949로 변환 저장 (변환 모듈 사용하지 않고 변환) :

• UTF-8 (브라우저) → UTF-8 (WAS) → MS949 (DB)

(2) 내부 시스템 2

EUC-KR DB에 맞도록 EUC-KR(+NCR)로 변환 저장하도록 조치된 결과 확인 :

• EUC-KR (브라우저) → EUC-KR (WAS) → EUC-KR(+NCR) (DB)

다) 투입 리소스 및 결과

(1) 특정 페이지에 변환 모듈 1회 적용 리소스

단계	소요 시간	비고
확장한글 문제의 이해 및 수행 계획 수립	–	미산정
분석/설계	1시간 30분	중급 개발자 (1명)
코드 수정 (변환 모듈 적용)	1시간 30분	〃
테스트 (QA/QC)	1시간 30분	〃
Total	**4시간 30분**	〃

(2) 모의운영 결과

내부시스템 1 : UTF-8 → MS949 변환 완료

내부시스템 2 : EUC-KR → EUC-KR(+NCR) 변환 동작 확인

5) 부분적 해결 방안 : '전환계획' 단원의 '유형별 전환계획' 참조

(3) 모의운영 수행기관 의견

(가) 상기 산출 내용은 특정 페이지에 변환 모듈 1회 적용에 대한 값으로, 실제 산출 시에는 전체 사용 단위를 계산하여야 한다.

(나) '가' 시스템의 모의운영 산출 값을 다른 시스템에 그대로 적용하기에는 무리가 있고, 시스템마다 사용 단위 및 환경이 다르므로 구체적인 사항은 해당 구축/유지보수 업체에 문의해서 확인해야 한다.

2) '나' 시스템

가) 시스템 유형

(1) ABC 유형 : '나' 시스템과 또 다른 '라' 시스템이 연계되는 유형의 시스템 구성

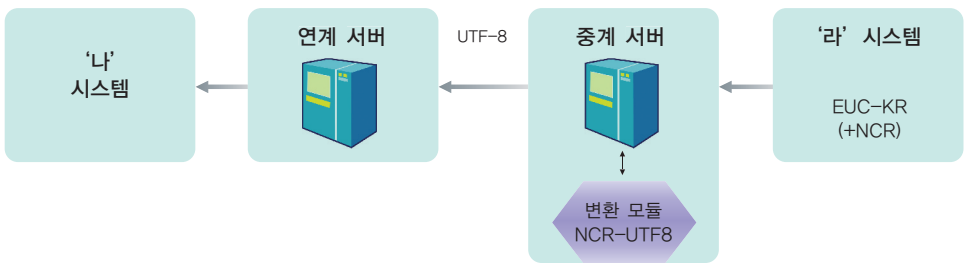
(2) '라' 시스템의 데이터가 'EUC-KR (+NCR)' 형태로 되어 있어, 이를 데이터 전송에 적합한 인코딩(UTF-8)으로 변환하는 과정 필요

나) 적용 과정 (예상 시나리오)

(1) '나' 시스템과 '라' 시스템을 연계하기 위해 양측에 연계 서버와 중계 서버를 구축한다.

(2) '라' 시스템에서 전송되는 데이터는 'EUC-KR (+NCR)'의 형태이므로, 중계 서버에서 변환 모듈을 사용하여 이를 UTF-8로 변환한다.

(3) UTF-8로 변환된 데이터를 연계 서버로 전송하여 '나' 시스템에서 활용한다.



다) 투입 리소스 및 결과

(1) 확장한글 문제 해결을 위해 총 3곳 수정

단계	작업량 (MM)	비고
분석/설계	0.5	중급 개발자 (1명)
코드 수정 (변환 모듈 적용)	0.5 (3곳)	중급 개발자 (1명)
DB 수정	1	〃
테스트 (QA/QC)	1	〃
Total	3 MM	〃

(2) 모의운영 결과

EUC-KR (+NCR) → UTF-8 변환

(3) 모의운영 수행기관 의견

(가) 상기 산출 내용은 변환 모듈 적용을 위해 계획한 예상 데이터로, 실제 적용 시 산출 값이 변경될 수 있다.

(나) 시스템마다 사용 단위 및 환경이 다르므로, 상기 산출 값은 작업 단위를 어느 정도 가늠할 수 있는 수준에서 참고해야 하며, 정확한 리소스 산출을 위해서는 구축/유지보수 업체에 문의해서 확인해야 한다.

2. 전환 비용 분석

확장한글 문제를 해결하기 위해서 시스템을 전환할 때 필요한 비용 요소들을 고려해본다.

가. 전환 비용 요소

시스템 전환을 세부 작업으로 구분하고, 모의운영 결과를 참고하여 부분별 전환 비용을 추정한다. 작업자의 수준에 따라 일정 및 비용에 차이가 있겠으나, '중급개발자 1인' 작업을 기준으로 비용 요소를 분석한다.

1) 응용프로그램 수정 비용 요소

비용 요소 기준표

구분	기간	비고
① 시스템 분석/설계		고정
• 확장한글 문제의 인식과 이해(학습) • 타겟 시스템에 대한 분석과 해결방안 선택 • 수행 계획 수립 (일정, 인원 포함)	10일	
② 개발 모듈 이해		고정
• 모듈 구조 및 API 이해 • 모듈 포팅 : 해당 시스템 플랫폼에 맞도록 모듈 코드 수정	2일	
③ 개발 작업		가변
• [A] 응용프로그램 수정 – 응용프로그램 분석 : 응용프로그램 입출력 부분 파악 – 응용프로그램 수정 : 응용프로그램 입출력 수정 및 모듈 적용 – 단위 테스트 : 작업 단위별 개발 테스트 ※ N_A : 응용프로그램 기능 (입출력) 수정 단위[6]	1일 * N_A	
• [C] 연계 수정 – 연계 입출력 API 분석 – 연계 입출력 API 수정 및 모듈 적용 – 단위 테스트 : 작업 단위별 개발 테스트 ※ N_C : 연계 입출력 API의 수정 단위 수	1일 * N_C	
④ 검증 (QA/QC)		가변
• 통합 테스트 : 전체 시스템 동작 구간에 대한 테스트 • 검증 기간(Q)은 개발 작업에 비례하여 산정. 예 50%	Q = ③ * 50%	
⑤ 반영		고정
• 실제 적용/운영	1일	

▶ N_A/N_C는 개발작업 각 구분에 따른 단위 작업의 개수이다.각 단위 작업은 유형에 따라 값이 다양하며 생략될 수도 있다.

▶ 본 산출식에서 '검증(QA/QC)' 기간은 일반적으로 '개발 작업' 시간 대비 50%로 산정하였다. 그러나 시스템의 영향 범위에 따라 실제 산출 값이 상이하므로, 실질적인 검증 시간은 해당 구축 및 유지보수 업체에 문의하여 개별 산정해야 한다.

▶ 실제로 시스템에 적용할 시에는, 시스템마다 구성과 복잡도가 다르기 때문에 해당 업체의 견적을 받아 산정하는 것이 타당하다.

2) DB 전환 비용 요소

DB 전환 비용 요소 기준표

구분	기간 (or 비용)	비고
⑥ DB 수정 작업		가변
• [B] DB 수정 → DB 마이그레이션 : DB 전체 데이터에 대한 인코딩 변경 (예 EUC-KR → UTF-8) – 전처리 : DB 작업용 시스템 구성 및 작업 설계– 수정 작업 : 인코딩 변경 작업 (batch 처리) → DB 클렌징 : 기존 DB에 잘못 저장된 확장한글 데이터를 정정 – 전처리 : 확장한글 문제 발생 패턴 수 분석 및 작업 설계 – 수정 작업 : 확장한글 복원 작업 (batch 처리) ※ 패턴으로 발견되지 않는 문제들은 이후 담당자가 개별 처리 → 고려사항 – N_B : DB 데이터 사이즈 (TB 단위) – DB 인코딩 설정 변경은 작업 기간 산정에 미포함 (예 EUC-KR → MS949) – DB 마이그레이션과 클렌징이 모두 필요한 경우, 두 가지 작업 기간을 모두 산정한다.	〈마이그레이션〉 전처리(5일) + 수정(1일 * N_B) 〈클렌징〉 전처리(5일) + 수정(1일 * N_B)	
⑦ DB 검증 (QA/QC)		가변
• DB 전환이 제대로 이루어졌는지 검증하는 절차 • 검증 기간(Q_B)은 개발 기간에 비례 산정 (예 50%)	Q_B = ⑥ * 50%	
⑧ 마이그레이션 작업용 스토리지 비용 (S)		가변
• 마이그레이션 과정에서 필요한 임시 작업 스토리지 비용 – 필요 작업 공간은 기존 용량의 두 배 – 실 증가 용량은 한글 포함 데이터 용량의 0.4배 • 스토리지 단가는 하이엔드 제품 기준, 20백만원/TB로 산정 – 정확한 가격과 운영 방안은 업체에 문의하여 산정 (임차 등)	필요 용량 = 작업 공간(N_B * 2) + 증가 용량(N_B * 0.4) S = 필요 용량 * 단가	

6) 입출력 기능 단위 : 하나의 데이터 레코드가 입력 또는 출력되는 기능을 단위 기준으로 한다.

나. 비용 산출식

비용 산출 시 활용할 수 있는 범용적인 계산 공식을 도출하고, 시스템 전환 유형에 따라 어떻게 적용하는지를 가상으로 적용시켜 본다.

1) 비용 산출 전제 조건

비용 요소들로부터 작업량을 산출하기 위하여 전제 조건을 아래와 같이 정의한다. 실제 비용 산출 시에는 아래 기준을 고려하여 상황에 맞게 재해석해야 한다. 스토리지 구매 비용, 라이센스 비용, 프로젝트 관리(PM) 비용 등 부대 비용은 이 산출식에 포함되지 않았으므로 별도로 고려하여야 한다.

가) 작업자 단위

(1) 한국소프트웨어산업협회에서 정하는 중급기술자 1인에 기준한다.
(2) 1인 기준으로 산출된 데이터이므로, 실제 산출 시 투입인원을 고려하여 재해석해야 한다.

나) 작업량 단위

(1) 기간의 합산 수치는 중급기술자의 총 작업량(맨먼스)[7]과 같은 수치를 갖는다.
(2) 각 요소별 산출 기간 기준이 상이하여도 (일 단위, 월 단위), 산출 결과는 월 단위로 환산한다.
(3) 일 단위에서 월 단위로 처리 시 환산 기준(월 평균근무일수)을 적용한다.
(4) 산출식에서 다루는 '기간'은 작업량 계산을 위한 수치이고, 실제 작업 기간과는 차이가 있을 수 있다.

다) 비용 단위

(1) 산출된 작업량을 기준으로 노임단가를 적용하여 소요 비용을 산출한다.(순수 맨먼스

7) 맨먼스(man-month, MM) : 일의 작업량을 나타내는 단위로, 평균적인 노동자 한 사람이 한 달 동안 해내는 일의 분량을 1MM이라 한다.

대비 비용이며, 제경비, 기술료 등은 별도로 계산해야 한다.)

(2) 산출식에서 다루는 '비용'은 전체 소요 비용을 가늠하기 위한 수치이고, 시스템에 따라 실제 작업 시 소요되는 비용과는 차이가 있을 수 있다.

2) 비용 산출 공식 유도

앞서 구분한 '전환 비용 요소'를 참고하여, 비용 산출을 위한 일반 공식을 도출한다.

가) 작업량

작업량을 도출하는 계산 형태는 아래와 같다.

작업량
① 시스템 분석/설계 + ② 개발 모듈 이해 + ③ 개발 작업 + ④ 검증(QA/QC) + ⑤ 반영

DB 전환 작업이 필요할 경우, 여기에 아래 작업이 추가된다.

DB 전환 작업량 (작업 절차)
⑥ DB 수정 작업 + ⑦ DB 검증 작업

나) 작업량 산출 공식

전환 기간에 고정 값과 가변 값을 적용시켜 수치화하면 아래와 같은 공식이 성립된다.

작업량(W_A) 산출 공식
작업일수(P_A) = 10일 + 2일 + D_A + Q_A + 1일 = 13일 + D_A + Q_A ※ 개발일수(D_A) = (1일 * N_A) + (1일 * N_C) ※ 검증일수(Q_A) = 개발일수(D_A) * 비례요율(50%) 작업량(WA) = 작업일수(P_A) / 1 month * 1 man = 작업일수(P_A) / (월 평균근무일수) * 1 man

DB 전환 작업량(W_B) 산출 공식
DB 작업일수(P_B) = DB수정일수(D_B) + 검증일수(Q_B) ※ DB수정일수(D_B) = (5일 + 1일 * NB) + (5일 + 1일 * N_B) ※ 검증일수(Q_B) = DB 수정일수 * 비례요율(50%) DB 작업량(WB) = DB 작업일수(P_B) / 1 month * 1 man = DB 작업일수(P_B) / (월 평균근무일수) * 1 man

다) 전환 비용 산출 공식

전환 기간을 표준 노임단가에 적용하여 전환 비용을 산출한다. 노임단가(Mc)는 'SW 기술자 노임단가' 또는 '엔지니어링 기술자 노임단가' 기준을 참고한다.

작업량

전환 비용(C) 산출 공식전환 비용(C) = 응용프로그램 수정 비용(C_A) + DB 전환 비용(C_B)
※ 응용프로그램 수정 비용(C_A) = 작업량(W_A) * 노임단가(M_C)
※ DB 전환 비용(C_B) = 작업량(W_B) * 노임단가(M_C) + 마이그레이션 작업 스토리지 비용(S)

여기서, 스토리지 비용(S) = {작업 공간(N_B * 2) + 증가 용량(N_B * 0.4)} * 스토리지 단가
= N_B * 2.4 * 스토리지 단가

참 고

한국소프트웨어산업협회
"2010년도 적용 SW기술자 노임단가"
(참고 : http://www.sw.or.kr/intro/view.asp?masteridx=1&idx=4813)

기술 등급	일 노임단가	월 노임단가 (한 달 21.6일 기준)	기술자격자
기술사	358,777원	7,749,583원	• 기술사
특급기술자	333,226원	7,197,682원	• 고급기술자 후 3년 업무
고급기술자	239,085원	5,164,236원	• 중급기술자 후 3년 업무 • 박사로서 기사 또는 공인민간자격 취득
중급기술자	188,139원	4,063,802원	• 기사 후 3년 또는 산업기사 후 7년 업무 • 공인민간자격 후 3년 업무 • 기사/공인민간자격으로 석사 후 2년 업무
초급기술자	146,620원	3,166,992원	• 기사, 산업기사, 또는 공인민간자격 취득

참 고

한국엔지니어링협회
"2010년도 적용 엔지니어링 기술자 노임단가 (건설 및 기타부분)"
(참고 : http://www.kenca.or.kr/2009/cyber/pen/wages.jsp)

인력 등급	일 노임단가	월 노임단가 (한 달 22.24일 기준)	자격 기준(학력 기준)
기술사	320,277원	7,122,960원	기술사
특급기술자	258,303원	5,744,658원	박사 3년 이상, 석사 9년 이상, 학사 12년 이상, 전문대졸 15년 이상
고급기술자	203,277원	4,520,880원	박사, 석사 6년 이상, 학사 9년 이상, 전문대졸 12년 이상, 고졸 15년 이상
중급기술자	174,482원	3,880,479원	석사 3년 이상, 학사 6년 이상, 전문대졸 9년 이상, 고졸 12년 이상
초급기술자	127,396원	2,833,287원	석사, 학사, 전문대졸, 고졸 3년 이상

3) 유형별 비용 산출의 구성

수준별 전환계획의 구성에 따라 전환할 수 있는 유형을 살펴보면 아래와 같이 구분할 수 있다.

구분		내용	비고
임시 대응		웹 브라우저를 활용한 임시 대응	비용 산출 제외
부분적 전환	A 유형	응용프로그램 수정을 통한 부분 대응	
	B 유형	DB 수정을 통한 부분 대응	
	C 유형	연계 수정을 통한 부분 대응	
전면적 전환		A,B,C 모든 단계를 수정하여 전면 대응	
신규 구축		신규 시스템 구성 시 확장한글을 고려하여 구축	

가) 임시 대응

(1) 웹 브라우저를 활용한 임시적인 대응 방법으로 별도의 비용이 발생하지 않는다. (비용 산출 대상에서 제외)

(2) 웹 브라우저에 따라 대응 방법이 다를 수 있으며, 지원되지 않는 웹 브라우저도 존재한다.

(3) 사용자가 수동으로 대응해야 하는 부분이 부수적으로 존재한다.

(4) 웹 브라우저를 사용하지 않는 C/S 환경에서는 해당 유형을 적용하기 어렵다.

나) 부분적 전환

(1) 수정이 필요한 각 요소에 따라 유형을 나누어 구분할 수 있다.

A 유형 : 응용프로그램 부분 수정

B 유형 : DB 부분 수정

C 유형 : 연계 부분 수정

(2) 시스템 환경에 따라 부분적 수정을 중복으로 적용해야 하는 경우도 있다.

예 AB, AC, BC 유형

(3) 시스템의 규모(size)에 따라 비용 산출 값에 차이가 나타나는 항목이 있으므로, 해당 항목은 변수로 처리해야 한다.

(4) 부분적 수정으로 모든 유형을 다 적용해야 하는 경우, 전면적 전환으로 간주하여 구분한다.

예 ABC 유형

다) 전면적 전환

(1) 전환 대상 시스템의 모든 요소를 수정해야 하는 경우에 해당한다.
(2) 모든 유형을 유니코드(또는 UTF-8)로 처리하도록 수정하는 경우에 해당한다.
(3) 부분적 전환의 최대치(ABC) 유형과 동일하게 간주한다.

라) 신규 구축

(1) 시스템 신규 구축 시 적용되는 항목이다.
(2) 신규 구축 시에는 모든 부분에서 유니코드(또는 UTF-8)을 사용하도록 한다.
(3) 시스템 구축에 소요되는 비용 이외에 확장한글 지원을 위한 별도의 추가 비용이 소요되지 않는다.
(4) DB 스토리지 구성 시 UTF-8 인코딩 타입을 고려하여 저장용량을 기존 보다 다소 높게 구성해야 한다. (이론상 최대 1.5배이지만, 실제 데이터는 한글, 숫자, 영문 등이 혼합 사용되므로 이보다 낮다.)

4) 비용 산출 절차

비용 산출을 진행하는 절차를 기술한다.

비용 요소 항목표

구분	기간	비고
① 시스템 분석/설계	10 일	고정
② 개발 모듈 이해	2 일	고정
③ 개발 작업	- 일	
[A] 응용프로그램 수정	- 일	가변
[C] 연계 수정	- 일	
④ 검증 (QA/QC)	- 일	가변
⑤ 반영	1 일	고정

DB 전환 비용 요소 항목표

구분	기간	비고
⑥ DB 수정 작업	– 일	
[B] DB 수정	– 일	가변
[C] 연계 수정	– 일	
⑦ DB 검증 (QA/QC)	– 일	가변
⑧ 마이그레이션 작업 필요 용량	– TB	가변

가) 개발 작업의 범위와 단위를 파악한다.

유형에 따라 개발 범위에 해당하는 작업을 파악하고 수치화한다.

나) 검증에 소요되는 기간을 파악한다.

검증 기간을 개발 작업의 50%로 산정하고 있으나, 시스템의 크기나 수정 범위에 따라 검증에 소요되는 기간이 상이하므로, 구체적인 사항은 구축 및 유지보수 업체에 문의하여 적용한다.

다) 작업량 산출 공식에 적용한다.

앞서 파악된 비용 요소들을 취합하여 작업량을 산출한다. (월 기준, MM)

라) 전환 비용을 산출한다.

산출된 작업량에 노임단가를 적용하여 전환 비용을 산출한다.
이렇게 산출된 전환 비용은 예상으로 입력한 작업량에 따라 산정된 가상 비용으로, 실제 작업량과 총 소요비용은 업체에 문의하여 산출한다.

다. 유형별 비용 산출

각 유형별 전환 비용을 가상으로 산출해 본다.
(유형 분류 기준은 '확장한글 문제 유형 분류' 단원을 참고한다.)

1) Type A

응용프로그램을 수정해야 하는 유형으로, 응용프로그램의 입출력 기능 단위 수에 따라 비용 산출식을 적용한다.

가) 시나리오

전라북도 ○○군청의 민원서비스 웹사이트에서 확장한글 문제가 발생하여 해당 시스템을 자체 분석해본 결과, 응용프로그램에서 수정해야할 '입출력 기능 단위'가 10개이고, 테스트는 개발 작업 기간 대비 50%가 소요된다고 할 때, 예상 비용은 다음과 같이 산출해 볼 수 있다.

나) 비용 요소 항목

Type A 비용 요소 적용표

구분	기간	비고
① 시스템 분석/설계	10 일	고정
② 개발 모듈 이해	2 일	고정
③ 개발 작업	10 일	가변
[A] 응용프로그램 수정	10 일	
[C] 연계 수정	– 일	
④ 검증 (QA/QC)	5 일	가변 (50%)
⑤ 반영	1 일	고정

다) 비용 산출

작업량(W) 산출 공식

W_A = ① 시스템 분석/설계 + ② 개발 모듈 이해 + ③ 개발 작업 + ④ 검증(QA/QC) + ⑤ 반영

P_A	=	①	+	②	+	③A	+	③C	+	④Q	+	⑤
P_A	=	10	+	2	+	1 * N_A	+	1 * N_C	+	③ * 50%	+	1
		고정		고정		가변		가변		가변		고정
28.0	=	10	+	2	+	10	+	0	+	5	+	1

W_A	=	P_A	/	1 Month	*	1 Man
1.30 MM	=	28.0	/	21.6 일	*	1 명

전환 비용(C) 산출 공식

전환 비용(C_A) = 작업량(W_A) * 노임단가(M_C)

C_A	=	W_A	*	M_C
5,267,892원	=	1.30	*	4,063,852명

〈참고〉 2010년 중급 SW기술자 노임단가(MC) = 188,139원/日, 4,063,802원/月.

또는 엔지니어링 기술자 노임단가 적용 시,
작업량(WA) = 28.0 / 22.24일 * 1명 = 1.26 MM.
전환비용(CA) = 1.26 * 3,880,480원 = 4,885,496원

2) Type B

DB를 수정해야 하는 유형으로, DB의 수정을 위한 작업을 파악하고 비용 산출식을 적용한다.(DB 마이그레이션과 DB 클렌징의 작업 유무 및 데이터 사이즈 파악)

가) 시나리오

경상남도 ○○군청의 민원서비스 웹사이트에서 확장한글 문제가 발생하여 해당 시스템을 자체 분석해본 결과, DB의 마이그레이션과 클렌징 작업이 필요하고 데이터 크기는 약 10TB라고 한다. 그리고 테스트는 개발 작업 기간 대비 50%가 소요된다고 할 때, 예상 비용은 다음과 같이 산출해 볼 수 있다.

나) 비용 요소 항목

Type B의 비용 요소 항목표

구분	기간	비고
① 시스템 분석/설계	10 일	고정
② 개발 모듈 이해	2 일	고정
③ 개발 작업	0 일	가변
[A] 응용프로그램 수정	– 일	
[C] 연계 수정	– 일	
④ 검증 (QA/QC)	0 일	가변 (50%)
⑤ 반영	1 일	고정

Type B의 DB 전환 비용 요소 항목표

구분	기간	비고
⑥ DB 수정 작업	30 일	
[B] DB 마이그레이션	15 일	가변
[B] DB 클렌징	15 일	
⑦ DB 검증 (QA/QC)	15 일	가변 (50%)
⑧ 마이그레이션 작업 필요 용량	24 TB	가변

다) 비용 산출

작업량(W) 산출 공식

W_A = ① 시스템 분석/설계 + ② 개발 모듈 이해 + ⑥ DB 수정 작업 + ⑦ DB 검증(QA/QC) + ⑤ 반영

P	=	①	+	②	+	⑥B	+	⑦Q	+	⑤
P	=	10	+	2	+	$5+1*N_B+5+1*N_B$	+	⑥*50%	+	1
		고정		고정		가변		가변		고정
58.0	=	10	+	2	+	15	+	15	+	15	+	1

W	=	P	/	1 Month	*	1 Man
2.69 MM	=	58.0	/	21.6 일	*	1 명

전환 비용(C) 산출 공식

전환 비용(C) = 작업량(W) * 노임단가(M_C) + DB 마이그레이션 작업 스토리지 비용(S)

C	=	W	*	M_C	+	⑧S
	=	2.69	*	4,063,802원	+	(10*2.4*20M원)
	=	**10,912,062원**				**(480 백만원)**

〈참고〉 2010년 중급 SW기술자 노임단가(MC) = 188,139원/日, 4,063,802원/月.
스토리지 단가 = 20M원/TB

또는 엔지니어링 기술자 노임단가 적용 시,
작업량(W) = 58.0 / 22.24일 * 1명 = 2.61 MM.
전환비용(C) = 2.61 * 3,880,480원 + (S) = 10,119,956원 + (480백만원)

3) Type C

시스템 간의 연계 부분을 수정해야 하는 유형으로, 연계 API의 수정을 위한 작업을 파악하고 비용 산출식을 적용한다. (연계되는 시스템들 중 현재 작업하고자 하는 시스템을 기준으로 한다.)

가) 시나리오

충청남도 ○○군청의 민원서비스 웹사이트에서 확장한글 문제가 발생하여 해당 시스템을 자체 분석해본 결과, 연계되는 중앙부처 시스템과의 데이터 송수신 부분에서 5개의 수정작업이 필요하고, 테스트는 개발 작업 기간 대비 80%가 소요된다고 할 때, 예상 비용은 다음과 같이 산출해 볼 수 있다.

나) 비용 요소 항목

Type C 비용 요소 항목표

구분	기간	비고
① 시스템 분석/설계	10 일	고정
② 개발 모듈 이해	2 일	고정
③ 개발 작업	10 일	
[A] 응용프로그램 수정	– 일	가변
[C] 연계 수정	5 일	
④ 검증 (QA/QC)	4 일	가변 (80%)
⑤ 반영	1 일	고정

다) 비용 산출

작업량(W) 산출 공식

W_A = ① 시스템 분석/설계 + ② 개발 모듈 이해 + ③ 개발 작업 + ④ 검증(QA/QC) + ⑤ 반영

P_A	=	①	+	②	+	③A	+	③C	+	④Q	+	⑤
P_A	=	10	+	2	+	$1 * N_A$	+	$1 * N_C$	+	③ * 80%	+	1
		고정		고정		가변		가변		가변		고정
22.0	=	10	+	2	+	0	+	5	+	4	+	1

W_A	=	P_A	/	1 Month	*	1 Man
1.02 MM	=	22	/	21.6 일	*	1 명

전환 비용(C) 산출 공식

전환 비용(C_A) = 작업량(W_A) * 노임단가(M_C)

C_A	=	W_A	*	M_C
4,139,058원	=	1.02	*	4,063,802원

〈참고〉 2010년 중급 SW기술자 노임단가(M_C) = 188,139원/日, 4,063,802원/月.

또는 엔지니어링 기술자 노임단가 적용 시,
작업량(W_A) = 22.0 / 22.24일 * 1명 = 0.99 MM.
전환비용(C_A) = 0.99 * 3,880,480원 = 3,838,604원

4) Type AB

응용프로그램과 DB를 수정해야 하는 유형으로, 응용프로그램의 입출력 기능 단위 수 및 DB의 수정을 위한 작업을 파악하고 비용 산출식을 적용한다.

가) 시나리오

경기도 ○○군청의 민원서비스 웹사이트에서 확장한글 문제가 발생하여 해당 시스템을 자체 분석해본 결과, 응용프로그램에서는 수정해야할 '입출력 기능 단위'가 10개이고, DB는 클렌징 작업만이 필요하고 데이터 크기는 약 10TB라고 한다. 그리고 테스트는 개발 작업 기간 대비 50%가 소요된다고 할 때, 예상 비용은 다음과 같이 산출해 볼 수 있다.

나) 비용 요소 항목

Type AB 비용 요소 적용표

구분	기간	비고
① 시스템 분석/설계	10 일	고정
② 개발 모듈 이해	2 일	고정
③ 개발 작업	10 일	
[A] 응용프로그램 수정	10 일	가변
[C] 연계 수정	– 일	
④ 검증 (QA/QC)	5 일	가변 (50%)
⑤ 반영	1 일	고정

Type AB의 DB 전환 비용 요소 항목표

구분	기간	비고
⑥ DB 수정 작업	15 일	
[B] DB 마이그레이션	– 일	가변
[B] DB 클렌징	15 일	
⑦ DB 검증 (QA/QC)	7.5 일	가변 (50%)
⑧ 마이그레이션 작업 필요 용량	– TB	가변

다) 비용 산출

작업량(W) 산출 공식

W_A = ① 시스템 분석/설계 + ② 개발 모듈 이해 + ③ 개발 작업 + ④ 검증(QA/QC) + ⑤ 반영

P_A	=	①	+	②	+	③A	+	③C	+	④Q	+	⑤
P_A	=	10	+	2	+	$1*N_A$	+	$1*N_C$	+	③*50%	+	1
		고정		고정		가변		가변		가변		고정
28.0	=	10	+	2	+	10	+	0	+	5	+	1

W_A	=	P_A	/	1 Month	*	1 Man
1.30 MM	=	28.0	/	21.6 일	*	1 명

W_B = ⑥ DB 수정 작업 + ⑦ DB 검증 작업

P_B	=	⑥B	+	⑦Q
P_B	=	$5 + 1*N_B + 5 + 1*N_B$	+	⑥*50%
		가변		가변
58.0	=	0 + 15	+	7.5

W_B	=	P_B	/	1 Month	*	1 Man
1.04 MM	=	22.5	/	21.6 일	*	1 명

전환 비용(C) 산출 공식

응용프로그램 수정 비용(C_A) = 수정 작업량(W_A) * 노임단가(M_C)

C_A	=	W_A	*	M_C
5,267,892원	=	1.30	*	4,063,802원

DB 전환 비용(C_B) = DB 작업량(W_B) * 노임단가(M_C) + 마이그레이션 작업 스토리지 비용(S)

C_B	=	W_B	*	M_C	+	⑧S
4,233,128원	=	1.04	*	4,063,802원	+	0

〈참고〉 2010년 중급 SW기술자 노임단가(M_C) = 188,139원/日, 4,063,802원/月.

또는 엔지니어링 기술자 노임단가 적용 시,
작업량(W_A) = 28.0 / 22.24일 * 1명 = 1.26 MM,
작업량(W_B) = 22.5 / 22.24일 * 1명 = 1.01 MM,
전환비용(C_A) = 1.26 * 3,880,480원 = 4,889,405원,
전환비용(C_B) = 1.01 * 3,880,480원 = 3,919,285원.

5) Type AC

응용프로그램과 시스템 간의 연계 부분을 수정해야 하는 유형으로, 응용프로그램의 입출력 기능 단위 수 및 연계 API의 수정을 위한 작업을 파악하고 비용 산출식을 적용한다. (연계되는 시스템들 중 현재 작업하고자 하는 시스템을 기준으로 한다.)

가) 시나리오

충청북도 ○○군청의 민원서비스 웹사이트에서 확장한글 문제가 발생하여 해당 시스템을 자체 분석해본 결과, 응용프로그램에서는 수정해야할 '입출력 기능 단위'가 10개이고, 연계되는 중앙부처 시스템과의 데이터 송수신 부분에서 5개의 수정작업이 필요하고, 테스트는 개발 작업 기간 대비 80%가 소요된다고 할 때, 예상 비용은 다음과 같이 산출해 볼 수 있다.

나) 비용 요소 항목

Type AC 비용 요소 적용표

구분	기간	비고
① 시스템 분석/설계	10 일	고정
② 개발 모듈 이해	2 일	고정
③ 개발 작업	15 일	가변
[A] 응용프로그램 수정	10 일	
[C] 연계 수정	5 일	
④ 검증 (QA/QC)	12 일	가변 (80%)
⑤ 반영	1 일	고정

다) 비용 산출

작업량(W) 산출 공식

W_A = ① 시스템 분석/설계 + ② 개발 모듈 이해 + ③ 개발 작업 + ④ 검증(QA/QC) + ⑤ 반영

P_A	=	①	+	②	+	③A	+	③C	+	④Q	+	⑤
P_A	=	10	+	2	+	$1 * N_A$	+	$1 * N_C$	+	③ * 80%	+	1
		고정		고정		가변		가변		가변		고정
40.0	=	10	+	2	+	10	+	5	+	12	+	1

W_A	=	P_A	/	1 Month	*	1 Man
1.85 MM	=	40	/	21.6 일	*	1 명

전환 비용(C) 산출 공식

전환 비용(C_A) = 작업량(W) * 노임단가(M_C)

C_A	=	W_A	*	M_C
7,525,560원	=	1.85	*	4,063,802원

〈참고〉 2010년 중급 SW기술자 노임단가(M_C) = 188,139원/日, 4,063,802원/月.

또는 엔지니어링 기술자 노임단가 적용 시,
작업량(W_A) = 40 / 22.24일 * 1명 = 1.80 MM
전환비용(C_A) = 1.80 * 3,880,480원 = 6,979,280원

6) Type BC

DB와 시스템 간의 연계 부분을 수정해야 하는 유형으로, DB의 수정을 위한 작업 및 연계 API의 수정을 위한 작업을 파악하고 비용 산출식을 적용한다.

가) 시나리오

경상북도 ○○군청의 민원서비스 웹사이트에서 확장한글 문제가 발생하여 해당 시스템을 자체 분석해본 결과, DB는 클렌징 작업만이 필요하고 데이터 크기는 약 5TB라고 한다. 그리고 연계되는 중앙부처 시스템과의 데이터 송수신 부분에서 8개의 수정작업이 필요하고, 테스트는 개발 작업 기간 대비 50%가 소요된다고 할 때, 예상 비용은 다음과 같이 산출해 볼 수 있다.

나) 비용 요소 항목

Type BC 비용 요소 적용표

구분	기간	비고
① 시스템 분석/설계	10 일	고정
② 개발 모듈 이해	2 일	고정
③ 개발 작업	8 일	
[A] 응용프로그램 수정	– 일	가변
[C] 연계 수정	8 일	
④ 검증 (QA/QC)	4 일	가변 (50%)
⑤ 반영	1 일	고정

Type BC의 DB 전환 비용 요소 항목표

구분	기간	비고
⑥ DB 수정 작업	10 일	
[B] DB 마이그레이션	– 일	가변
[C] DB 클렌징	10 일	
⑦ DB 검증 (QA/QC)	5 일	가변 (50%)
⑧ 마이그레이션 작업 필요 용량	– TB	가변

다) 비용 산출

작업량(W) 산출 공식

W_A = ① 시스템 분석/설계 + ② 개발 모듈 이해 + ③ 개발 작업 + ④ 검증(QA/QC) + ⑤ 반영

P_A	=	①	+	②	+	③A	+	③C	+	④Q	+	⑤
P_A	=	10	+	2	+	$1*N_A$	+	$1*N_C$	+	③*50%	+	1
		고정		고정		가변		가변		가변		고정
25.0	=	10	+	2	+	0	+	8	+	4	+	1

W_A	=	P_A	/	1 Month	*	1 Man
1.16 MM	=	25	/	21.6 일	*	1 명

W_B = ⑥ DB 수정 작업 + ⑦ DB 검증 작업

P_B	=	⑥B	+	⑦Q
P_B	=	$5 + 1*N_B + 5 + 1*N_B$	+	⑥*50%
		가변		가변
15.0	=	0	+	15	+	5.0

W_B	=	P_B	/	1 Month	*	1 Man
0.69 MM	=	15.0	/	21.6 일	*	1 명

전환 비용(C) 산출 공식

응용프로그램 수정 비용(C_A) = 수정 작업량(W_A) * 노임단가(M_C)

C_A	=	W_A	*	M_C
4,703,475원	=	1.16	*	4,063,802원

DB 전환 비용(C_B) = DB 작업량(W_B) * 노임단가(M_C) + 마이그레이션 작업 스토리지 비용(S)

C_B	=	W_B	*	M_C	+	⑧S
2,822,085원	=	0.69	*	4,063,802원	+	0

〈참고〉 2010년 중급 SW기술자 노임단가(M_C) = 188,139원/日, 4,063,802원/月.

또는 엔지니어링 기술자 노임단가 적용 시,
작업량(W_A) = 25 / 22.24일 * 1명 = 1.12 MM,
작업량(W_B) = 15 / 22.24일 * 1명 = 0.67 MM,
전환비용(C_A) = 1.12 * 3,880,480원 = 4,346,138원,
전환비용(C_B) = 0.67 * 3,880,480원 = 2,599,922원.

7) 전면적 전환

전면적 전환은 응용프로그램, DB, 연계의 각 부분들 중에서 유니코드(또는 UTF-8)를 사용하지 않는 부분을 모두 유니코드(또는 UTF-8)로 전환하는 경우이다.

가) 시나리오

강원도 ○○군청의 민원서비스 웹사이트에서 확장한글 문제가 발생하여 해당 시스템을 자체 분석해본 결과, 응용프로그램/DB/연계 등 모든 구성에서 확장한글을 지원하지 않는 문자 인코딩(EUC-KR)을 사용하고 있어 전면적 전환이 필요하다.

- 응용프로그램 : 유니코드(또는 UTF-8) 입출력 기능 단위 수는 30개이다.
- DB : DB 마이그레이션 및 클렌징 작업이 필요하고, 데이터 크기는 약 5TB이다.
- 연계 : 기존에 다른 시스템과 연계하는 방식을 유지하기 위한 수정작업은 6개이다.
- 테스트 기간은 업체 문의 결과 개발 작업 기간 대비 60%가 소요된다고 한다.

나) 비용 요소 항목

Type ABC 비용 요소 적용표

구분	기간	비고
① 시스템 분석/설계	10 일	고정
② 개발 모듈 이해	2 일	고정
③ 개발 작업	36 일	
[A] 응용프로그램 수정	30 일	가변
[C] 연계 수정	6 일	
④ 검증 (QA/QC)	21.6 일	가변 (60%)
⑤ 반영	1 일	고정

Type ABC의 DB 전환 비용 요소 항목표

구분	기간	비고
⑥ DB 수정 작업	20 일	
[B] DB 마이그레이션	10 일	가변
[C] DB 클렌징	10 일	
⑦ DB 검증 (QA/QC)	12 일	가변 (60%)
⑧ 마이그레이션 작업 필요 용량	12 TB	가변

다) 비용 산출

작업량(W) 산출 공식

W_A = ① 시스템 분석/설계 + ② 개발 모듈 이해 + ③ 개발 작업 + ④ 검증(QA/QC) + ⑤ 반영

P_A	=	①	+	②	+	③A	+	③C	+	④Q	+	⑤
P_A	=	10	+	2	+	1 * N_A	+	1 * N_C	+	③ * 60%	+	1
		고정		고정		가변		가변		가변		고정
70.6	=	10	+	2	+	30	+	6	+	21.6	+	1

W_A	=	P_A	/	1 Month	*	1 Man
3.27 MM	=	70.6	/	21.6 일	*	1 명

W_B = ⑥ DB 수정 작업 + ⑦ DB 검증 작업

P_B	=	⑥B	+	⑦Q
P_B	=	5 + 1 * N_B + 5 + 1 * N_B	+	⑥ * 60%
		가변		가변
32.0	=	10 + 10	+	12

W_B	=	P_B	/	1 Month	*	1 Man
1.48 MM	=	32.0	/	21.6 일	*	1 명

전환 비용(C) 산출 공식

응용프로그램 수정 비용(C_A) = 수정 작업량(W_A) * 노임단가(M_C)

C_A	=	W_A	*	M_C
13,282,613원	=	3.27	*	4,063,802원

DB 전환 비용(C_B) = DB 작업량(W_B) * 노임단가(M_C) + 마이그레이션 작업 스토리지 비용(S)

C_B	=	W_B	*	M_C	+	⑧S
	=	0.69	*	4,063,802원	+	(5 * 2.4 * 20M원)
	=	**6,020,448원**			+	**(240 백만원)**

〈참고〉 2010년 중급 SW기술자 노임단가(M_C) = 188,139원/日, 4,063,802원/月
스토리지 단가 = 20M원/TB

또는 엔지니어링 기술자 노임단가 적용 시,
작업량(W_A) = 70.6 / 22.24일 * 1명 = 3.17 MM,
작업량(W_B) = 32.0 / 22.24일 * 1명 = 1.44 MM,
전환 비용(C_A) = 3.17 * 3,880,480원 = 12,301,121원,
전환 비용(C_B) = 1.48 * 3,880,480원 + (S) = 5,743,110원 + (240 백만원)

첨부 자료

첨부 1 DB별 MS949 문자 인코딩 지원 여부 확인

국내에서 많이 사용되는 대표적인 DBMS들을 대상으로 MS949 지원 여부를 확인한다.

가. Oracle Database

Oracle은 미국 오라클(ORACLE)사의 관계형 데이터베이스 관리 시스템의 이름으로, UNIX 및 Linux, Windows 등의 운영체제 환경을 모두 지원하는 RDBMS이다. 검색이나 업데이트용 언어로는 국제표준화기구의 표준 구조화 조회 언어와 PL/SQL을 지원한다. 국내의 많은 기업 및 기관에서 Oracle Database를 사용하고 있으며, 현재(2010년 6월) 'Oracle Database 11g Release 2 (11.2.0.1.0)' 버전까지 출시되어 있다.

1) ORACLE에서 지원하는 한글 문자 인코딩 종류

ORACLE 인코딩 명칭	구분
KO16KSC5601	EUC-KR
KO16MSWIN949	MS949
UTF8	UTF-8
AL32UTF8	UTF-8 (권장)

2) 한글 문자 인코딩 지원 ORACLE 버전

Oracle Technology Network 참조

	KO16KSC5601	KO16MSWIN949	UTF8	AL32UTF8
한글 지원상태	한글완성형 2350자	KO16KSC5601 + 확장 8822자 (총 11172자)	한글 11172자	한글 11172자
한글바이트	2바이트	2바이트	3바이트	3바이트
지원버전	7.x	8.0.6 이상	8.0 이후	9i Release1 이상

http://www.oracle.com/technology/global/kr/pub/columns/oracle_nls_1.html#mozTocId563482)

3) MS949 지원 버전

상기 표에서와 같이 MS949 문자 인코딩은 Oracle 8.0.6 이상부터 지원한다.

나. Microsoft SQL Server

Microsoft SQL Server는 Microsoft가 1993년 Sybase를 기반으로 개발한 관계형 데이터베이스로, 현재(2010년 6월) 'Microsoft SQL Server 2008 R2' 버전까지 출시되어 있다.
Microsoft Windows 상에서만 동작하는 DBMS로, Microsoft 자사의 문자 인코딩인 MS949를 기본 지원한다.

다. MySQL

MySQL은 멀티스레드, 멀티유저를 지원하는 구조질의어 형식의 데이터베이스 관리 시스템(DBMS)으로, 현재 (2010년 6월) 'MySQL Enterprise Server 5.1' 버전까지 출시되어 있다.

1) 문자 인코딩 확인 명령

MySQL은 컴파일 시 지정하는 문자 인코딩 설정에 따라 지원 환경이 다를 수 있으므로, 아래와 같이 지원되는 문자 인코딩 리스트를 확인하는 명령을 사용하여 찾아본다.

mysql> SHOW CHARACTER SET;

```
mysql> SHOW CHARACTER SET;
+----------+-----------------------------+---------------------+--------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+--------+
| dec8     | DEC West European           | dec8_swedish_ci     |      1 |
| cp850    | DOS West European           | cp850_general_ci    |      1 |
| hp8      | HP West European            | hp8_english_ci      |      1 |
| koi8r    | KOI8-R Relcom Russian       | koi8r_general_ci    |      1 |
| latin1   | ISO 8859-1 West European    | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |      1 |
| ascii    | US ASCII                    | ascii_general_ci    |      1 |
| hebrew   | ISO 8859-8 Hebrew           | hebrew_general_ci   |      1 |
| euckr    | EUC-KR Korean               | euckr               |      1 |
| koi8u    | KOI8-U Ukrainian            | koi8u_general_ci    |      1 |
| greek    | ISO 8859-7 Greek            | greek_general_ci    |      1 |
| cp1250   | Windows Central European    | cp1250_general_ci   |      1 |
```

```
| latin5 | ISO 8859-9 Turkish | latin5_turkish_ci | 1 |
| armscii8 | ARMSCII-8 Armenian | armscii8_general_ci | 1 |
| utf8 | UTF-8 Unicode | utf8_general_ci | 3 |
| cp866 | DOS Russian | cp866_general_ci | 1 |
| keybcs2 | DOS Kamenicky Czech-Slovak | keybcs2_general_ci | 1 |
| macce | Mac Central European | macce_general_ci | 1 |
| macroman | Mac West European | macroman_general_ci | 1 |
| cp852 | DOS Central European | cp852_general_ci | 1 |
| latin7 | ISO 8859-13 Baltic | latin7_general_ci | 1 |
| cp1256 | Windows Arabic | cp1256_general_ci | 1 |
| cp1257 | Windows Baltic | cp1257_general_ci | 1 |
| binary | Binary pseudo charset | binary | 1 |
| geostd8 | GEOSTD8 Georgian | geostd8_general_ci | 1 |
+----------+-----------------------------+--------------------+--------+
```

2) 문자 인코딩 표기

MySQL 5.1.38 이상의 버전에서는 EUC-KR 인코딩에 확장형(MS949) 인코딩을 포함한다.

- 참고 URL : Changes in MySQL 5.1.38 (01 September 2009)
  (http://dev.mysql.com/doc/refman/5.1/en/news-5-1-38.html)

3) MySQL 리빌드

만약 MySQL의 버전이 5.1.38 이상이면, '2)' 항과 같이 EUC-KR에서 MS949 문자 인코딩을 지원할 수 있게 된다. 그러나 지원 '문자 인코딩 확인 명령' 사용 시 EUC-KR이 나오지 않는다면, 아래 옵션을 사용하여 mysql을 리빌드하고, 지원 목록을 다시 살펴봐야 한다.

```
--with-extra-charsets=all
```

첨부 2 문자 인코딩 종류에 따른 데이터 용량 비교

정보시스템에서 확장한글 해결을 위해 문자 인코딩 변경 시, 발생할 수 있는 스토리지 증설 여부를 확인하기 위한 참고 자료로서, 주요 문자 인코딩인 EUC-KR과 UTF-8을 기준으로 특정 데이터의 용량을 비교한다.

가. 사용자 입력 시의 데이터 크기 비교

1) 코드 글자별 byte 할당 수

구분	EUC-KR	UTF-8
숫자	1	1
영문자	1	1
한글	2	3
한자	2	3

2) 비교 분석

가) 데이터 입력 페이지(예 민원24 회원가입 페이지)

나) 코드 값 (byte value)

구분	데이터	EUC-KR	UTF-8
성명	홍길동	홍 길 동 C8 AB B1 E6 B5 BF	홍 길 동 ED 99 8D EA B8 B8 EB 8F 99
주민등록 번호(전)	701010	7 0 1 0 1 0 37 30 31 30 31 30	7 0 1 0 1 0 37 30 31 30 31 30
주민등록 번호(후)	1234567	1 2 3 4 5 6 7 31 32 33 34 35 36 37	1 2 3 4 5 6 7 31 32 33 34 35 36 37
입력확인	MINW	N I N W 4D 49 4E 57	N I N W 4D 49 4E 57

다) 데이터 할당 바이트 수

구분	데이터	Type	EUC-KR	UTF-8	비고
성명	홍길동	한글	6 byte	9 byte	+3
주민등록번호(전)	701010	숫자	6 byte	6 byte	0
주민등록번호(후)	1234567	숫자	7 byte	7 byte	0
입력확인	MINW	영문	4 byte	4 byte	0
합계			23 byte	26 byte	+3

라) 산출 내용 분석

(1) 한글과 한자 등의 문자 코드 입력 시, EUC-KR은 2byte를 할당하지만, UTF-8은 3byte를 할당한다.

(2) 숫자, 영문자 등은 EUC-KR과 UTF-8 모두 같은 1byte를 사용한다.

(3) 이 페이지의 경우 23byte의 데이터가 26byte로 3byte 증가되었다.

3) 결과

(1) EUC-KR로 구성된 DB 시스템을 UTF-8로 전환 시,한글 및 한자 등이 포함된 글자 수 만큼 byte 수가 늘어난다. (1.5배)

(2) 모든 데이터가 한글 및 한자로 구성되어 있을 시,최대 1.5배(150%)까지 저장 공간이 필요할 수 있다.

(3) 전체 데이터 중 한글 및 한자 데이터의 비율이 저장 공간 계산에 중요한 요소이다. DB 필드 속성(용도) 및 할당량(byte)을 분석하여 적합한 비율을 계산한다.

나. DB 저장 시의 데이터 크기 비교

1) 테스트 개요

기존 정보시스템의 DB(EUC-KR 등 사용)를 현대 한글을 모두 표현할 수 있는 유니코드 기반의 DB(UTF-8 사용)로 전환하였을 때, 증가되는 데이터 용량을 파악하여 전환 비용 예상에 참고하기 위함이다.

2) 테스트 방법

- 공공기관 정보시스템에서 가장 많이 사용하는 Oracle Database를 기준으로 테스트 한다.(Oracle Database 11g Release 2 - 11.2.0.1)
- Primary Key(일련번호)와 VARCHAR 등 문자열용 필드 하나로만 구성된 테이블 두 개를 만들고, 각 테이블의 필드 속성을 EUC-KR과 UTF-8로 지정한다.
- 동일한 데이터(샘플 데이터)를 각 인코딩 방식에 따라 일정 단위로 삽입한다.

  100 건
  1,000 건
  10,000 건
  100,000 건
  1,000,000 건
  10,000,000 건
- 두 개의 테이블 크기를 비교한다. (EUC-KR vs UTF-8)

3) 테스트 환경

샘플 테이블

```
CREATE TABLE WEBAPP.UTFTEST
(
        ID              NUMBER (10) NOT NULL,
        ADDRESS         VARCHAR2 (1500) NOT NULL,
        CONSTRAINT ID_PK PRIMARY KEY (ID)
) ;
```

샘플 데이터 (임의의 텍스트 문구)

```
BEGIN
FOR I IN 1..8043248 LOOP
INSERT INTO KSC_TEST values (id_pk.nextval,'3~4의 테스트를 건수를 늘려가면서(추가로 데이터 삽입) 반복합니다. 10건-100건-1000건-10000건 이런식으로 해서 최소 1G의 용량이 나올때까지는 또는 천만건 이상 까지 테스트를 진행합니다3~4의 테스트를 건수를 늘려가면서(추가로 데이터 삽입) 반복합니다. 10건-100건-1000건-10000건 이런식으로 해서 최소 1G의 용량이 나올때까지는 또는 천만건 이상 까지 테스트를 진행합니다3~4의 테스트를 건수를 늘려가면서(추가로 데이터 삽입) 반복합니다');
commit;
end loop;
end;
```

샘플 데이터 구성

전체 – 259자 (419 byte in EUC-KR 인코딩, 579 byte in UTF-8 인코딩)
한글 – 160자 (320 byte in EUC-KR 인코딩, 480 byte in UTF-8 인코딩)
공백, 숫자, 알파벳 – 99자 (99 byte in both)

데이터 크기 계산 값 (이론상, 예상치) * 579 / 419 = 1.38 (단위 : MB)

건 수	EUC-KR	UTF-8	비율
100 건	0.039959	0.055218	1.38
1,000 건	0.39959	0.552177	1.38
10,000 건	3.995895	5.521774	1.38
100,000 건	39.958954	55.217743	1.38
1,000,000 건	399.589539	552.177429	1.38
10,000,000 건	3995.895386	5521.774292	1.38

통계정보 갱신

실행 후 통계정보 갱신을 위해 analyze를 수행

```
SQL> analyze table KSC_TEST compute statistics;
SQL> analyze table UTFTEST compute statistics;
```

4) 테스트 결과

가) KO16KSC5601 (EUC-KR)

(단위 : MB)

100 건

```
SQL>select sum(blocks * 8192)/1024/1024 from user_tables where table_name='KSC_TEST'

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                   .1015625
```

1,000 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name='KSC_TEST';

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                       .625
```

10,000 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name= 'KSC_TEST' ;

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                    5.84375
```

100,000 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name= 'KSC_TEST' ;

SUM(BLOCKS*8192)/1024/1024
----------------------------
                    55.0625
```

1,000,000 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name= 'KSC_TEST' ;

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                  549.039063
```

10,000,000 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name= 'KSC_TEST' ;

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                  6250.07031
```

나) UTF-8

(단위 : MB)

100 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name='UTFTEST';

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                   .1015625
```

1,000 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name='UTFTEST';

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                      .6875
```

10,000 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name='UTFTEST';

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                    6.828125
```

100,000 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name='UTFTEST';

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                   70.9140625
```

1,000,000 건

```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name='UTFTEST';

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                   652.632813
```

10,000,000 건

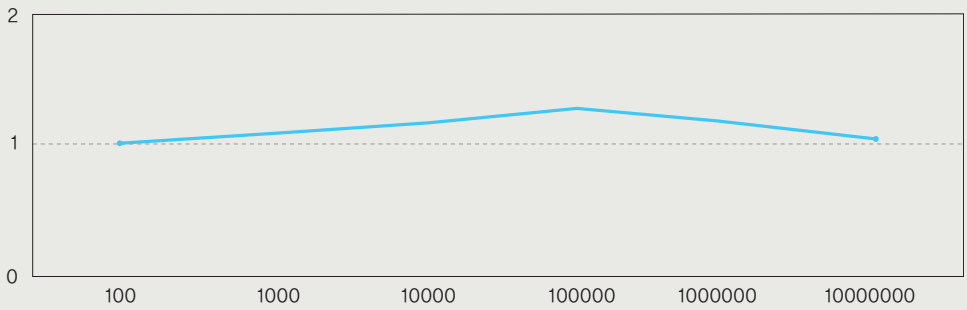
```
SQL> select sum(blocks * 8192)/1024/1024 from user_tables where table_name='UTFTEST';

SUM(BLOCKS * 8192)/1024/1024
----------------------------
                   6549.80469
```

다) 인코딩별 비교

(단위 : MB)

건 수	EUC-KR	UTF-8	비율
100	0.101	0.101	1.000
1,000	0.625	0.687	1.099
10,000	5.843	6.828	1.169
100,000	55.062	70.914	1.288
1,000,000	549.039	652.632	1.189
10,000,000	6250.070	6549.804	1.048

비율 증감 그래프



라) 분석 결과

- 10,000,000건을 기준으로 보았을 때, 약 300MB의 차이가 나타난다. (약 1.05배 증가)
- 이론상 증가 비율(1.38배)과 실제 증가 비율이 다르게 나타나는 이유는 DB에 저장된 데이터는 텍스트 데이터 이외에도 인덱스 등과 같은 Database 처리 정보가 같이 포함되어 있기 때문이며, 이 정보들은 데이터의 건 수, 칼럼 수, 칼럼 속성 등에 따라 용량 기준이 가변적인 것으로 판단된다.
- 건 수는 10배씩 고정폭으로 증가하지만, 증가 비율은 가변폭으로 나타난다.(1.1 → 1.2 → 1.1 → 1.0 식으로, 특정 패턴 파악 불가)

마) 결론

- 상기 테스트는 텍스트 데이터가 포함된 하나의 칼럼만을 사용한 결과로서, 여러 칼럼을 사용하는 실제 환경에서는 더 많은 데이터 증가율을 예상할 수 있다.

- LOB(Large Object) 데이터 타입과 같은 바이너리 형태의 데이터가 있으면 비율의 차이가 달리 나타날 수 있다. (이미지, 동영상 데이터 등)
- 정확한 근사치를 확인하기 위해서는, 다양한 실제 데이터가 포함되어 있는 시스템을 기준으로, EUC-KR에서 UTF-8로 전환(마이그레이션) 시 증가되는 비율을 확인해 보아야 한다.
- 시스템마다 포함되어 있는 데이터 타입과 테이블 구조, 필드 타입 등이 모두 상이하므로, 각 시스템마다 Database 테이블을 추출하여 변환 테스트를 수행한 뒤, 그 용량 증가 비율을 기준으로 전체 증가 비율을 예상해 보아야 한다.
- UTF-8로 인코딩 변환 시 다소 용량이 증가하기는 하지만, 그 정도가 미미하여 용량 증가에 의한 스토리지 비용의 차이는 크지 않을 것으로 판단된다.

첨부 3 변환 모듈 속도 테스트

변환 모듈 사용 여부에 따른 속도 테스트를 수행한다. Web 환경(Java)과 C/S 환경(C++)에서 각각 테스트를 수행하였다.

가. Java 변환 모듈 (Web 환경)

1) 테스트 시스템 정보

구분	내용
OS	Windows XP
CPU	Intel Core2 CPU 6600 @ 2.4GHz
Memory	2,048MB
DB	mysql 5.0

2) 테스트 조건

가) 테스트 시스템 로컬에 DB가 함께 설치되어 있음.

나) 테스트 문자열 : '주소' 필드에 해당하는 값으로 120byte 기준

다) 문자열 데이터를 DB에 바로 입력하는 경우와, 변환 모듈을 사용한 뒤 입력하는 경우로 나누어 산출

라) ConnectionPool을 사용하지 않고 Connection 한 개를 이용하여, PreparedStatement를 10,000번 호출하여 결과 값 산출

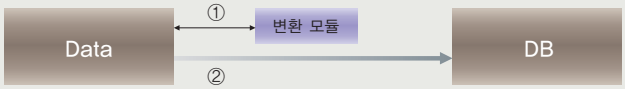
3) 테스트 방법

가) 변환 모듈 미사용

: 일반 UTF-8 데이터를 DB에 바로 입력 (10,000번)DBData?

나) 변환 모듈 사용

: 일반 UTF-8 데이터를 EUC-KR 코드로 변환한 후, DB에 입력 (10,000번)



4) 테스트 결과

가) 소요 시간

변환 모듈 미사용	UTF-8 → DB	2,156 millisecond
변환 모듈 사용	UTF-8 → EUC-KR 변환 → DB	2,344 millisecond

※ 1회 처리 시간이 매우 짧은 관계로 10,000번 수행한 결과를 기준으로 판단

나) 산출 내용 분석

(1) 변환 모듈 10,000회 동작 시간 (변환 모듈 사용과 미사용에 대한 소요 시간 차이)

2,344 - 2,156 = 188 millisecond (0.188 second)

→ 변환 모듈 10,000번 동작 시, 변환 모듈 사용하지 않은 경우보다 0.188초 추가 소요

(2) 변환 모듈 1회 동작 시간

188 / 10,000 = 0.0188 millisecond (0.0000188 second)

→ 변환 모듈 1번 동작 시, 변환 모듈 사용하지 않은 경우보다 0.0000188초 추가 소요

다) 고려 사항

(1) 본 테스트의 목적은 변환 모듈의 성능(속도)을 체크하기 위함이다.

(2) 테스트 시스템에 DB가 같이 설치되어 있는 환경에서 테스트한 결과로서, 실제 운영 시스템(DB 별도 구축)을 기준으로 할 때, DB 커넥션에 더 많은 시간이 소요되며, 이는 변환 모듈 동작 시간 비율이 현재보다 더 낮을 것으로 판단된다.

(3) 브라우저에서 사용자 입력이 이루어지고, 최종적으로 DB 저장에 이르기까지의 모든 과정을 고려한다면, 변환 모듈 소요시간 비율이 훨씬 낮을 것으로 판단된다.

(4) 실제 운영 시스템에서는 여러 환경 조건이 다르므로, 본 테스트의 결과와 일부 다르게 나타날 수 있다.

라) 결과

(1) 변환 모듈 동작에는 가늠하기 힘들 만큼의 짧은 처리 시간이 소요됨
(2) 현재 운영 중인 시스템에 변환 모듈을 적용하더라도 시스템 처리시간에 미치는 영향은 극히 미미함

나. C++ 변환 모듈 (C/S 환경)

1) 테스트 시스템 정보

구분	내용
OS	Windows 7
CPU	Intel Q9650 @ 3.0GHz
Memory	4,096MB

2) 테스트 조건

가) 테스트 시스템에서 C++ 변환 모듈 동작 속도만 확인함
나) DB 커넥션/저장 및 변환 모듈 미사용 등과 관련 정보는 Java 변환 모듈의 자료를 참조

3) 테스트 방법

가) UTF-8 데이터를 EUC-KR로 변환하는 변환 모듈의 함수 처리 속도 확인

4) 테스트 결과

가) 소요 시간 (변환 모듈 한정)

변환 모듈 사용	Unicode → EUC-KR 변환	34 millisecond

※ 1회 처리 시간이 매우 짧은 관계로 10,000번 수행한 결과를 기준으로 판단

나) 산출 내용 분석

(1) 변환 모듈 10,000회 동작 시간 : 34 millisecond (0.034 second)

(2) 변환 모듈 1회 동작 시간 : 0.0034 millisecond (0.0000034 second)

다) 고려 사항

(1) Java 변환 모듈과 비교 시, 개발언어의 종류(Java vs C++)만 다를 뿐, 나머지 조건(함수 호출, DB 커넥션, 저장 등)은 동일하므로, C++ 변환 모듈의 동작 속도만 확인

(2) 일반적으로 Java 보다 C++의 동작 속도가 빠르다.

라) 결과

(1) 변환 모듈 동작에는 가늠하기 힘들 만큼의 짧은 처리 시간이 소요됨

(2) 현재 운영 중인 시스템에 변환 모듈을 적용하더라도 시스템 처리시간에 미치는 영향은 극히 미미함

첨부 4

웹 브라우저 인코딩 처리

웹브라우저에서 웹서버로 데이터 전송 시, 인코딩 과정에서 처리되지 않는 (해당 인코딩으로 처리할 수 없는) 코드를 자동으로 변환 처리하는 기능의 존재유무를 확인하고, 변환 처리하는 값의 유형(형태)을 조사한다.

가. 테스트 환경 (시나리오)

1) 확장한글 문제가 발생 가능한 환경으로 구성
2) 웹서버에서 입력을 받는 웹페이지의 인코딩 타입을 euc-kr로 선언
3) 웹서버에 접속한 PC의 웹브라우저에서 입력 데이터를 웹서버로 전송
4) 웹브라우저는 웹페이지의 인코딩 타입에 따라 입력 데이터를 euc-kr로 인코딩하여 발송
5) 입력 데이터 중 인코딩 되지 않는 코드가 존재(예 포스코 더 아파트)
6) 웹브라우저에서 해당 문구를 자동으로 처리하여 발송하는지 확인

나. 테스트 방법

1) 문자열 : 웹브라우저에서 웹서버로 전달된 데이터 결과
2) HEX Code : 문자열을 바이트 추출 후 바이트 값을 헥사코드로 변환
3) Java Code : 문자열을 자바 소스코드로 변환
4) HTML Entity Code : 문자열을 HTML 엔티티 코드로 변환
5) HEX String → 문자열 : 문자열을 핵사스트링으로 변환하고, 다시 문자열로 변환

다. 테스트 결과

1) 인터넷 익스플로러, 크롬, 사파리 등의 웹브라우저에서는 확장한글 이슈가 있는 문자열 전송 시, 별도의 처리 없이 (해당 문자 인코딩으로 인코딩 되지 않는) 오류 코드를 그대로 발송함을 확인

2) 파이어폭스 웹브라우저에서는 확장한글 이슈가 있는 문자열 전송 시, KS채움쪽자 방식으로 "풀어읽기 / 채워쓰기" 과정을 거쳐, 정상적으로 문자열을 인식 처리함을 확인

3) NCR 방식의 처리 결과는 어느 웹브라우저에서도 찾아볼 수 없음.

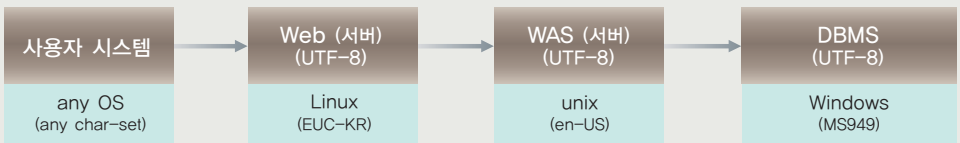
구분	IE / Chrome / Safari	Firefox
문자열	포스코 더 ? 아파트	포스코 더 샾 아파트
HEX Code	-58 : byte to hexa code : 0xc6 -9 : byte to hexa code : 0xf7 -67 : byte to hexa code : 0xbd -70 : byte to hexa code : 0xba -60 : byte to hexa code : 0xc4 -38 : byte to hexa code : 0xda 32 : byte to hexa code : 0x20 -76 : byte to hexa code : 0xb4 -11 : byte to hexa code : 0xf5 32 : byte to hexa code : 0x20 **63 : byte to hexa code : 0x3f** 32 : byte to hexa code : 0x20 -66 : byte to hexa code : 0xbe -58 : byte to hexa code : 0xc6 -58 : byte to hexa code : 0xc6 -60 : byte to hexa code : 0xc4 -58 : byte to hexa code : 0xc6 -82 : byte to hexa code : 0xae	-58 : byte to hexa code : 0xc6 -9 : byte to hexa code : 0xf7 -67 : byte to hexa code : 0xbd -70 : byte to hexa code : 0xba -60 : byte to hexa code : 0xc4 -38 : byte to hexa code : 0xda 32 : byte to hexa code : 0x20 -76 : byte to hexa code : 0xb4 -11 : byte to hexa code : 0xf5 32 : byte to hexa code : 0x20 **-92 : byte to hexa code : 0xa4** **-44 : byte to hexa code : 0xd4** **-92 : byte to hexa code : 0xa4** **-75 : byte to hexa code : 0xb5** **-92 : byte to hexa code : 0xa4** **-63 : byte to hexa code : 0xc1** **-92 : byte to hexa code : 0xa4** **-67 : byte to hexa code : 0xbd** 32 : byte to hexa code : 0x20 -66 : byte to hexa code : 0xbe -58 : byte to hexa code : 0xc6 -58 : byte to hexa code : 0xc6 -60 : byte to hexa code : 0xc4 -58 : byte to hexa code : 0xc6 -82 : byte to hexa code : 0xae
Java Code	\ud3ec\uc2a4\ucf54 \ub354 **\ufffd** \uc544\ud30c\ud2b8	\ud3ec\uc2a4\ucf54 \ub354 **\u3164\u3145\u3151\u314d** \uc544\ud30c\ud2b8
HTML Entity Code	포스코 더 ◆ 아파트	포스코 더 ㅅㅑㅍ 아파트
HEX String ↓ 문자열	hexa string : 00d3ec , hexa string to character 포 hexa string : 00c2a4 , hexa string to character 스 hexa string : 00cf54 , hexa string to character 코 hexa string : 0020 , hexa string to character hexa string : 00b354 , hexa string to character 더 hexa string : 0020 , hexa string to character **hexa string : 00fffd , hexa string to character ?** hexa string : 0020 , hexa string to character hexa string : 00c544 , hexa string to character 아 hexa string : 00d30c , hexa string to character 파 hexa string : 00d2b8 , hexa string to character 트	hexa string : 00d3ec , hexa string to character 포 hexa string : 00c2a4 , hexa string to character 스 hexa string : 00cf54 , hexa string to character 코 hexa string : 0020 , hexa string to character hexa string : 00b354 , hexa string to character 더 hexa string : 0020 , hexa string to character **hexa string : 003164 , hexa string to character** **hexa string : 003145 , hexa string to character ㅅ** **hexa string : 003151 , hexa string to character ㅑ** **hexa string : 00314d , hexa string to character ㅍ** hexa string : 0020 , hexa string to character hexa string : 00c544 , hexa string to character 아 hexa string : 00d30c , hexa string to character 파 hexa string : 00d2b8 , hexa string to character 트

첨부 5 WAS/Web 서버 간 문자 인코딩 처리 확인

각 기관 정보시스템을 구성하고 있는, WAS/Web 서버 사이의 데이터 통신에 있어, 운영체제(OS)의 문자 인코딩에 따른 영향이 발생하는지 확인한다.

가. OS의 문자 인코딩 설정

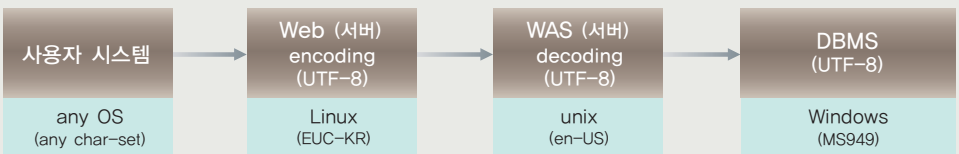
1) 정보시스템 구성 (예제1)



가) 각 단계별(사용자PC, Web서버, WAS서버, DBMS서버 등) 운영체제가 각각 상이하고, 사용되는 문자 인코딩도 다양한 케이스를 예상

나) 각 단계마다 흐르는 데이터는 모두 유니코드(UTF-8)로 일관되게 동작함을 예상

2) 정보시스템 구성 (예제2)



가) 각 단계별 운영체제가 각각 상이하고 사용되는 문자 인코딩도 다양함

나) 단계마다 흐르는 데이터가 변환되는 케이스

예 민원24

나. 확인사항

1) 사용자 시스템의 문자 인코딩은 어떤 것이든 상관없다.

가) 웹서버에 있는 웹페이지의 문자 인코딩 설정에 따라 입력되는 문자 인코딩이 결정되므로, 사용자 시스템은 이후 서버 간의 데이터 통신에 영향을 미치지 않는다.

웹페이지에서의 문자 인코딩 정의 예제

```
<meta http-equiv="content-type" content="text/html;charset=UTF-8" />
```

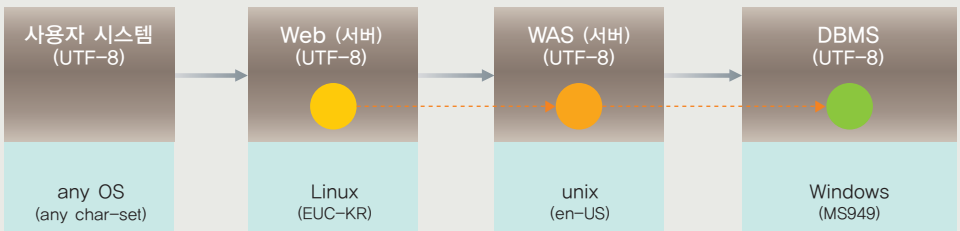
나) 위와 같이 웹페이지의 문자 인코딩이 선언되면, 이 문자 인코딩 설정에 따라사용자가 입력한 데이터도 인코딩되어 웹서버에 전달된다.

다) 즉, 웹페이지가 EUC-KR로 선언되어 있으면 그 페이지에서 입력한 데이터도 EUC-KR로 인코딩되어 전달되고, 웹페이지가 UTF-8로 선언되어 있으면 사용자가 거기서 입력한 데이터도 UTF-8로 인코딩되어 서버로 전달된다.

2) OS의 문자 인코딩 영향

가) OS 상에서 동작하는 Web, WAS, DB가 데이터를 어떻게 다루는지가 중요한 것이며, OS의 문자 인코딩 설정은 데이터의 문제와 직접적인 관련이 없다.

나) 각 서버(Web/WAS/DB)에서 데이터 처리를 by-pass로 진행한다면,OS는 단지 데이터를 전달하는 역할만을 수행하므로 데이터에 변화가 발생하지 않는다.

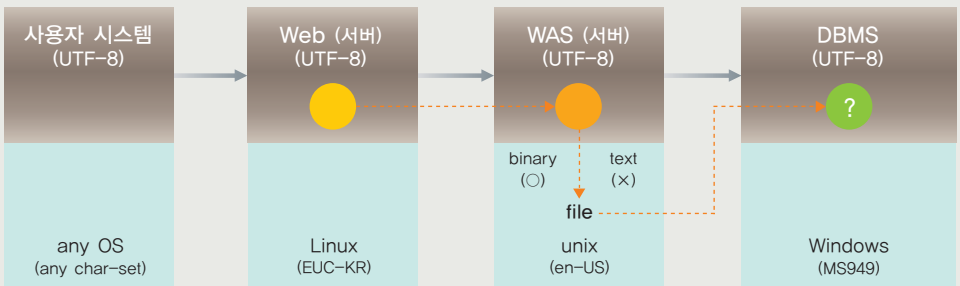


다) 각 서버(Web/WAS/DB)에서 별도 데이터 처리를 위해 운영체제의 파일시스템을 거치게 된다면, (OS의 파일시스템에 저장/가공하는 과정을 수행한 뒤, 그 결과를

전송한다면,) 이렇게 저장되는 문자열(데이터)은 처리 방법에 따라 이슈가 발생할 수 있다.

(1) 바이너리 입출력 함수를 사용하여 파일을 저장하면 인코딩 문제가 발생하지 않는다.

(2) 그러나 텍스트 입출력 함수를 사용하면 컴파일러와 OS에 영향을 받아 내용이 변경될 수 있다.



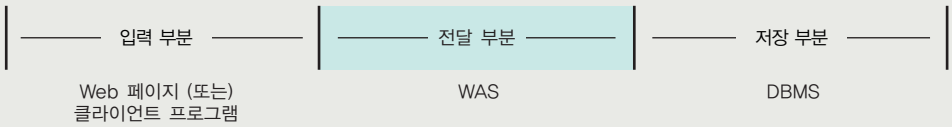
(3) 위와 같이 Web, WAS, DBMS 단에서 운영체제의 파일시스템을 이용하는 경우는, 대부분 파일 첨부 전송에 해당되며, 이 경우 바이너리 입출력을 하므로 문제가 발생하지 않는다.

다. 결론

1) 입력되는 데이터의 무결성을 확보하는 방안

가) 일단 DB에 데이터가 정상적으로 입력된다면, 읽고 활용하는 것은 어떠한 방법으로도 가능하므로, 가장 중요한 것은 DB에 데이터를 정상적으로 입력하도록 하는 것이 중요하다.

나) 상기 시스템 구성을 아래와 같이 크게 3부분으로 나누어 볼 때, 중간의 전달 부분에서 데이터를 잘 못 가공/처리하는 부분이 없는지 살펴보아야 한다.운영체제의 파일시스템에 저장/가공하지 않고, 운영체제는 단지 데이터를 전달하는 역할만을 수행한다면, 데이터 훼손의 문제는 발생하지 않는다.



다) 즉 입력 부분에서 입력되는 데이터와 저장 부분에서 저장되는 데이터가 동일하게 유지되도록 전달 부분을 처리해 주어야 한다.

라) 가장 이상적인 방향은, 앞서 본 예제와 같이 전달되는 데이터가 하나의 문자 인코딩으로 통일되는 것이며, 중간에 운영체제의 파일시스템을 경유하는 과정 없이 상위 어플리케이션 단에서 모두 처리되어 by-pass로 전송되는 것이다.

마) 확장한글 문제와 같이 고려할 때, (운영체제의 문자 인코딩과는 상관없이) 입력/전송/저장되는 데이터는 확장한글을 모두 지원하는 유니코드(UTF-8)로 통일하고, 중간에 운영체제와의 의존 관계(파일시스템 사용)를 만들지 않는 것이 가장 효과적인 방안이다.

첨부 6

KS채움쪽자 사용자 활용 팁

KS 표준으로 규정되어 있는, 쪽자 방식을 사용자 단에서 간단히 구현할 수 있는 방법을 제시한다.

- KS채움쪽자 변환을 위한 별도의 모듈을 사용하지 않고도, 일반 사용자가 이를 구현할 수 있는 방법 고려
- 전제 조건 (아래 환경 사용을 기준으로 함)
- 사용자의 PC는 윈도우(XP 이상) 운영체제를 사용
- 웹 브라우저는 KS채움쪽자를 지원하는 파이어폭스(Firefox) 사용

가. Windows 문자표 활용

1) 문자표 검색

가) 문자표 실행

(1) 시작 〉 프로그램 〉 보조프로그램 〉 시스템도구 〉 문자표
(또는) 시작 〉 실행 〉 "charmap" 실행

나) 채움 문자 검색

(1) '유니코드로 이동' 란에 "3164" 입력
※ 하단 상태표시줄에 "U+3164 : Hangul Filler"로 표시되면 정상

다) 선택/복사

(1) "선택" 버튼을 클릭한다.

(2) "복사할 문자"란에 특정 코드가 입력되지만, "채움"코드에 해당하는 적절한 폰트가 없으므로, 화면에는 공백으로 표시된다.

(3) 실제로는 해당코드가 입력되어있는 상태이므로, 마우스로 블록처리하고 "복사"를 실행한다.

2) 문서 작성

가) 웹브라우저에서 확인할 수 있도록 HTML 파일을 생성한다.파일 생성시 상단의 Meta 태그에 euc-kr 문자 인코딩을 반드시 명시한다.

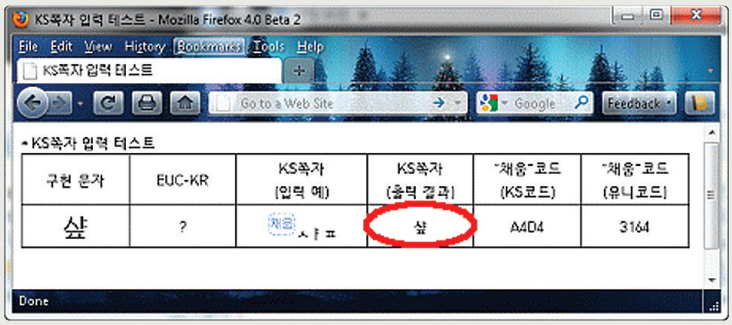
```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=euc-kr">
```

나) 앞서 문자표에서 복사한 코드 값을 아래의 "KS채움쪽자 입력 예"와 같이 붙여넣고, (실제로는 코드 자리만 차지하고 보이지 않음)
"샾"의 초성, 중성, 종성을 모두 띄어서 각각 입력한다. (채움 ㅅ ㅑ ㅍ)

구현 문자	EUC-KR	KS채움쪽자 입력 예	KS채움쪽자 출력 결과	"채움"코드 (KS코드)	"채움"코드 (유니코드)
샾	?	채움 ㅅ ㅑ ㅍ	ㅅ ㅑ ㅍ	A4D4	3164

3) 문서 확인

가) 작업 완료한 HTML 문서를 저장 후, 웹브라우저(Firefox)에서 확인한다. ("KS채움쪽자 출력 결과" 전/후 비교)



나. [참고] 한컴오피스 문자표 사용하기

Windows의 문자표 이외에도 한컴오피스의 문자표를 사용하여 "채움" 코드를 찾을 수도 있다.

가) 상단 메뉴에서 "입력 〉 문자표"를 실행하거나, 단축키 "Ctrl + F10"을 사용하여 문자표를 띄운다.

나) 상단 탭에서 "완성형(KS) 문자표"를 선택하고, 좌측의 "문자 영역"에서 "한글 낱자"를 선택한다.

다) 문자표 리스트 마지막에 있는 채움을 선택하고 "넣기" 버튼을 클릭한다.

라) 본문 상에 입력된 문자(커서 부분에 1문자의 자리는 차지하고 있으나, 보이지는 않음)를 복사하여 이후 단계를 진행한다.

마) 이후 단계는 앞서 설명한 Windows 문자표 활용법 2, 3번과 동일하다.

첨부 7 KS표준 명칭(naming) 규정

KS표준 제정/개정 시 명칭 지정에 대한 규정을 확인한다.
예 KS X 1005가 KS X ISO/IEC 10646으로 바뀐 규정의 근거

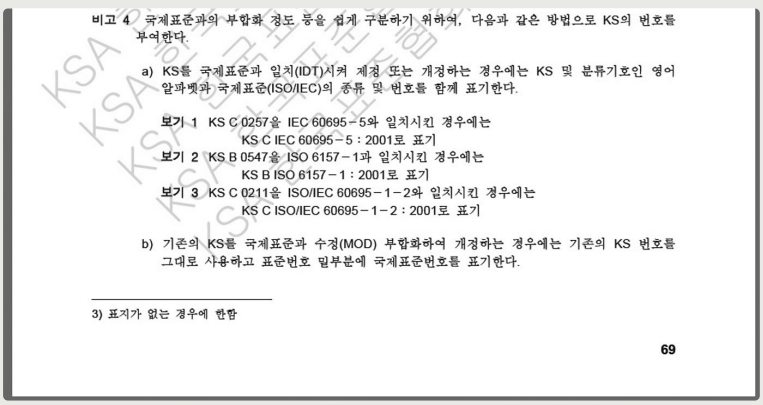
가. KS표준 명칭 규정 (index)

KS A 0001 표준서의 서식 및 작성방법
부속서 K
K.1 표준의 체제
비고 4. a) 국제표준과의 부합화 정도에 대해.

나. 설명

비고 4 국제표준과의 부합화 정도 등을 쉽게 구분하기 위하여, 다음과 같은 방법으로 KS의 번호를 부여한다.
a) KS를 국제표준과 일치(IDT)시켜 제정 또는 개정하는 경우에는 KS 및 분류기호인 영어 알파벳과 국제표준(ISO/IEC)의 종류 및 번호를 함께 표기한다.

다. 원문



비고 4 국제표준과의 부합화 정도 등을 쉽게 구분하기 위하여, 다음과 같은 방법으로 KS의 번호를 부여한다.

a) KS를 국제표준과 일치(IDT)시켜 제정 또는 개정하는 경우에는 KS 및 분류기호인 영어 알파벳과 국제표준(ISO/IEC)의 종류 및 번호를 함께 표기한다.

보기 1 KS C 0257을 IEC 60695-5와 일치시킨 경우에는
KS C IEC 60695-5 : 2001로 표기
보기 2 KS B 0547을 ISO 6157-1과 일치시킨 경우에는
KS B ISO 6157-1 : 2001로 표기
보기 3 KS C 0211을 ISO/IEC 60695-1-2와 일치시킨 경우에는
KS C ISO/IEC 60695-1-2 : 2001로 표기

b) 기존의 KS를 국제표준과 수정(MOD) 부합화하여 개정하는 경우에는 기존의 KS 번호를 그대로 사용하고 표준번호 일부분에 국제표준번호를 표기한다.

3) 표지가 없는 경우에 한함

69

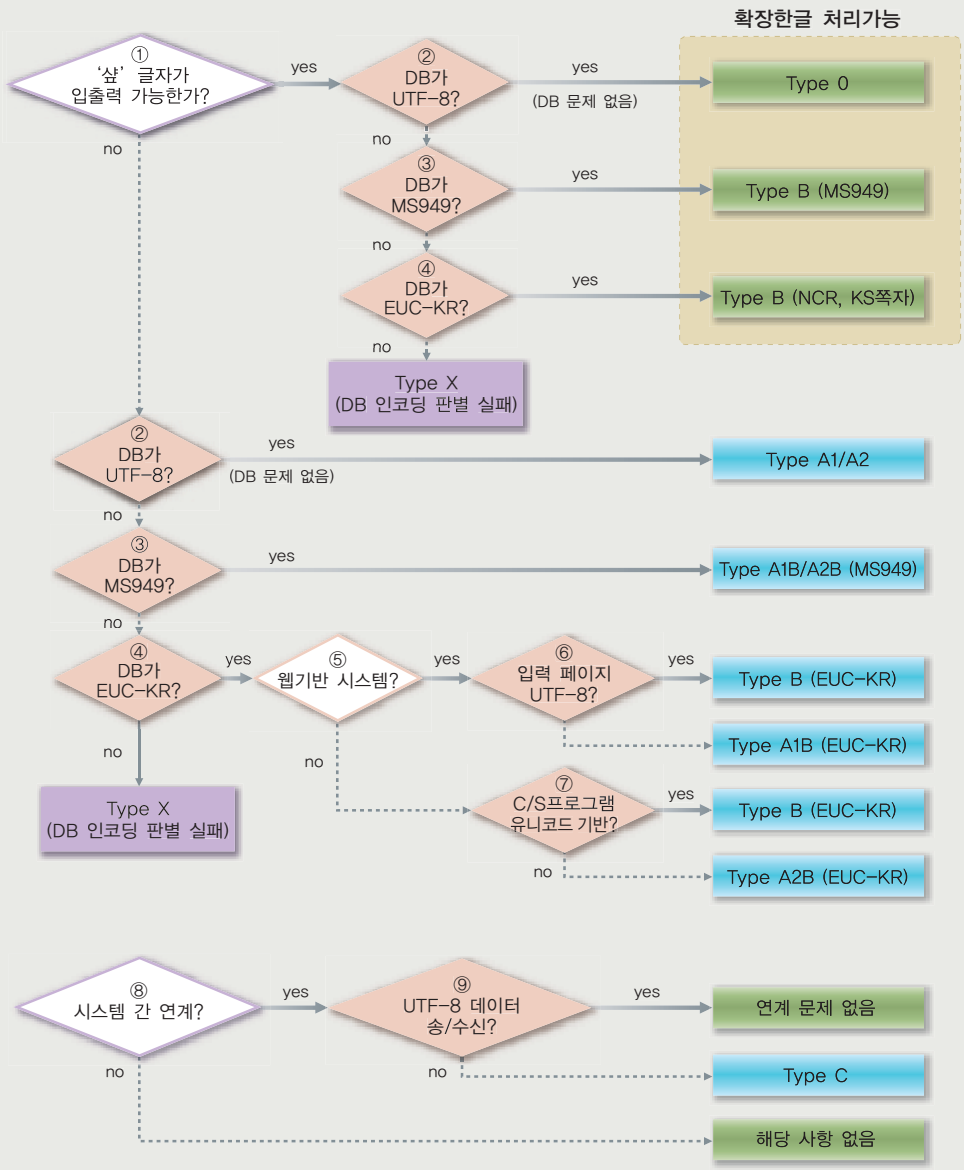
첨부 8 확장한글 발생 예

한글 문자코드 사용 중 접할 수 있는 확장한글의 사례들을 조사해 본다.

현재의 대부분 운영체제는 유니코드를 지원하므로 확장한글 표현에 문제가 없으나, 이를 저장하는 데이터베이스 또는 e-메일 시스템에서 EUC-KR 코드를 사용하는 경우, 아래와 같은 확장한글 문자들은 정상 표현되지 않는다.

확장한글 예	EUC-KR 변환 시
발음 표기	
'태권도'의 발음 → 태꿘도	태?도
'협의'의 발음 → 혀븨	혀?
준말	
전화입니다 → 전홥니다	전?니다
받침 없는 글자	
뢨 – 뢔	뢨 – ?
쌿 – 쌰	쌿 – ?
쏃 – 쎄	쏃 – ?
쓩 – 쓔	쓩 – ?
쭁 – 쬬	쭁 – ?
비표준어	
늉큼	?큼
외래어	
펩시콜라 – 펲시콜라	펩시콜라 – ?시콜라
커피숍 – 커피숖	커피숍 – 커피?
인터넷 언어	
먄	?
뷔	?
햏햏	??

첨부 9 유형 판별 차트



※ 연계된 시스템의 경우, A/B유형 판별 후 C유형 판별 결과를 추가 파악해야 한다.

예 Type A1B + Type C = Type A1BC

체크 사항	해당 여부 (O 또는 X)	비고
① 시스템에서 '샾' 글자가 입출력 가능한가?		
② DBMS에서 UTF-8을 사용하고 있는가?		
③ DBMS에서 MS949를 사용하고 있는가?		
④ DBMS에서 EUC-KR을 사용하고 있는가?		
⑤ 웹 기반 시스템인가?		
⑥ 사용자 데이터 입출력 페이지가 UTF-8을 사용하는가?		
⑦ C/S 시스템인 경우, 유니코드 기반 응용프로그램인가?		
⑧ 다른 단일시스템과 연계하고 있는가?		
⑨ 송/수신 데이터가 UTF-8 형식인가?		

문항 체크 결과 (① ~ ⑦)							시스템 유형 (A/B 기준)	비고
①	②	③	④	⑤	⑥	⑦		
O	O	×	×				Type 0	확장한글 처리 가능
O	×	O	×				Type B (MS949 DB)	
O	×	×	O				Type B (NCR, KS쪽자 DB)	
×	O	×	×	O			Type A1	
×	O	×	×	×			Type A2	
×	×	O	×	O			Type A1B (MS949 DB)	
×	×	O	×	×			Type A2B (MS949 DB)	
×	×	×	O	O	O		Type B (EUC-KR DB)	
×	×	×	O	O	×		Type A1B (EUC-KR DB)	
×	×	×	O	×		O	Type B (EUC-KR DB)	
×	×	×	O	×		×	Type A2B (EUC-KR DB)	

문항 체크 결과 (⑧, ⑨)		시스템 유형 (C 기준)	비고
⑧	⑨		
O	O	-	연계 OK
O	×	Type C	
×		-	해당 없음

※ 연계된 시스템의 경우, A/B유형 판별 후 C유형 판별 결과를 추가 파악해야 한다.

예 Type A1B + Type C = Type A1BC

첨부 10 현재 시스템의 문자 인코딩 확인

가. DBMS : MySQL의 예

MySQL DB에서 사용 중인 문자 인코딩은 아래와 같이 mysql 프롬프트 상태에서 status 명령어로 확인할 수 있다. 그리고 charset 명령을 이용해 다른 인코딩으로 변경할 수 있다.

```
mysql> status
--------------
mysql  Ver 14.14 Distrib 5.1.41, for debian-linux-gnu (i486) using readline 6.1

Connection id:          34
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.1.41-3ubuntu12.6 (Ubuntu)
Protocol version:       10
Connection:             Localhost via UNIX socket
Client characterset:    euckr
Server characterset:    euckr
UNIX socket:            /var/run/mysqld/mysqld.sock
Uptime:                 13 hours 11 min 57 sec

Threads: 1  Questions:  104  Slow queries: 0  Opens: 99  Flush tables: 1  Open tables: 23  Queries per second avg: 0.2
--------------

mysql> charset utf8
Charset changed
mysql> status
```

```
--------------
mysql  Ver 14.14 Distrib 5.1.41, for debian-linux-gnu (i486) using readline 6.1

Connection id:          34
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.1.41-3ubuntu12.6 (Ubuntu)
Protocol version:       10
Connection:             Localhost via UNIX socket
Client characterset:    utf8
Server characterset:    utf8
UNIX socket:            /var/run/mysqld/mysqld.sock
Uptime:                 13 hours 12 min 8 sec

Threads: 1  Questions: 107  Slow queries: 0  Opens: 99  Flush tables: 1  Open
tables: 23  Queries per second avg: 0.2
--------------
```

나. WAS : JEUS 6의 예

JEUS 서버에서 HTTP 요청/응답과 관련된 인코딩은 jeus-web-dd.xml 파일 또는 WEBMain.xml 파일 등에 설정된다. 이밖에 다른 곳에서도 문자 인코딩 설정이 가능한데, 그것들 사이에 우선순위가 있으므로 자세한 사항은 JEUS 매뉴얼을 참고하도록 한다.

또한 이러한 미들웨어가 제공하는 기능을 기반으로 상위에서 동작하는 프로그램들이 어떠한 문자 인코딩을 사용하도록 만들어졌는지도 또한 확인해보아야 한다.

- 참고 : JEUS Web Container 안내서 – 5장 Context Group – 5.2.7. 인코딩 (http://technet.tmax.co.kr/kr/edocs/jeus/60/web-container/chapter_context_group.html#sect_encoding)

다. Web : Apache의 예

Apache 웹 서버의 기본 인코딩은 .htaccess 파일에서 AddCharset 등의 지시어를 통해서 설정 가능하다. 예를 들어 .html 확장자를 가지는 모든 파일에 대해 UTF-8 인코딩을 적용하려면

```
AddCharset UTF-8 .html
```

과 같은 내용을 .htaccess 파일에 넣어주면 된다.
이밖에도 각각의 웹 페이지가 올바른 인코딩 설정을 갖게 하려면, 페이지의 데이터 자체가 해당 인코딩으로 저장되어야 하고, 그 인코딩이 페이지 안에 명시되어 있어야 한다. 이와 관련된 내용이 아래 링크들에 자세히 나와있으므로 참고하도록 한다.

- Internationalization Techniques: Authoring HTML & CSS
  (http://www.w3.org/International/ techniques/authoring-html#changing)
- W3C I18N Techniques: Setting up a server
  (http://www.w3.org/International/techniques/ server-setup)

- Changing (X)HTML page encoding to UTF-8
  (http://www.w3.org/International/questions/qa-changing-encoding)
- Setting the HTTP charset parameter
  (http://www.w3.org/International/O-HTTP-charset)
- Setting charset information in .htaccess
  (http://www.w3.org/International/questions/qa-htaccess-charset)

라. OS : Linux와 Windows의 예

1) Linux의 예

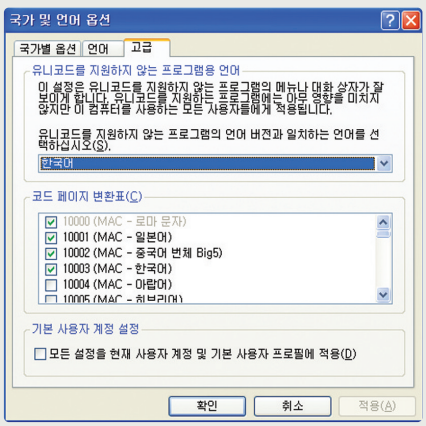
Linux 운영체제에서 사용 중인 문자 인코딩은 아래와 같이 locale 명령어로 확인할 수 있다. 유니코드가 아닌 MBCS 문자열을 다루는 함수들은 현재 로케일 설정에서 LC_CTYPE 설정값의 영향을 받게 된다.

```
ubuntu:~$ locale
LANG=ko_KR.utf8
LC_CTYPE="ko_KR.utf8"
LC_NUMERIC="ko_KR.utf8"
LC_TIME="ko_KR.utf8"
LC_COLLATE="ko_KR.utf8"
LC_MONETARY="ko_KR.utf8"
LC_MESSAGES="ko_KR.utf8"
LC_PAPER="ko_KR.utf8"
LC_NAME="ko_KR.utf8"
LC_ADDRESS="ko_KR.utf8"
LC_TELEPHONE="ko_KR.utf8"
LC_MEASUREMENT="ko_KR.utf8"
LC_IDENTIFICATION="ko_KR.utf8"
LC_ALL=
```

- 참고 : Linux Unicode programming
  (http://www.ibm.com/developerworks/library/l-linuni.html)
  (http://www.ibm.com/developerworks/kr/linux/library/l-linuni.html – 한국어. 번역은 매끄럽지 못함)
- 참고 : Using UTF-8 with Gentoo
  (http://www.gentoo.org/doc/en/utf-8.xml)

2) Windows의 예

Windows XP에서는 제어판의 '국가 및 언어 옵션'에서 '유니코드를 지원하지 않는 프로그램용 언어'를 확인하고 설정할 수 있다. 유니코드 기반으로 작성된 응용프로그램은 이 설정과 무관하게 문자를 표시하고 입출력할 수 있으나, 유니코드 기반이 아닌 응용프로그램은 프로그램 작성 시에 사용한 문자 인코딩(코드 페이지)과 이 설정이 일치해야 문자 표시와 입출력에 문제가 없게 된다.



- 참고 : Windows XP 다국어 기능 지원 개요
  (http://msdn.microsoft.com/ko-kr/goglobal/bb688163.aspx)

첨부 11 참고 자료

[Unicode]

- 유니코드에 대해
  http://www.unicode.org/standard/translations/korean.html
- Glossary of Unicode Terms
  http://www.unicode.org/glossary/
- Unicode nearing 50% of the web – The Official Google Blog
  http://googleblog.blogspot.com/2010/01/unicode-nearing-50-of-web.html
- Unicode and Character Sets
  http://www.joelonsoftware.com/articles/Unicode.html

[Web]

- W3C Internationalization (I18n) Activity
  http://www.w3.org/International/
- Internationalization Techniques: Authoring HTML & CSS
  (http://www.w3.org/International/techniques/authoring-html#changing)
- Migrating to Unicode
  http://www.w3.org/International/articles/unicode-migration/

[Database]

- Oracle Globalization Support
  http://www.oracle.com/technetwork/database/features/globalization
- Character Set Migration Best Practices for Oracle Database 10g
  http://www.oracle.com/technetwork/database/features/globalization/twp-character-set-migration-best-pr-128766.pdf
- Character Set Migration Best Practices for Oracle9i
  http://www.oracle.com/technetwork/database/features/globalization/mwp-129407.pdf

[Windows]

- Windows Codepage 949
  http://msdn.microsoft.com/ko-kr/goglobal/cc305154(en-us).aspx
- 세계화 단계별 연습: 유니코드 사용
  http://msdn.microsoft.com/ko-kr/goglobal/bb688113.aspx
- Windows XP 다국어 기능 지원 개요
  http://msdn.microsoft.com/ko-kr/goglobal/bb688163.aspx

[Linux]

- Linux Unicode programming
  http://www.ibm.com/developerworks/library/l-linuni.html
- Using UTF-8 with Gentoo
  http://www.gentoo.org/doc/en/utf-8.xml

'샆' 등 한글 표현 문제 해결을 위한

공공 정보시스템
한글 처리 가이드라인

2010년 12월 인쇄
2010년 12월 발행

발행처 행정안전부/한국정보화진흥원
인 쇄 신생용사촌

3. 본 가이드라인에 대한 문의 및 개선에 관한 의견이 있으신 분들은 아래의 문의처로 연락주시기 바랍니다.

• 행정안전부 정보자원정책과 손성주 사무관
(e-mail : son1977@mopas.go.kr)
• 한국정보화진흥원 정보기술전략기획부 나영인 책임
(e-mail : nyi@nia.or.kr)