EN-CORE Research Center

[데이터 이행을 위한 실전 방안]

새로운 서비스를 운영하기 위한 새로운 정보 시스템이 요구되고, 기존 정보 시스템의 수명 주기로 인해 기업들은 과거 시스템으로는 더이상 현업의 요구 사항에 신속하게 대응하는 데 한계가 발생하고 차세대 시스템 구축 요구가 발생됨에 따라 기업들은 대규모 데이터 이행에 착수하였다.

또한 다양한 기종의 DBMS들이 서로 연동하는 인터페이스 부분과 서로 다른 SQL 구문 및 응용 프로그램의 개발 언어로 불필요한 비용이 지출되는 문제가 대두되면서 IT 관리자들은 이를 해결하기 위한 방안의 하나로 데이터 이행을 고민하게 됐다.

이번 엔코아 리포트에서는 비용 절감과 업무 효율도를 높이기 위한 데이터 이행을 보다 효율적으로 수행할 수 있는 방안을 제시하고자 한다.

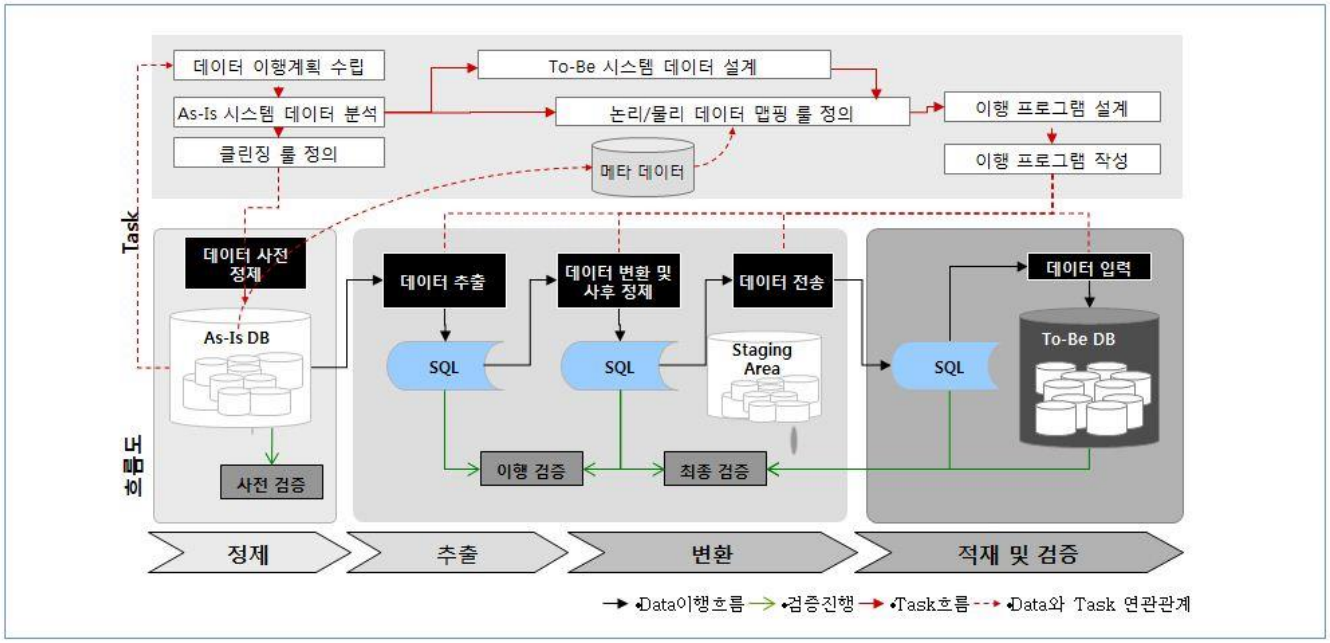
1. 데이터 이행(Data migration)이란?

데이터 이행이라함은 한 개 이상의 시스템(Source System)으로부터 다른 시스템(Target System)으로 데이터 또는 스키마를 옮기는 것을 의미한다. 동일한 형태와 내용으로 옮겨질 수도 있지만 타겟 시스템이 새로 개발된 신규 시스템이라면 데이터만을 새로이 가공해서 옮길 수도 있다. 데이터 이행을 그냥 이전 시스템에 존재하고 있는 데이터를 새로운 시스템으로 옮기는 작업이라고만 생각할 수도 있지만 그것 보다는 이전 시스템에 있는 데이터를 새로운 시스템으로 옮기면서 정제를 한다고 생각하는 것이 가장 올바른 것이라고 할 수 있다.

그렇기 때문에 데이터 이행의 첫 번째 과제는 '정제'라고 할 수 있다. 즉, 얼마나 정확하게 데이터를 Target(목표) 시스템의 구조에 맞게 이행을 시켜서 이전 시스템의 데이터와

새로운 시스템의 데이터가 공존을 할 수 있도록 해주느냐이다

원본 시스템이 오래될수록 그리고 복잡할수록 정제 작업의 중요성과 소요 시간은 늘어나지만, 100% 완벽하게 정제를 하기에는 많은 어려움이 따른다. 지속적으로 클라이언트와의 대화를 통해 지속적으로 정리를 하고 적정선에서 타협을 하는 것이 관건이다. 대규모 시스템에서는 대량의 데이터를 처리해야 하기 때문에 정확한 업무 이해와 튜닝이 중요하다. 이로써 정확하고, 표준화되고 일치된 데이터를 통해 투입된 투자 대비 높은 ROI를 실현할 수 있는 선행 조건을 갖추게 되는 것이다.



<데이터 이행 구성도>

데이터 정제가 끝나면 데이터 이행에서 중요한 것은 얼마나 빨리 이행을 하느냐일 것이다. 대부분 몇 년 이상 운영한 시스템의 경우라면 한 테이블이 천만 건 이상이 되는 경우가 많기 때문에 이행은 엄청난 시간이 소요될 것이다.

즉, 데이터 이행은 주어진 이행 시간내에 기존 운영 데이터(ASIS)를 목표 시스템(TOBE)으로 정확하고, 빠르게 적재하는 것이다. 물론 기존 데이터에 대한 누락이 없어야 하며 잘못된 데이터에 대해서는 바르게 수정해야 함은 기본이다. 특히 대용량 데이터인 경우 전체 이행 시간에서 차지하는 비중이 높은 편이다. 따라서 이를 얼마나 빠른 시간 내에 완료할 수 있는지가 전체 이행 시간을 좌지우지 할 수 있다.

2. 데이터 이행의 유형

데이터 이행의 유형을 보면 크게 2 가지 방식으로 나누어진다.

첫 번째 방법은 하드웨어의 구성만 바뀌고 소프트웨어와 데이터는 그대로 사용하는 리-호스팅을 위한 데이터 이행이다. 예를 들면 IBM 메인프레임을 사용하던 기업에서 개방 시스템을 도입하여 구조만 바꾸고 모든 데이터와 애플리케이션을 그대로 사용하는 방법으로 리스크를 최소화하는 방안으로 많이 선택되고 있다. 이 경우 소스 데이터베이스와 목표 데이터베이스가 거의 1 대 1 의 상황으로 단순히 소스의 데이터를 목표 시스템으로 넘기기만 하면 된다고 생각하지만 여기에도 복병이 숨어 있다.

IBM 의 경우 EBCDIC 코드를 사용하고 있고, 개방 시스템의 경우 ASCII 코드를 사용하고 있으므로 우선 코드 전환 작업이 선행되어야 하는 어려움이 있다. 또한 사용되는 코드가 다른 부분은 한글에도 같이 적용이 되므로 한글 코드도 변경을 해야 하는데 한글의 경우 소스 데이터에 문제를 갖고 있는 상태에서 전환 역시 더 큰 어려움으로 대두된다. 실제로 IBM 의 한글 데이터를 전환해 보면 종종 한글이 깨진 현상이 발생을 하는데 소스에서 한글이 정확한 SOSI - 한글 on, 한글 off - 작업이 이루어지지 않은 경우가 대부분이다. 문제를 이해하면 해결 방법을 찾기가 쉬운데 불규칙적으로 데이터가 깨지거나 이러한 데이터를 찾기 위해 일일이 데이터를 검색하여 확인해야 하는 어려움이 발생할 수도 있다. 또한 오래된 과거 데이터의 경우 어디에서도 데이터의 구조를 설명하는 자료를 찾기가 어렵다는 것이다.

데이터 이행의 두 번째 방법은 차세대 시스템과 같이 과거 시스템의 구조와는 상관없이 새로운 비즈니스 구조에 맞추어 새로운 데이터베이스를 구축하고 새로운 애플리케이션을 개발하는 것이다. 현재 사용 중인 시스템을 완벽하게 이해하고 새로운 시스템의 데이터베이스 구조도 정확하게 이해한 후 매핑을 통해 데이터를 넘기는 작업을 수행해야 한다. 대부분의 기업들이 프로젝트가 완료된 이후에 시스템과 관련해서 발생하는 변경이나 추가 사항에 대해 실제적으로 사용할 수 있는 문서를 만들어 관리하고 보관하는 곳을 찾기란 쉽지 않기 때문에 결코 쉽지 않은 작업이다. 결국 현행 시스템의 ER-Diagram 을 그리는 것부터 프로젝트를 시작하는 경우가 많다.

또한 어렵게 현행 시스템의 구조를 이행했다고 해도 새로운 시스템에 대한 이해를 다시 해야 한다. 물론 업무를 이해하고 있어 빠르게 진행할 수는 있으나 새로 설계되는 시스템은 새로운 비즈니스를 중심으로 설계되므로 현 시스템을 알고 있는 사람은 오히려 더 이해를 못하는 경우도 발생한다. 가끔은 이행팀에서 새로운 시스템의 문제점을 찾아 주기도 한다.

이외에도 현행과 새로운 시스템간의 함수를 만들어야 하는데 1 대 1 의 함수는 아주 쉽지만 거의 찾아 볼 수 없는 구조이다. 많은 부분이 1 대 N 의 구조를 보이거나 N 대 1 의 구조를 나타내는 경우도 종종 있다. 가장 큰 문제는 N 대 N 의 구조로 여러 개의 소스의 데이터를 모아서 여러 개의 목표로 보내야 하는데 이런 경우 설계하거나 구현하는 작업에 많은 어려움이 따른다.

그러나 이행이 완료되지 않은 상태에서는 새로운 시스템이 아무리 잘 구축되었다고 하더라도 시스템을 사용할 수 없기 때문에 이행은 해야만 한다. 이런 이유로 시스템 구축 기간의 후반부로 넘어 오면서부터 데이터 이행이 가장 큰 이슈로 자리잡게 된다. 주어진 시간 안에 완벽하게 이행을 완료할 수 있는가? 이행된 데이터의 완전성은 보장 받을 수 있는가? 각각의 단계별 문제 발생 시 해결 대책은 강구하고 있는가? 프로젝트가 막바지에 이르면 프로젝트의 성패는 데이터 이행에 달렸다고 할만큼 그 중요성이 확대되기에 이른다.

만일 주어진 시간 동안에 데이터 이행을 완료하지 못했다거나 또는 완료는 했지만 데이터의 완성도가 떨어져 사용하기 어렵다고 판단되는 경우, 모든 비난은 데이터 이행팀으로 몰리게 된다.

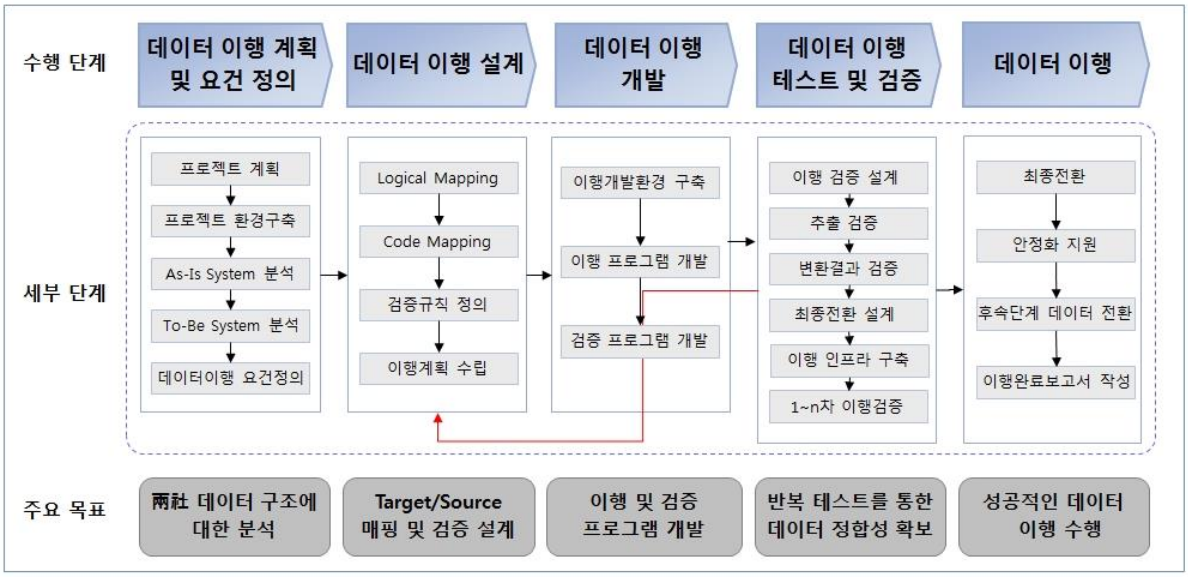
이를 방지하기 위해 한 가지 방안을 제시해 보자면, 기존의 데이터 이행 방식은 현 시스템이 OFF 된 시점에서부터 모든 데이터를 옮기기 시작해서 주어진 시간 동안 모든 데이터를 넘겨야 하는 시간 제약이 있는 업무라고 볼 수 있는데 이 시간 제약을 분산시키는 방법이다.
시간 간격을 두고 업무별로 데이터 이행을 수행하고 이후에 발생하는 변경 정보는 실시간 데이터 통합(RTDI)에서 설명한 변경 데이터 집적(CDC) 기능을 이용하여 새로운 시스템의 데이터와 동기화를 수행하게 된다. 이러한 방법을 순차적으로 수행하면 이행 작업을 수행하기 전에 모든 데이터의 이행을 완료할 수 있다. 또한 사전 점검을 통한 완전성의 확보로 데이터의 무결성도 보장할 수 있다.

예를 들면 현행 시스템이 회계, 인사, 영업, 물류 등 많은 업무를 포함하고 있는 경우 한 주에 한 업무씩 이행을 하는 것이다. 이번 주말에는 회계 시스템 데이터를 이행하고 이 주가 끝나면 그 이후부터 변경되는 정보는 변경데이터집적(CDC) 기능을 통하여 현 시스템의 변경 데이터를 새로운 시스템으로 트랜잭션 단위로 작업을 수행한다. 현 시스템의 데이터를 변경할 때 마다 새로운 시스템의 데이터도 자동으로 변경된다. 개발자들은 그 동안 데이터의 검증이나 프로그램의 검증 작업을 수행하게 된다. 차주에는 다른 업무, 그 다음 주에는 또 다른 업무, 이러한 형태로 작업이 수행되는 경우 최종적으로 전체적인 이행 작업을 수행하기 전에 모든 데이터에 대한 이행 작업이 완료될 수 있는 환경을 만들 수 있게 되는 것이다.

3. 효과적인 데이터 이행 전략

데이터 이행 규칙이란 결국 데이터 집합간의 변환, 이동을 의미하는 것이므로 소스 집합을 알고, 결과 집합을 안다면 그리 어렵지 않게 데이터를 이행 할 수 있다. 데이터

이행은 수많은 예외 경우를 처리하는 것이 아니라 데이터와 데이터간의 집합적인 매핑에 불과한 것이다.



<데이터 이행 수행 절차>

이를 위해서는 현행 시스템과 목표 시스템의 데이터 아키텍처를 명확하게 수립하는 것 말고는 절대 더 좋은 대안이 없을 것이다. 그러나 실전에서 수행되는 대부분의 데이터 이행은 유형별 사례를 무수히 도출한 후에 애플리케이션에서 'CASE-BY-CASE'로 알고리즘을 작성하여 해결하고 있다.

이처럼 애플리케이션에서 수많은 예외 처리를 일일이 지정하려 애쓰지 말고, 규칙을 벗어나는 데이터를 정제하여 규칙을 준수하도록 교정하고, 애플리케이션에서는 단지 규칙을 준수한 집합만을 처리하도록 하는 것이 가장 바람직한 처리 방법일 것이다.

현행 데이터 아키텍처를 수립함으로써 현재의 매우 구체적인 데이터 정보가 리파지토리에 보관되어 있고, 목표 데이터 아키텍처를 수립했다면 그것 또한 리파지토리에 관리되어 있으므로 이를 이용해 명확한 매핑룰을 수립하는 것이 가장 이상적으로 데이터를 이행하는 방법이다.

일반적으로 대다수의 프로젝트에서는 데이터 이행만을 전담하는 조직이 별도로 구성된다. 상식적으로 본다면 그들은 반드시 현행 데이터 구조와 목표 데이터 구조를 가장 정확하게 알아야 한다. 그러나 일반적인 경우 데이터 모델링에 이들이 깊숙이 개입하는 경우는 흔치 않다.

그들에게 나름대로의 방법론이 있을 것 같지만, 사실은 데이터의 유형별 사례를 무수히 도출한 후 애플리케이션에서 'CASE-BY-CASE'로 알고리듬을 작성해 해결하는 것이 대부분이다. 심지어 소스 코드가 2 만 라인이 넘는 경우도 있고, 데이터 처리에 15 일 동안이나 수행되는 애플리케이션을 본 적도 있다.

어떤 시스템은 차세대 시스템이 구축된지 2 년 만에 EDW 프로젝트를 하였는데, 새 시스템이 가동된지 얼마 되지 않았지만, 이런 저런 데이터들이 어지러이 쌓여 있었다. 시스템에 문제가 생기지 않는지 놀라울 정도였다.

데이터의 발생 형태가 조사된 것들은 매핑이 되었지만 그렇지 않은 경우는 애플리케이션의 기타 조건(IF 문의 ELSE 조건)에 의해 지정해둔 기본값(default value)이 들어가 있었다. 처음에는 어떤 날짜 속성이 왜 '01-01-0001'로 되어 있는 데이터가 그렇게 많은지 이해할 수 없었다. 물론 만약 현행과 목표 데이터 아키텍처가 대동소이하였다면 매핑이 단순해지는 것은 당연하다. 이런 경우라면 데이터 이행이 별로 어려울 것이 없다.

시스템을 재구축한다는 것은 기존 데이터 모델이 그리 우수한 것은 아니었다는 반증인 것이다. 천문학적인 비용을 들여서 구축한 시스템의 골격이 과거와 유사했다는 것은 이미 데이터 이행의 문제가 아니다. 문제의 시작은 시스템 설계자와 데이터 이행팀 조직부터 분리돼 있다는 사실이다. 현행 시스템의 데이터 아키텍처를 리버스하는 것부터 서로 힘을 합치는 것이 마땅하다. 먼저 시스템의 골격이 되는 데이터를 구체적으로 정의하고, 그 이후 개발 단계에서 각 목표 시스템의 애플리케이션을 개발하거나 데이터 이행을 위한 애플리케이션을 개발하면 된다. 데이터를 정제하는 일도 데이터 구조를 정확히 이해하는 것에서부터 출발해야 한다. 무엇이 옳은 형태인지 명확히 정의되어야 어떤 것이 잘못된 데이터인지 알 수 있다.

또한 데이터를 이행하는 애플리케이션에다 수 많은 예외 처리를 일일이 지정하려고 애쓰지 말고, 규칙을 벗어나는 데이터를 정제해 규칙을 준수하도록 교정하자. 애플리케이션에서는 단지 규칙을 준수한 집합만을 처리하도록 하는 것이 가장 이상적인 방법이다. 관계형 데이터베이스를 사용한다면 SQL 이 바로 집합을 처리하는 개념으로 되어 있기 때문에 길어도 몇 십 라인의 SQL 로 모든 처리를 하는 것도 가능하다.

일례로 금융권에서 드물게 오픈 시스템으로 다운사이징을 한다고 해서 관심의 초점이 되었던 프로젝트가 있었다. 그런데 오픈 일자가 가까워졌는데도 한 번도 완전하게 모든

데이터를 이행해보지 못했다. 특정 부분을 처리하는 데만 200 시간이 넘게 걸렸기 때문에 결과의 이상 유무 프로젝트 일정에 쫓겼던 것이다.

애플리케이션을 분석해 본 결과 앞서 지적한 유형의 오류를 그대로 반복하고 있었다. 결국 애플리케이션들을 거의 SQL 하나씩으로 대체해 다시 작성함으로써 모든 작업을 5 시간 내에 종료할 수 있었다. 수만 라인에 해당 하는 애플리케이션 소스 코드이 수십 라인의 SQL 로 모두 변경했으니 가능한 일이었다.

이처럼 데이터 이행은 수 많은 예외 경우를 처리하는 것이 아니라 데이터와 데이터 간의 매핑에 불과하다. 이를 위해서는 현행과 목표 시스템의 데이터 아키텍처를 명확하게 수립하는 것이 가장 좋은 대안이다. 데이터 아키텍처를 중심으로 데이터 이행을 수행하는 전략이 수립된다면 데이터의 양이 아무리 많더라도 성공적으로 데이터 이행을 완료할 수 있을 것이다.

데이터 이행 프레임워크

모 증권사의 데이터 이행을 하면서 작성한 데이터 이행 프레임워크를 살펴보면. 데이터 이행의 첫 번째는 ASIS DA 정보와 TOBE DA 정보를 관리하는 것이 출발점이다. 그리고 표준 정보를 참조할 수 있어야 데이터 검증에 잘 활용 할 수 있다.

이렇게 ASIS, TOBE 에 대한 테이블 GAP 관리를 잘하고 난 상태에서 매핑 관리를 시스템을 통하여 하게 된다. 기본적인 매핑 관계를 작성을 하고 난 후에는 이행 룰

설계를 하고 이행 룰 설계를 하고 나면 자동적으로 이행 소스가 생성이 된다. 그리고 이행 결과를 관리 한다. 검증 룰과 JOB 실행 관리, 데이터 수집 대상 정의도 마찬가지이다.

이런 전반적인 부분이 시스템에 의해서 체계적으로 관리가 되어지고 각 단계별 GAP 을 체크 할 수가 있고 설계를 하면 바로 구현 소스가 생성되는 체계를 만들어야지 급변하게 바뀌는 프로젝트의 테이블 수정을 잘 반영해서 갈 수가 있고 컨설턴트는 핵심 로직 설계나 튜닝에 초점을 맞출 수 있게 된다.

데이터 이행시 일정 비율 및 업무 분담

데이터 이행에서 업무 분담은 중요한 부분을 차지한다. 업무 분담만 잘해도 그 프로젝트는 반 이상은 성공했다고도 말할 수 있다.

작은 데이터 이행 프로젝트에서는 이행을 잘하는 한 두 사람만 있어도 별 문제 없이 해결이 되지만 대용량 데이터 및 업무 복잡도가 높은 이행 작업에서는 업무 분담을 잘못하면 몇몇 주요 사람만 매일 야근을 해야 하는 일이 생겨날 수도 있다. PM 입장에서는 이런 일을 방지하기 위해서 일정 및 업무 분담을 잘 해야 하지만 현실적으로 참 어려운 일이다.

데이터 이행을 하면서 아래의 절차를 지키면 그래도 위험에 대해서는 약간 방지를 할 수 있을 것이다. 기간은 리버스부터 이행까지의 일정을 1 년으로, DA 분야를 최소 10 년 이상 경험했을 경우를 가정한 것이다.

*** 리버스 테이블 인당 테이블 부여 기준**

1. 일반적으로 리버스 테이블보다 최종 타겟 테이블 수가 작아지기 때문에 인당 200~250 개 정도가 가장 이상적이다.

2. 최종 타겟 기준으로 보면 인당 100~130 개가 가장 이상적이다.

3. 사이즈는 타겟 기준 인당 500G~1T 정도

4. 특정 사람에게 어려운 업무가 부여될 수 있기 때문에 사이즈 기준으로 일정하게 나누면 복잡도가 비슷하게 배분될 수 있다.

위의 표를 간단하게 설명을 하면 리버스 대상 테이블만 200 개 정도를 업무의 영역을 나눠서 합리적인 수준으로 배분을 해야 한다. 대부분의 사이트는 리버스 대상보다 보통 최종 타켓은 반 정도 줄어든다고 볼 수 있다(집합의 통합, 조사해 보니 사용하지 않는 테이블, 기타 등). 이상적인 것은 최종 타켓이 인당 100~130 개 정도가 그 사람이 계속 모델을 관리할 수 있는 수준이라고 볼 수 있다..

사이즈가 중요한 팩트가 되는 이유는 데이터가 작으면 현실적으로 시행 착오를 많이 해도 금방 다시 수행 및 테스트를 할 수가 있는 반면, 한 테이블의 사이즈가 300G 가 넘는다면 해당 데이터에 대한 생성에 대한 시간 및 테스트에 대한 시간이 많이 소요되기 때문에 어떤 사람이 하루에 10 번 할 수 있는 것을 제대로 한 번도 못하는 경우가 허다하다.

이런 경우에 전체로 테스트를 하지 말고 한 달치 혹은 1/10 정도만 샘플링해서 해당 데이터에 대해서 완벽하게 맞추면 9/10 의 데이터도 99%정도는 맞는다고 볼 수도 있다.

데이터 이행 시 체크 리스트

(1) 동일 회사의 기존 시스템을 차세대 시스템으로 변경하는 경우인가?
- 맞으면, 기존 시스템과 차세대 시스템이 동일한 DB 를 사용하는가? 다른 DB 를 사용하는가?
(2) 합병에 따른 모회사의 시스템에 자회사의 시스템을 통합하는 경우인가?
- 맞으면, 양사가 동일한 DB 를 사용하는가? 다른 DB 를 사용하는가?
(3) TO-BE 시스템의 데이터 모델은 AS-IS 시스템 데이터 모델과 어떤 차이가 있는가?
- 모델의 변경이 거의 없는가? 집합의 통합에 따른 Key 레벨 변경 등 모델의 변경이 많은가?
(4) AS-IS 와 TO-BE 시스템의 모델은 알고 있는가?
- TO-BE 모델, AS-IS 모델을 다 알고 있는 경우
- TO-BE 모델은 알지만, AS-IS 모델은 모르는 경우
- TO-BE 모델은 모르지만, AS-IS 모델은 아는 경우
- TO-BE 모델, AS-IS 모델을 다 모르는 경우

이외에도 여러 가지 다양한 고려 사항들이 있을 것이다. 시스템적인 고려 사항에 대해서는 별도로 다뤄야 될 정도이다. 위와 같은 것들을 생각해 보지도 않고 무턱대고 데이터 이행을 시작한다면 불필요한 데이터만 옮겨질 것이다.

4. 결론 및 시사점

데이터 이행 방법이 그릇되면 데이터 이행을 수행하기 위해 많은 시간이 소요되고 해당 시스템의 서비스 정지 시간 또한 증가하게 되어 그에 따른 비용이 증가하게 된다.

때문에 올바른 데이터 이행 방법이 적용되어야 하며 이행된 데이터에 대해서도 지속적인 품질 관리를 해주어야 한다.

※ 다음 엔코아 리포트에서는 데이터 통합 및 이행 방법론(ETL)에 대해 연구한 자료를 공유하고자 한다.