1. 원장성 테이블(Master Table)

1) 구성형태

업무의 핵심 개체(entity)들로서 주로 업무성격의 "주체(Subject, Source)"에 대한 정보를 담고 있으며, 부속된 많은 자식 테이블(child table)들을 거느리고 있습니다. 또한 이력 테이블(History Table)도 부속되어 있습니다. 그렇기에 dw시스템을 구축할 때도 이와 같은 원장성 테이블은 아주 중요한 핵심 테이블(Key Table) 역할을 하게 됩니다.

여기서 주체(Source)란 행위(Behavior)의 주체를 말합니다. 쉽게 예를 들어 설명하면 모든 업무의 프로세스(process)는 주체와 행위, 그리고 대상(Target, Object)로 집약될 수 있습니다. 즉, "홍길동은 카드로 멋진 컴퓨터를 구입했다"라는 내용을 업무 프로세스 관점에서 바라보면 [홍길도]과 [카드]는 주체로서 각각 사람주체와 사물주체를 말하는 것이고, [구입했다]는 행위이며, [컴퓨터]는 대상이 되는 것입니다.

전산 설계 관점에서 바라보면, [홍길동]은 고객정보 마스터 테이블이 될 것이고, [카드]는 카드정보 마스터 테이블이 될 것이며, [구입했다]는 거래내역 슬레이브 테이블이 될것이고, [컴퓨터]는 주문상품정보 대상 테이블이 될 것입니다.

2) 데이터 성격

현재(가장 최신)의 상태와 속성정보를 유지, 관리합니다. 가령 고객정보 마스터 테이블을 예로 들면, [홍길동]이란 사람은 많은 속성을 가지고 있고 늘 최신의 정보상태를 유지합니다. 속성들도 시간이 흘러도 변하지 않는 불변 속성과 시간의 경과에 따라 상태가 변하는 가변 속성이 존재합니다. "생년월일", "주민번호", "성별" 등은 불변속성이기에 처음 올바로 등록했다면 그 값은 절대로 변하지 않습니다. "건강상태", "주소", "전화번호" 등은 가변속성이기에 처음 등록 이후, 계속 변하게 됩니다. 따라서 최신의 정보를 늘 유지하기 위해 관리되는 항목이며, 이러한 가변속성들을 관리하기 위해 이력 테이블(History Table)이 존재하게 됩니다.

그럴리야 없겠지만, 원장성 테이블에 부속된 이력 테이블이 없다면, 많은 문제점이 노출될 수 밖에 없으며, 이는 DW설계시 보다 섬세한 여러 가지가 고려되어야 합니다.

2. 거래성 테이블(Transactional Table)

1) 구성형태

"행위(behavior)"테이블을 말하는 것으로, "카드거래내역 테이블, 입출금내역 테이블" 등이 해당되겠습니다.

2) 데이터 성격

원칙적으로 거래성 테이블은 등록만 되는 개념이지, 변경되는 개념이 아닙니다. 즉, "카드거래내역 테이블"을 본다면, 카드 승인시에 승인기록이 저장되고, 이를 정정하기 위해서는 변경이 발생하는 것이 아니라, 취소기록을 등록하게 됩니다. 어떠한 경우도 한번 등록된 내용을 변경하는 경우는 없습니다. 따라서 이력 테이블은 존재하지 않습니다.

3. 양면성 테이블(Duplicity Table)

1) 구성형태

예를 들면 카드를 발급받기 위해서는 카드신청서를 작성하듯 이 프로세스를 전산 시트메에 구축하게 되면 "카드신청 테이블"이 존재하게 될텐데요. 신청이라는 행위의 활동이므로 이는 분명 거래성 테이블로 분류될 수 있는데, 온전한 거래성 테이블로 보기 힘든 이유가 신청서를 잘못 작성했다고 동일한 내용을 수정(update)하는 경우가 발생하기 때문입니다. 즉, 신청시 주소가 틀렸다고 수정하고, 영문이름이 틀렸다고 수정하고 한다는 것입니다. 따라서 이런 특성 때문에 이력 테이블을 거느리는 형태를 띄게 됩니다. 꼭 이력 테이블이 존재한다는 의미보다는 설계 당시 이력 테이블이 고려되지 않았다면 실제로 이력테이블이 없을 수도 있겠습니다.

신청이란 행위의 정보를 저장한다는 측면에서 거래성 테이블(Transactional Table)의 성격을 가지고 있고, 변경이 발생하고 이력 테이블을 거느리고 있다는 측면에서 원장성 테이블(Master Table)의 성격을 가지고 있는 것입니다. 따라서 업무를 분석할 때, 양면성 테이블인지, 거래성 테이블인지 정확한 파악이 필요하다 하겠습니다.

2) 데이터 성격

원장성 테이블

최신의 상태정보를 유지하고 있다고 봐야합니다. 다만 거래성 테이블의 특성을 겸하고 있으므로 이 테이블의 키값(key value) 구성은 [신청번호]가 될 수 밖에 없습니다.

4. 이력 테이블(History Table)

1) 구성형태

이력 테이블은 원장성 테이블(master table)의 변경된 상태 속성 정보를 보관하는 역할을 합니다. 원장성 테이블의 외부에 동일한 구조를 가지고 있는 외부 이력 테이블(Outer History Table) 형태가 있을 수 있고 원장성 테이블의 내부에 상위키(upper key) 속성칼럼을 포함시켜 내부순환(recursive) 구조를 가진 내부 이력 테이블(Inner History Table) 형태가 있을 수 있으며, 원장성 테이브르이 중요 상태정보만을 이력관리하는 중요항목 이력 테이블(Hot-Item History Table) 구조로 구성될 수 있습니다.

2) 데이터 성격

변경된 상태정보를 시작시점~종료시점의 구간으로 하는 선분이력(Line History) 형태로 관리될 수도 있고 변경시점을 기준으로 하는 점 이력(Point History) 형태로 이력은 관리될 수 있습니다. 선분 이력 테이블(Line History Table)은 주로 원장성 테이블에 구축되며 점이력 테이블(Point History Table)은 주로 양면성 테이블에 일반적으로 구축됩니다.

이상으로 짧게나마 테이블의 종류와 특성을 알아보았습니다. 원천 테이블의 종류와 특성을 정확히 파악하는 것은 시계열(time serial)로 구축될 대상이 무엇인지, 시계열 간격(일, 월, 년)은 어떻게 할것인지 또한 이력 테이블의 구축이 필요한 대상이 무엇인지, 스냅샷(snapshot history)이력으로 할것인지, 선분이력(Line History) 또는 점이력(Point History)로 구축할 것인지 결정하는 것이 아주 중요한 작업입니다.

원칙적으로 원장성 테이블일 경우 시계열로 구축이 될 것이고 아울러 기간계의 이력테이블이 없거나 부실하다면 DW시스템에 새로 이력테이블을 구축해야 할 것입니다. 또한 원장성 테이블의 특성에 따라서 스냅샷 이력 테이블로 구축할지 선분이력 테이블로 구축할지도 고려되어야 합니다.

거래성 테이블은 DW 구축시 그냥 1:1로 가져온다고 생각하시면 될 것 같습니다. 양면성 테이블은 해당 프로젝트 사이트 업무 환경에 따라 그 설계방식이 차이가 발생할 수 있습니다. 시계열로 구축이 필요한지 아닌지 여러 각도로 살펴보아야 합니다. 왜냐하면 거래성 테이블 성격도 가지고 있기 때문에 데이터 크기가 상당히 큰 경우가 대부분입니다. 따라서 효율적으로 설계하지 못하면 ETL시 많은 부하와 적재시간이 소요됩니다. 양면성 테이블의 경우 이력이 필요할 경우 주로 점이력 테이블을 구축하게 됩니다.