

[Data Scientist와 AI Engineer를 위한 데이터 엔지니어링]

Module 1. 데이터 엔지니어링 개요

Contents

I. 개요

1. 데이터 직무 구분
2. 데이터 엔지니어 역할
3. 데이터 엔지니어 업무
4. 데이터 엔지니어링 역량 개발 경로

II. 업무 논리 구조

III. 방법론

1. 데이터 직무 구분

I. 개요

데이터 (Data Scientist) 직무는 데이터 플랫폼 엔지니어링, 데이터 엔지니어링, 데이터 분석 등으로 직무 분야를 구분함

데이터 시장 구분¹⁾

대구분	중구분	예시 및 관련회사
데이터 솔루션	데이터특정영역 특화솔루션	<ul style="list-style-type: none"> NoSQL Spotfire, BI Matrix
	데이터 통합/분석솔루션	<ul style="list-style-type: none"> IBM, SAS Brightics AI, Accu Insight+
데이터 구축/컨설팅	데이터구축	<ul style="list-style-type: none"> SI구축
	데이터컨설팅	<ul style="list-style-type: none"> 플랫폼구축전략 분석과제도출
데이터 서비스	데이터분석 서비스	<ul style="list-style-type: none"> 컨텐츠추천 부띠크업체
	데이터거래	<ul style="list-style-type: none"> 데이터판매 NICE

데이터 사업영역 및 직무구분

데이터 사업영역	DS 직무분야
데이터 플랫폼	데이터 플랫폼 아키텍처링 <ul style="list-style-type: none"> 표준 솔루션 기반으로 실제 솔루션을 최적으로 설치 및 설정함 플랫폼의 공통 기능/모듈을 개발하여 배포함
	데이터 엔지니어링 <ul style="list-style-type: none"> 데이터 파이프라인의 논리 및 아키텍처를 설계하고 구현함(데이터수집, 처리, 저장) 사용자/분석가들이 활용할 수 있는 데이터를 생성하고 제공(Dataset, Visualization)함 분석모델을 운영시스템에 배포하여 Legacy시스템에 적용함
데이터 분석	데이터 분석 <ul style="list-style-type: none"> 고객의 문제를 도출 및 정의하고, 분석모델을 이용하여 분석과제를 수행함
	사업개발 <ul style="list-style-type: none"> 데이터 플랫폼 및 분석 사업 기회를 모색하고 발굴함

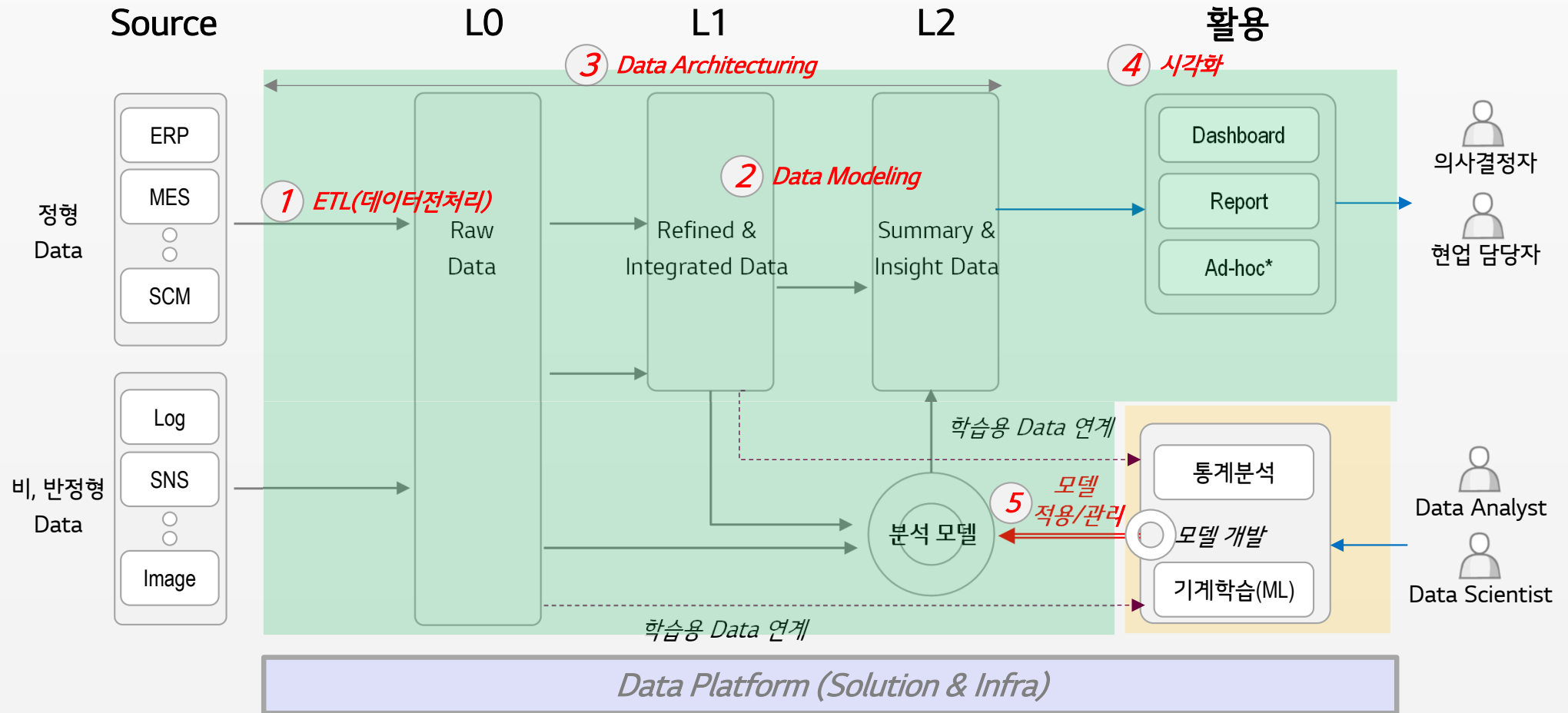
ED사업담당보유직무

직무분야	역할	수행 업무
데이터 플랫폼 아키텍처링	솔루션아키텍트	<ul style="list-style-type: none"> • 데이터 플랫폼을 구성하는 솔루션 선정/검증 및 아키텍처 구성 • 데이터 플랫폼 활용 프로그래밍/모델링에 대한 가이드(개발 표준, 템플릿 코드) 제공 및 Inspection • 데이터 마이그레이션/백업/복구 방안 수립 및 가이드
	테크니컬아키텍트	<ul style="list-style-type: none"> • 데이터 플랫폼 환경구성 및 TCO산출, 성능 최적화, Trouble Shooting • 데이터 플랫폼 환경운영 및 유지보수
데이터 엔지니어링	시각화전문가	<ul style="list-style-type: none"> • 시각화 솔루션을 활용해 분석화면을 설계, 개발
	ETL(데이터전처리)전문가	<ul style="list-style-type: none"> • 데이터 수집, 처리, 저장 프로그램 및 배치 설계/개발 • 데이터 수집, 처리, 저장 프로그램 실행 및 운영
	데이터모델러	<ul style="list-style-type: none"> • 데이터 모델링 및 품질검증 • 성능 최적화 수행
	데이터아키텍트	<ul style="list-style-type: none"> • 데이터 거버넌스 수립(메타 데이터, 데이터 품질, 데이터 life-cycle) • 데이터 파이프라인 개념적, 논리적, 물리적 설계 • 데이터 플랫폼에 분석모델배포 설계
데이터 분석	데이터분석가	<ul style="list-style-type: none"> • 비즈니스 이슈 정의 및 분석과제를 발굴 • 문제 해결 위한 분석 수행, 가이드 제시 • 비즈니스에 대한 이해를 바탕으로 분석 결과에 대한 비즈니스 인사이트 도출
	데이터사이언티스트	<ul style="list-style-type: none"> • 비즈니스의 특성을 수식화하여 비즈니스에 적용 • 통계모델, 머신러닝, 프로그래밍(R, Python, scala등)을 이용하여 데이터 분석 모델 개발 및 검증 • 데이터를 변환해 시각화하여 결과 적용

3. 데이터 엔지니어링 업무

I. 개요

데이터 엔지니어링 업무는 ① ETL(데이터전처리), ② Data Modeling, ③ Data Architecturing, ④ 시각화, ⑤ 모델배포가 있음



데이터 플랫폼

데이터 엔지니어링

데이터 분석

① ② ③ ④ ⑤ 데이터 엔지니어 역할

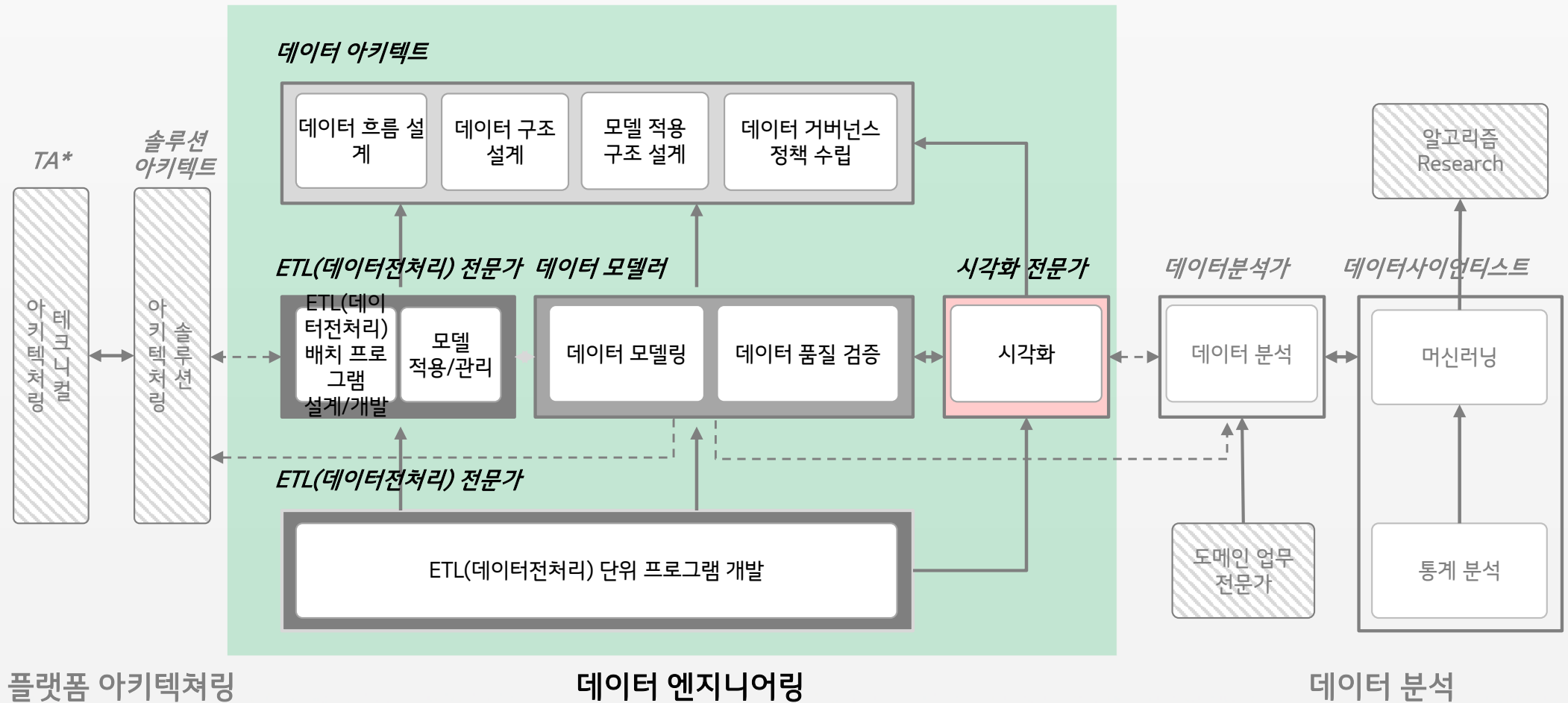
ETL(데이터전처리) 학습용 Data 연계 모델 적용/관리 사용

분석 결과 데이터를 생성하기 위하여 분석모델을 운영환경에서 동작시키는 작업
(ex. 매일 07시, 제품 품질 분석 결과 생성)

4. 데이터 엔지니어링 역량 개발 경로

I. 개요

데이터 엔지니어 직무는 4개 역할간 역량개발이 가능하고, 플랫폼 아키텍트 및 데이터분석가 직무로 전환도 가능함



* TA - Technical Architect

ED사업담당 미보유 직무

Contents

I. 개요

II. 업무 논리 구조

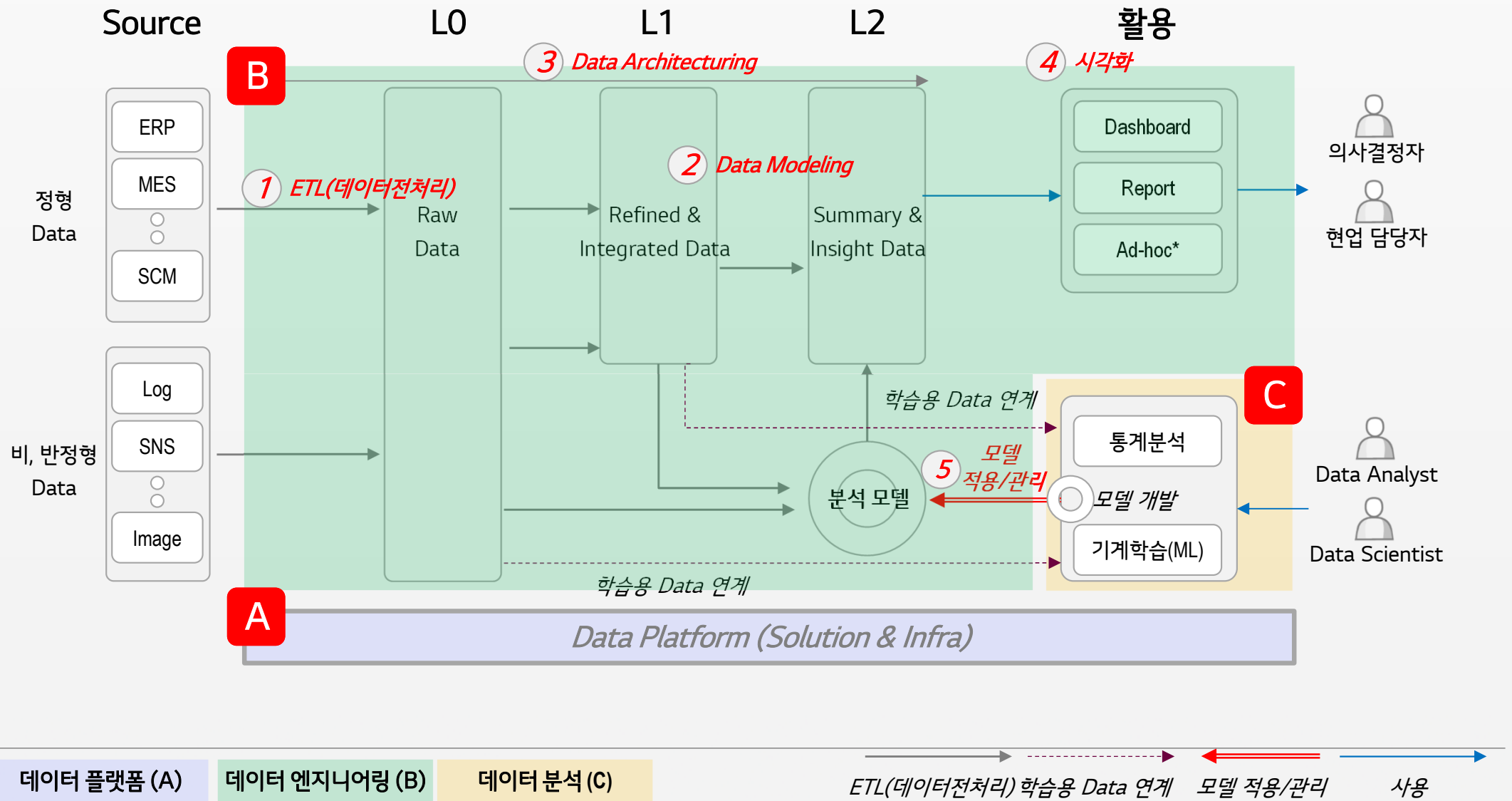
1. 업무 Map
2. 데이터 플랫폼
3. 데이터 엔지니어링
4. 데이터 분석

III. 방법론

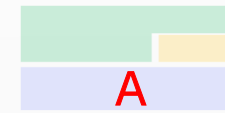
1. 업무 Map

II. 업무 논리 구조

데이터 엔지니어는 본연의 역할(B)과 플랫폼(A) 및 데이터 분석(C)의 업무에 관여함



2. 데이터 플랫폼 (A)

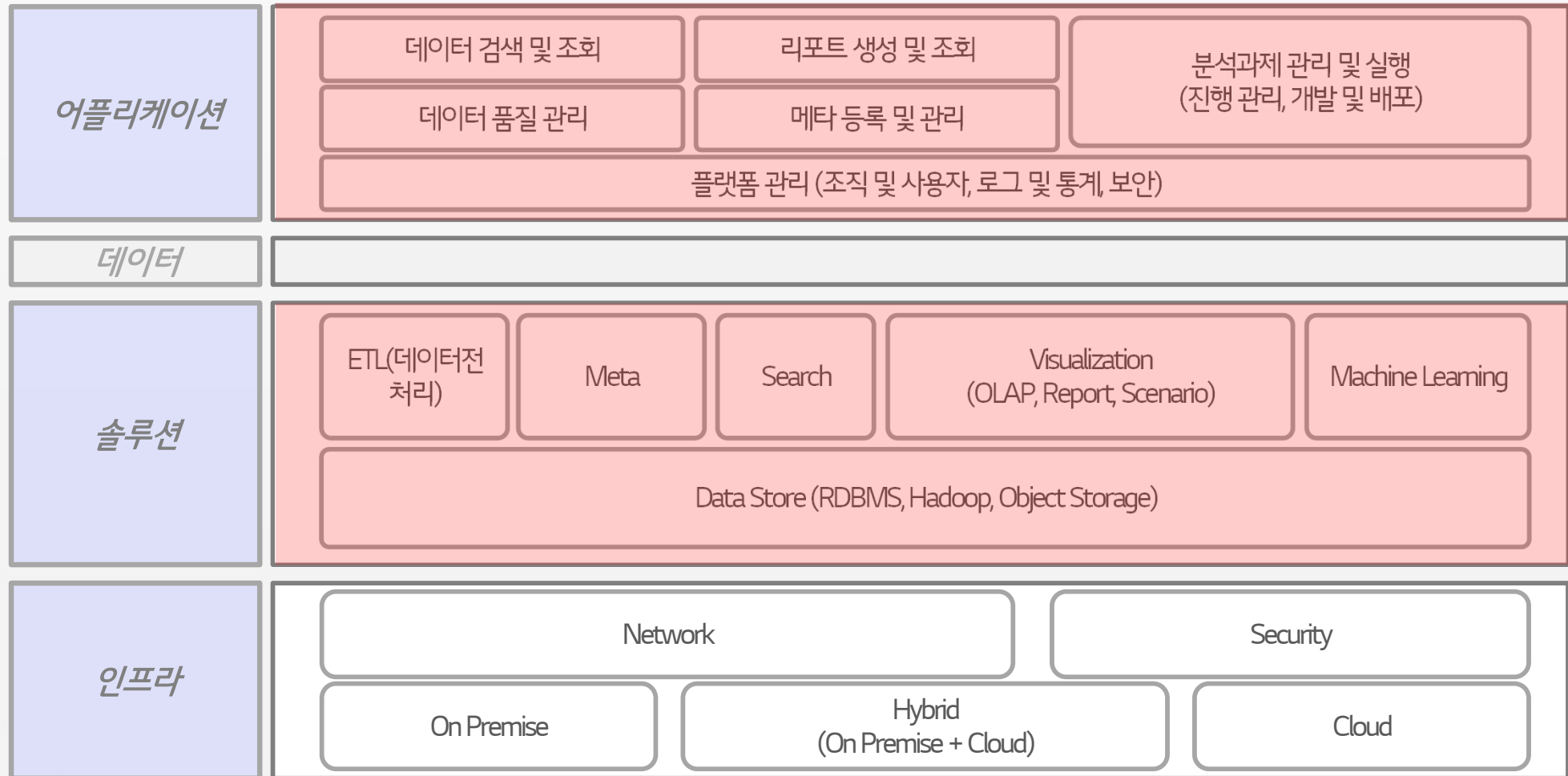


II. 업무 논리 구조

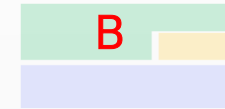
데이터 플랫폼의 논리 구조는 인프라-솔루션-데이터-어플리케이션으로 구성함. 데이터 엔지니어는 플랫폼이 제공하는 솔루션을 이용하여 데이터의 구조를 설계하고, 적재-제공하는 역할을 주로 수행함

■ 데이터 플랫폼 논리 구조

Data Engineer 관련 업무



3. 데이터 엔지니어링 (B)

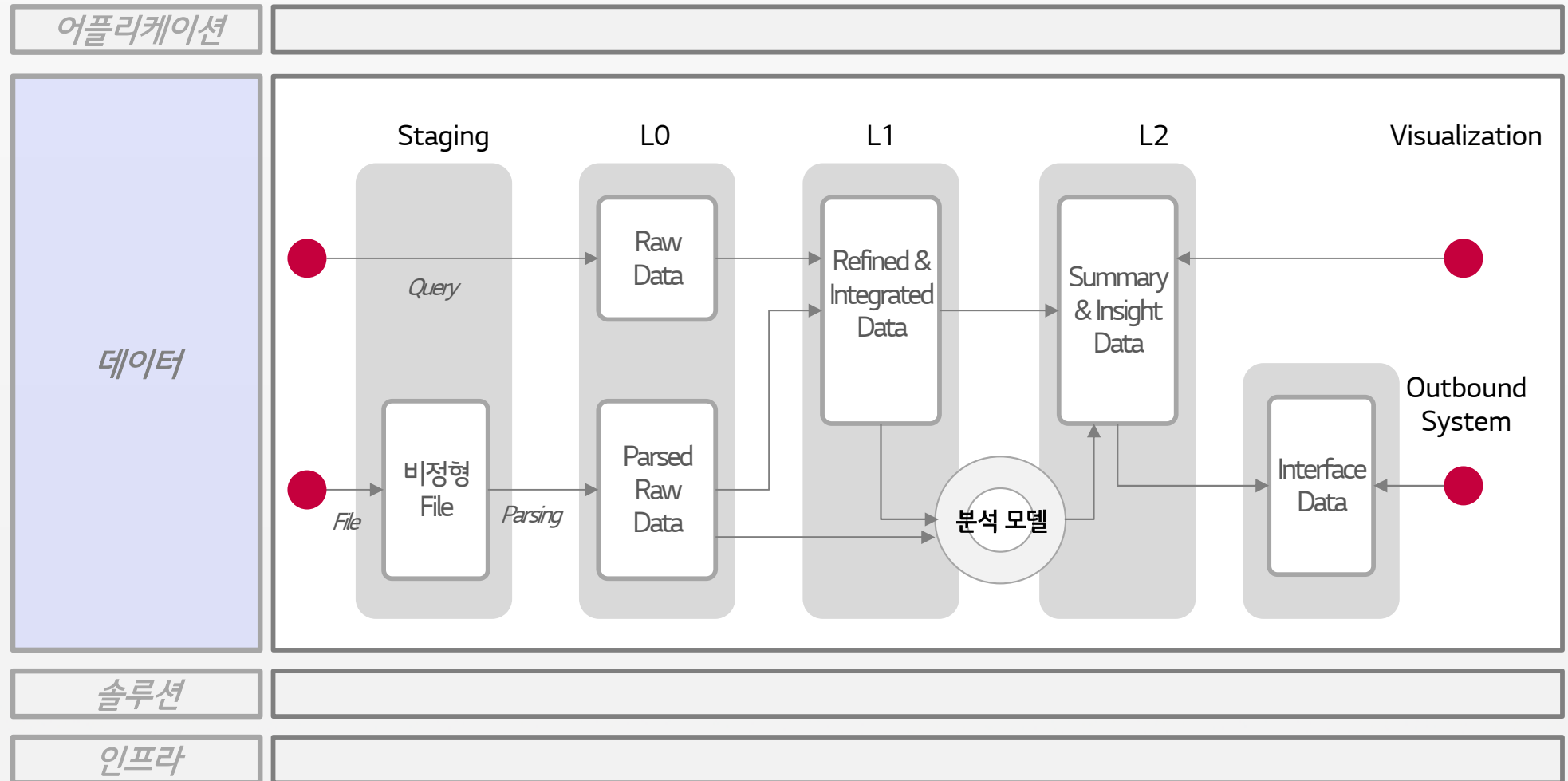


II. 업무 논리 구조

데이터 엔지니어 본연의 역할(B)은 데이터 플랫폼에 데이터를 잘 활용할 수 있도록 구조화하여 적재하고 제공하는 역할임

■ 데이터 플랫폼 논리 구조

Data Engineer 관련 업무

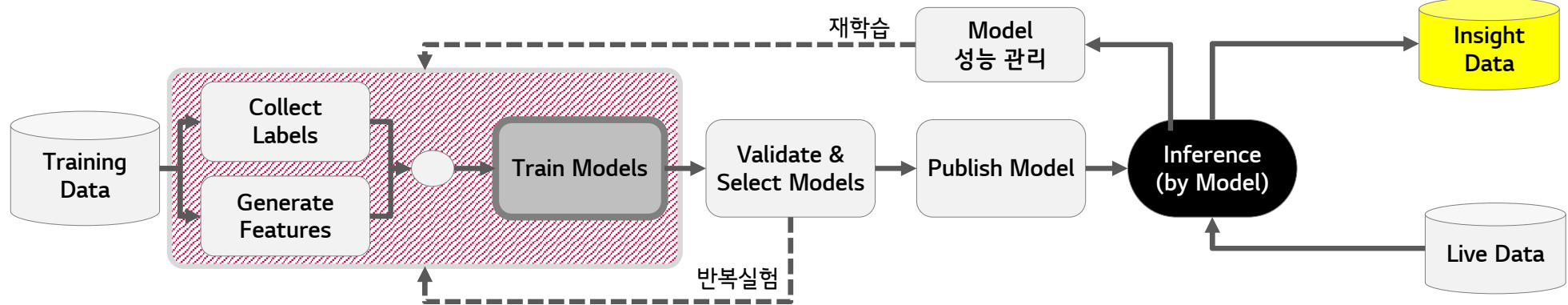


■ 분석 결과 데이터 제공 - 대량 데이터 분석

Training (학습)

분석 모델 실행

수행 프로세스



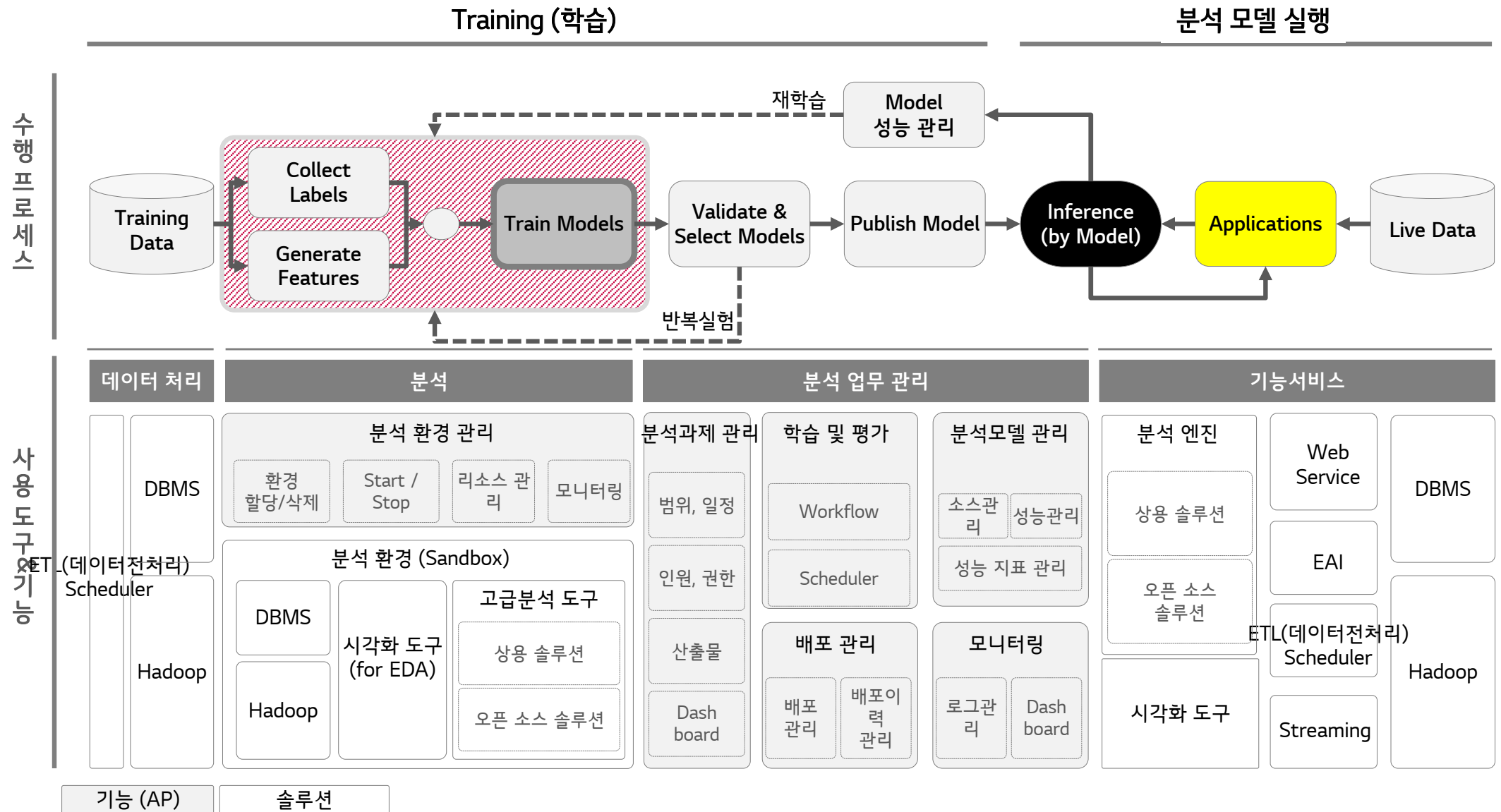
사용 도구 기능



기능 (AP)

솔루션

■ 분석 추론 (Inference) – Transaction 건 단위 실행



Contents

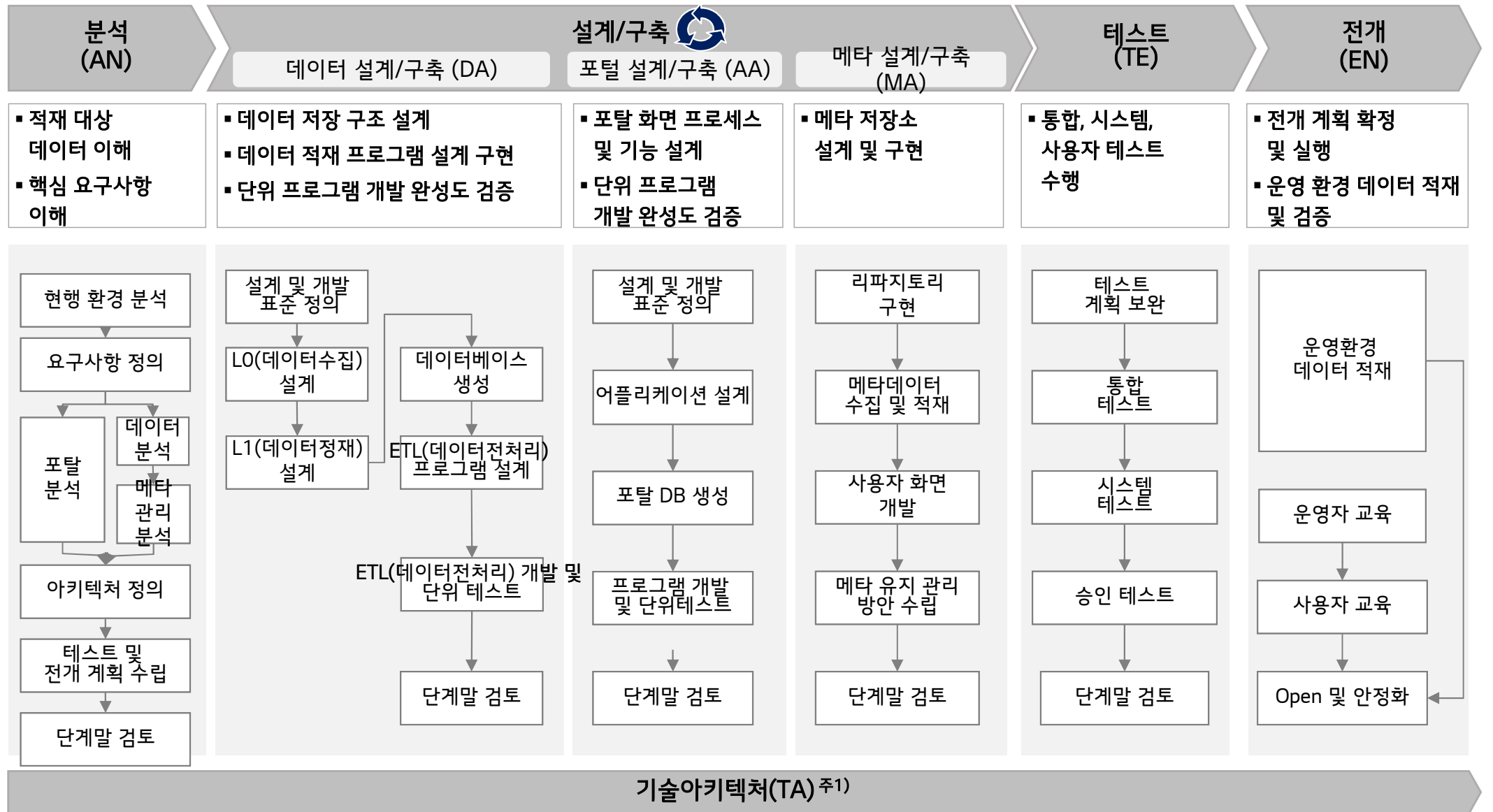
I. 개요

II. 업무 논리 구조

III. 방법론

1. 데이터 플랫폼 구축
2. 데이터 분석 기획 및 실행
3. BI/DW

■ 전체 방법론은 Way4U에서 다운로드 가능



주1) 데이터 플랫폼의 기술 아키텍처는 “인프라, 데이터, 솔루션, 보안”을 포함하며 프로젝트 이행 시 기술아키텍처 경로를 테일러링(tailoring) 하여 활용한다.

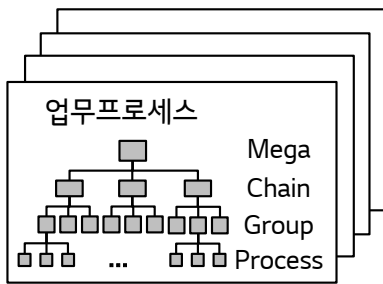
■ 분석 기획

■ 전체 방법론은 Way4U에서 다운로드 가능

100 Understand

사업 방향 및
분석 일반현황의 이해

• 분석 일반 현황



200 Discover

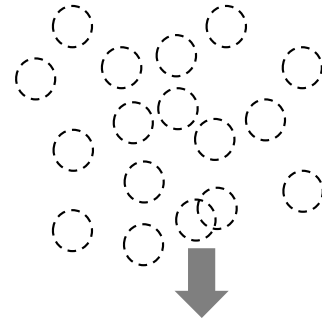
분석 기회 발굴 및 평가

• Biz Pain Point

From Process, 인터뷰

• 분석 기회 발굴

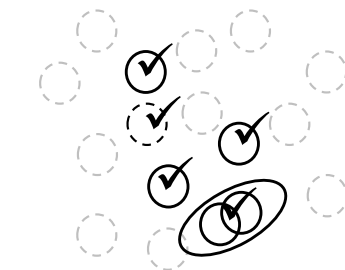
예시) Long List 40 ~ 50개



• 분석 주제 선별

분석가능성 판단, Grouping
분석기회 평가

→ 예시) Short List 10 ~ 15개

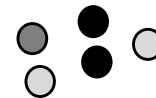


300 Analyze & Assess

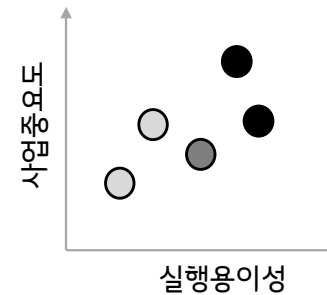
분석 기회 분석 및 선정

• Data Readiness 분석, 분석 주제 조정(구체화) → 분석과제로 선정

Business Requirement
→ Data Analytics Requirement



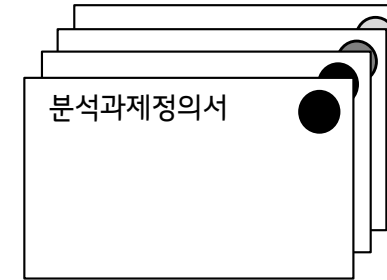
• 추진 우선순위 선정



400 Design

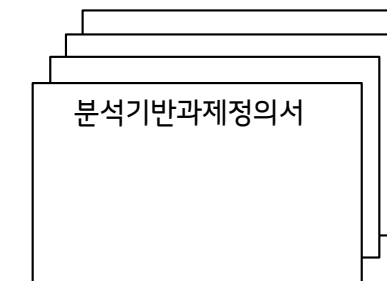
분석 및 분석기반
추진 과제 설계

• 분석과제 설계



• 분석 기반 과제 설계

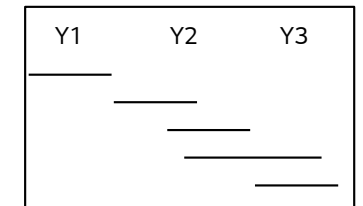
Data, IT Sys, 운영 및 관리체계,
조직/역량 측면



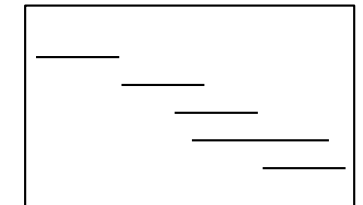
500 Plan

중장기 Roadmap 및
단기 실행계획 수립

• Roadmap 수립

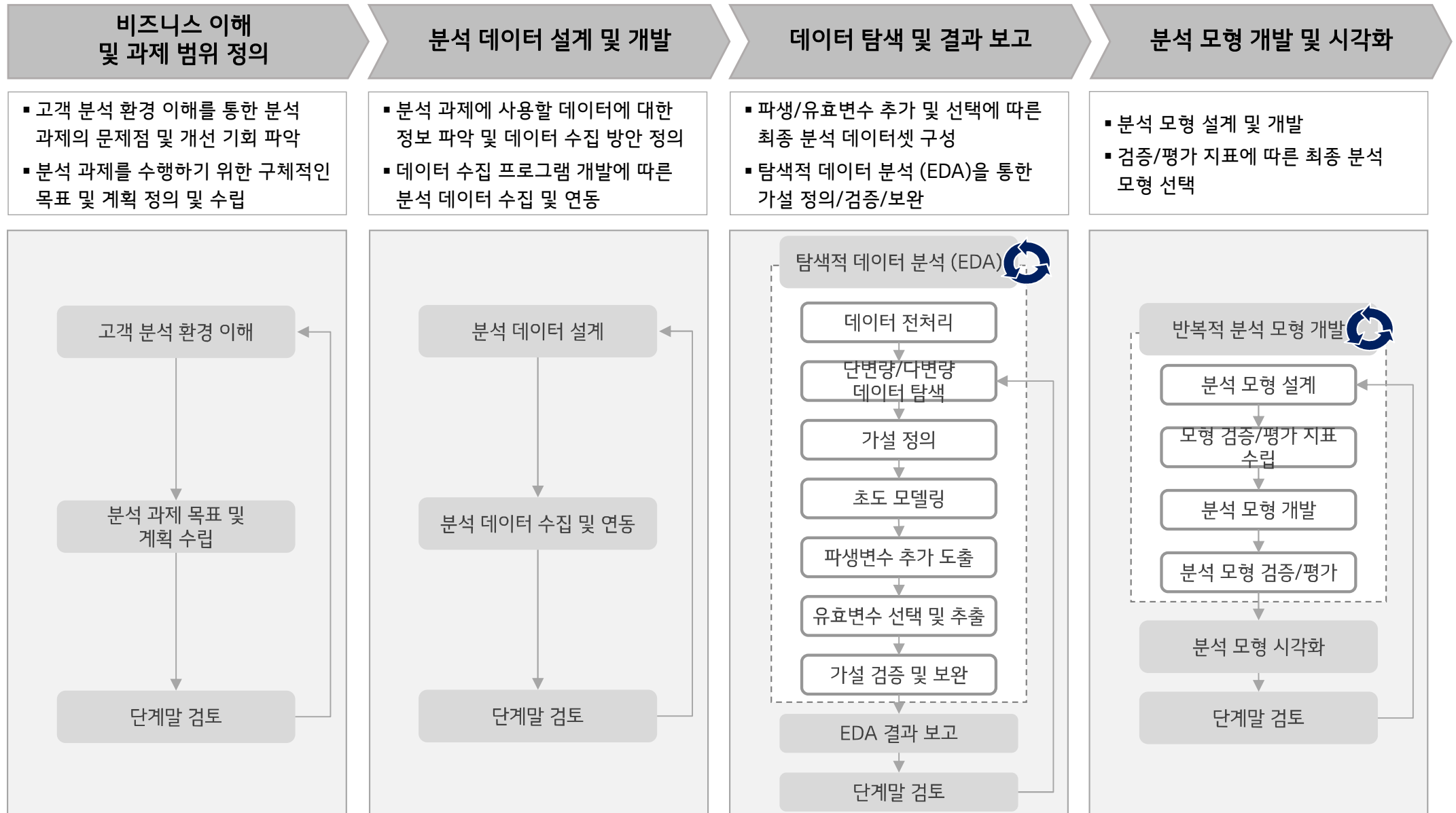


• 활용시나리오 수립

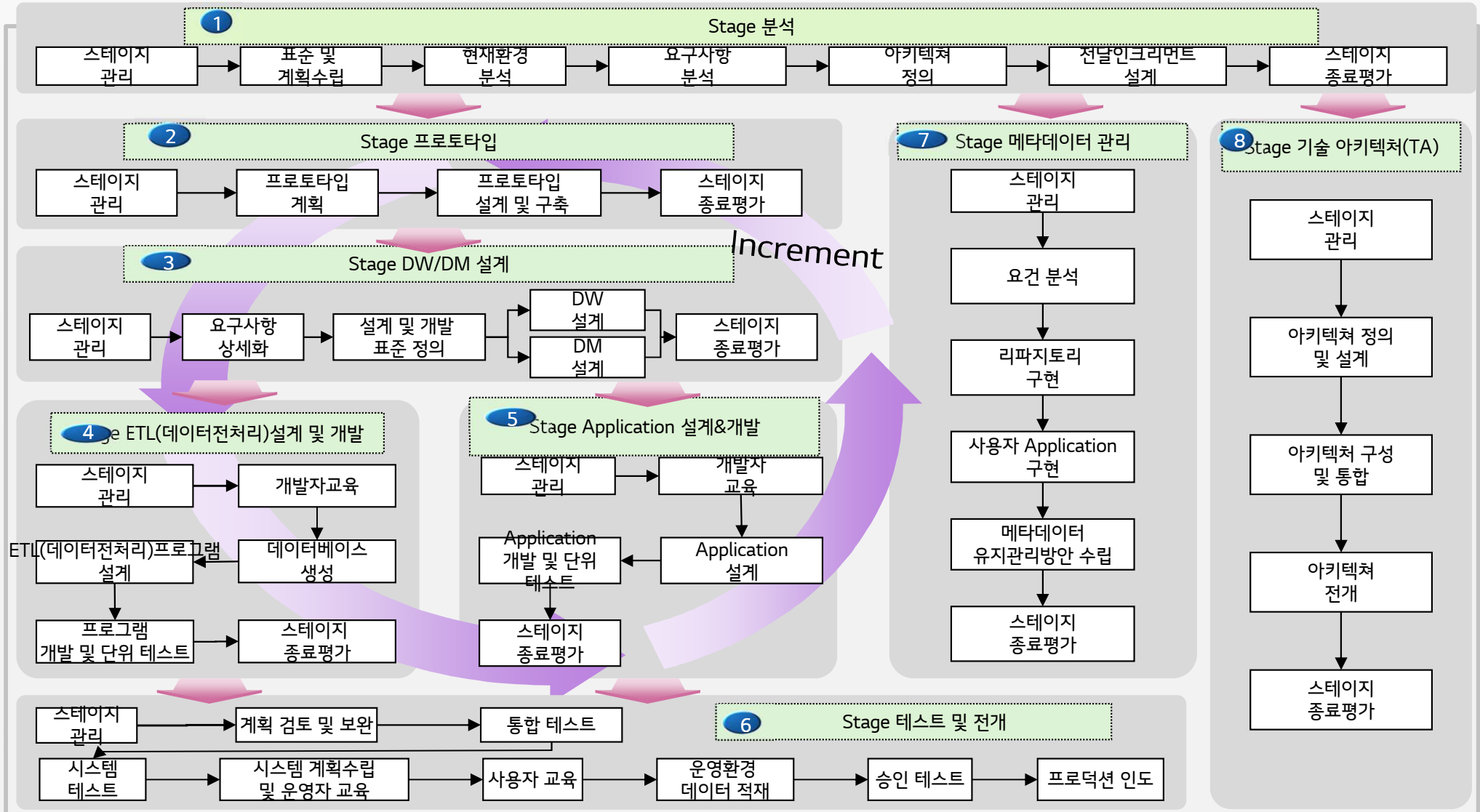


■ 분석 실행

■ 전체 방법론은 Way4U에서 다운로드 가능



BI/DW의 구축 방법론은 8개의 Stage로 구성되며, 소프트웨어 개발 프로세스로 Waterfall 방식 또는 Incremental 방식으로 구축이 이루어짐



8개의 Stage로 구성되며, 교육 과정 내 상세 내용에서 해당되는 Stage 별 구축 내용을 상세화함

▪ Stage 설명

Stage	상세 설명
Stage IP : 분석	<ul style="list-style-type: none"> 정보 사용자들의 요구사항을 정의(기능/비 기능 요구사항), BI/DW의 Scope 정의 Reverse-Engineering기법 등을 사용하여 현행 시스템 분석 비즈니스 솔루션을 어떻게 구현할지에 대해 상세히 기술한 전달 계획서를 작성
Stage DWP : 프로토타입	<ul style="list-style-type: none"> 프로토타입을 통해 사용자 요구사항을 최종 검증 PoC등을 사용하여 기술 요소 검증(성능,기술 요소들의 Integration 등)
Stage DWD : DW/DM 설계	<ul style="list-style-type: none"> BI/DW의 Scope에 따른 상세 정보 요구사항을 이해 DW/DM 영역의 목적에 부합하는 데이터 구조를 설계
Stage DMT : ETL(데이터전처리) 설계 및 개발	<ul style="list-style-type: none"> 데이터를 DW/DM로 변환, 적재 위한 설계 및 구축
Stage DSA : Application 설계&개발	<ul style="list-style-type: none"> Business Intelligence and Analytics Platform 도구를 사용하여 설계 및 개발
Stage IMP : 테스트 및 전개	<ul style="list-style-type: none"> 전체 시스템 운영 환경 이관 및 이행 계획 수립 및 이행 통합 테스트 및 시스템 테스트, 사용자/운영자 테스트 수행 사용자 교육/운영자 교육 수행 인수인계 시험통한 운영 이관 및 오픈 후 시스템 안정화 활동
Stage MM : 메타데이터 관리	<ul style="list-style-type: none"> 업무 메타 및 기술 메타에 대한 구축 전략 및 이행, 안정화를 위한 단계
Stage TA : 기술 아키텍처(TA)	<ul style="list-style-type: none"> BI/DW 기술 기반구조의 계획 수립 및 이행 데이터 관리 프로세스 및 표준 정의

▪ Stage별 주요 Activity 및 산출물 (1/5)

Stage	주요 Task	주요 Activity	주요 Output
Stage IP : 분석	표준 및 계획수립	<ul style="list-style-type: none"> 산출물 목록 및 Template정의 용어표준화 계획수립 	<ul style="list-style-type: none"> 교육계획서 총괄테스트 계획서
	현재 환경 분석	<ul style="list-style-type: none"> 현행 시스템 자료수집 현행 시스템 프로세스 식별 데이터 원천자료 식별 주제영역 데이터모델 검토 현행 의사결정 지원환경분석 개선과제 도출 	<ul style="list-style-type: none"> 현행시스템 분석서
	요구사항 분석	<ul style="list-style-type: none"> 요구사항 수집/ 요구사항 분석 주제영역 정의 	<ul style="list-style-type: none"> 요구사항 정의서 주제영역 정의서
	아키텍처 정의	<ul style="list-style-type: none"> 주요 데이터원천자료 식별 데이터 아키텍처 정의 메타데이터 아키텍처 정의 의사결정지원 아키텍처 정의 보안 아키텍처 정의 	<ul style="list-style-type: none"> 개념 아키텍처 정의서
	단계별 구축 계획 수립	<ul style="list-style-type: none"> 단계별 구축 계획 수립 	<ul style="list-style-type: none"> 단계별 구축 계획서
Stage DWP : 프로토타입	프로토타입 계획 수립	<ul style="list-style-type: none"> 프로토타입 범위 정의/ 계획 수립 	<ul style="list-style-type: none"> 프로토타입 계획서
	프로토타입 설계 및 구축	<ul style="list-style-type: none"> 프로토타입 설계 프로토타입 구축 테스트 수행 프로토타입 시연 프로토타입 결과 검토 및 보완 	<ul style="list-style-type: none"> 프로토타입 설계서 프로토타입 결과서

▪ Stage별 주요 Activity 및 산출물 (2/5)

Stage	주요 Task	주요 Activity	주요 Output
Stage DWD : DW/DM 설계	개념 설계	<ul style="list-style-type: none"> • 디멘전 정의/ 디멘전 계층 정의 • 팩트 정의 • 팩트 디멘전 매트릭스 작성 	<ul style="list-style-type: none"> • 디멘전 정의서 • 디멘전 계층 정의서 • 팩트 정의서 • 버스 아키텍처 정의서
	설계 및 개발표준 정의	<ul style="list-style-type: none"> • 데이터모델링 표준정의 • ETL(데이터전처리)개발표준정의 • Application개발표준정의 	<ul style="list-style-type: none"> • 데이터모델링 표준 정의서 • DB Naming표준정의서 • ETL(데이터전처리)개발표준정의서 • Application개발표준정의서
	DW 설계	<ul style="list-style-type: none"> • 원천자료 데이터목록 작성 • ODS논리모델설계 • DW논리모델설계 • 코드설계 • 물리모델설계 • DW용량산정 • 추출 및 로드전략 개발 • DW설계 검토 및 보완 	<ul style="list-style-type: none"> • ODS/DW 모델(ER/다차원) • 코드설계서 • 용량산정서
	DM 설계	<ul style="list-style-type: none"> • DM논리모델설계 • DM용량산정 • 추출 및 로드전략 개발 • DM물리모델설계 • DM설계 검토 및 보완 	<ul style="list-style-type: none"> • DM 모델(다차원) • 코드설계서 • 용량산정서

▪ Stage별 주요 Activity 및 산출물 (3/5)

Stage	주요 Task	주요 Activity	주요 Output
Stage DMT : ETL(데이터전처리) 설계 및 개발	개발자 교육	<ul style="list-style-type: none"> 분석, 설계단계 작업내역 공유 솔루션 교육/ 개발 표준 교육 	
	DW/DM 데이터베이스 생성	<ul style="list-style-type: none"> ODS/DW 데이터베이스 생성 DM 데이터베이스 생성 인덱스 생성 	
	ETL(데이터전처리)프로그램 설계	<ul style="list-style-type: none"> 추출 주기 및 방법 결정 매핑 흐름 정의 매핑 정의서 작성 	<ul style="list-style-type: none"> 매핑 흐름도 프로그램 목록 매핑 정의서
	ETL(데이터전처리)프로그램 개발 및 단위테스트	<ul style="list-style-type: none"> 단위테스트 계획수립 개발 및 단위테스트 선 후행 프로그램 식별 배치 프로그램개발 및 단위테스트 	<ul style="list-style-type: none"> 단위테스트 결과서
Stage DSA : Application 설계&개발	개발자 교육	<ul style="list-style-type: none"> 분석, 설계 단계 작업내역 공유 솔루션 교육 개발 표준 교육 	
	Application 설계	<ul style="list-style-type: none"> 사용자 및 권한정의 Application 화면 설계 Application 메뉴 구성 아키텍처&메타데이터 영향도 검토 	<ul style="list-style-type: none"> 사용자 및 권한 정의서 화면 설계서 프로그램 목록 메뉴 구성도
	Application 개발 및 단위테스트	<ul style="list-style-type: none"> 단위테스트 계획수립 OLAP 메타데이터 생성 Application 개발 및 단위테스트 	<ul style="list-style-type: none"> 단위테스트 결과서

▪ Stage별 주요 Activity 및 산출물 (4/5)

Stage	주요 Task	주요 Activity	주요 Output
Stage IMP : 테스트 및 전개	계획검토 및 보완	<ul style="list-style-type: none"> 교육계획 검토 및 보완 테스트 계획 검토 및 보완 전개 계획 검토 및 보완 	
	통합 테스트	<ul style="list-style-type: none"> 통합 테스트 계획수립 테스트 시스템 구축 테스트 준비/ 테스트 실행 및 결과기록 테스트 결과 검토 및 보완 	<ul style="list-style-type: none"> 통합테스트 계획서 통합테스트 결과서
	시스템 테스트	<ul style="list-style-type: none"> 시스템 테스트 계획수립 테스트 시스템 구축 테스트 준비/테스트 실행 및 결과기록 테스트 결과 검토 및 보완 	<ul style="list-style-type: none"> 시스템 테스트 계획서 시스템 테스트 결과서
	시스템 관리 계획수립 및 운영자 교육	<ul style="list-style-type: none"> 시스템 운영방안 수립 운영절차 문서화/운영자 교육수행 	<ul style="list-style-type: none"> 운영자 지침서
	사용자 교육	<ul style="list-style-type: none"> 사용자지침서 작성 사용자 교육자료 작성 사용자 교육수행/사용자 교육결과 검토 	<ul style="list-style-type: none"> 사용자 지침서 사용자 교육자료
	운영환경 데이터 이행	<ul style="list-style-type: none"> 사용자 운영환경 점검 실 운영 환경으로 이관, 데이터 적재 데이터 품질 검토,전환결과 평가 	
	승인테스트	<ul style="list-style-type: none"> 승인테스트 계획수립 승인테스트 케이스 작성 사용자 교육 및 절차 수립 승인테스트 실행/승인테스트 결과 검토 및 보완 	<ul style="list-style-type: none"> 승인 테스트 계획서 승인 테스트 케이스 승인 테스트 결과서
	안정화	<ul style="list-style-type: none"> User Help Desk 구성 및 운영 개선사항 도출 및 반영 	

▪ Stage별 주요 Activity 및 산출물 (5/5)

Stage	주요 Task	주요 Activity	주요 Output
Stage MM : 메타데이터 관리	요건 분석	<ul style="list-style-type: none"> • 현행 데이터관리 프로세스 분석 • 요구사항 수집/정의 • 메타데이터 관리도구와의 Gap분석 • 구현방안 정의 • 요구사항 검토 및 보완 	<ul style="list-style-type: none"> • 요구사항 정의서
	Repository 구현	<ul style="list-style-type: none"> • 메타데이터 원천자료 식별 • 메타모델 Gap분석 • 메타모델 수정, 보완 • 메타데이터 매핑 • 메타데이터 수집 및 로드 • 메타데이터 적정성 검토 	<ul style="list-style-type: none"> • 메타데이터 매핑 설계서
	사용자 Application 구현	<ul style="list-style-type: none"> • 사용자 및 권한 정의 • 사용자 Application 설계 • 사용자 Application 구현 • 사용자 교육 및 확산 실행 	<ul style="list-style-type: none"> • 사용자 권한 정의서 • Application 설계서 • 단위테스트 결과서 • 사용자 교육자료
	메타데이터 유지관리방안 수립	<ul style="list-style-type: none"> • 관리절차 수립 • 운영자 교육 	<ul style="list-style-type: none"> • 운영자 지침서
Stage TA : 기술 아키텍처(TA)	아키텍처 정의 및 설계	<ul style="list-style-type: none"> • 아키텍처 구성요소 정의 • 아키텍처 구성요소 설계 	<ul style="list-style-type: none"> • 아키텍처 정의서 • 아키텍처 설계서
	아키텍처 구성 및 통합	<ul style="list-style-type: none"> • 시스템 설치 및 구축계획 • 시스템 구축 • 설치 확인 및 검수 	<ul style="list-style-type: none"> • 설치계획서
	아키텍처 전개	<ul style="list-style-type: none"> • 시스템 지원 및 모니터링 • 테스트 지원 • 인수인계 	<ul style="list-style-type: none"> • 테스트 케이스 및 결과서

[Data Scientist와 AI Engineer를 위한 데이터 엔지니어링]

Module2. 데이터 전처리 및 마트 구성

I. ETL(데이터전처리) 개요

1. ETL(데이터전처리) 정의 및 구성
2. ETL(데이터전처리) 아키텍처
3. ETL(데이터전처리) 활용 분야

II. ETL(데이터전처리) 도구

1. 도구의 역할 및 구성
2. 도구의 국내/국외 시장현황
3. 도구의 최근 동향
4. 도구 평가/선정 가이드

III. ETL(데이터전처리) 구축절차 및 기법

1. ETL(데이터전처리) 구축절차
2. 추출 및 전송
3. 변환
4. 정제
5. 적재
6. 데이터 검증
7. 운영 전환

IV. ETL(데이터전처리) 관련 데이터 품질관리

1. 데이터 품질
2. 데이터 품질 관리 프로세스
3. 데이터 품질 기준 작성
4. 데이터 정제 정책 수립
5. 데이터 수정 및 변환
6. 데이터 품질 관리(QA) 방안

V. 프로젝트 적용 사례

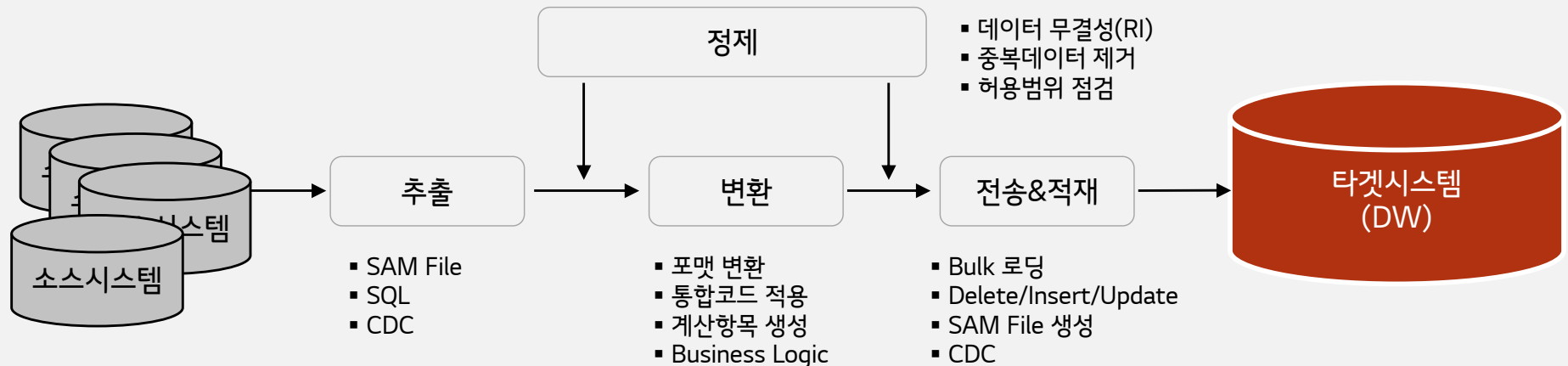
1. ETL(데이터전처리) 개발 표준 정의서
2. ETL(데이터전처리) 개발 체크리스트

ETL(데이터전처리)¹⁾은 다양한 소스시스템으로부터 필요한 데이터를 추출(Extract)하여 변환(Transformation) 작업을 거쳐 타겟 시스템(Target System)으로 전송 및 로딩>Loading)하는 모든 과정을 말합니다.

ETL(데이터전처리)의 구성

□ ETL(데이터전처리)의 구성

- Extraction(추출)
- Transformation (변환)
- Loading (적재)
- Transportation(전송) - ETT
- Cleansing(정제) - ETCL or ECTL
- Automation(자동화) - ETTA

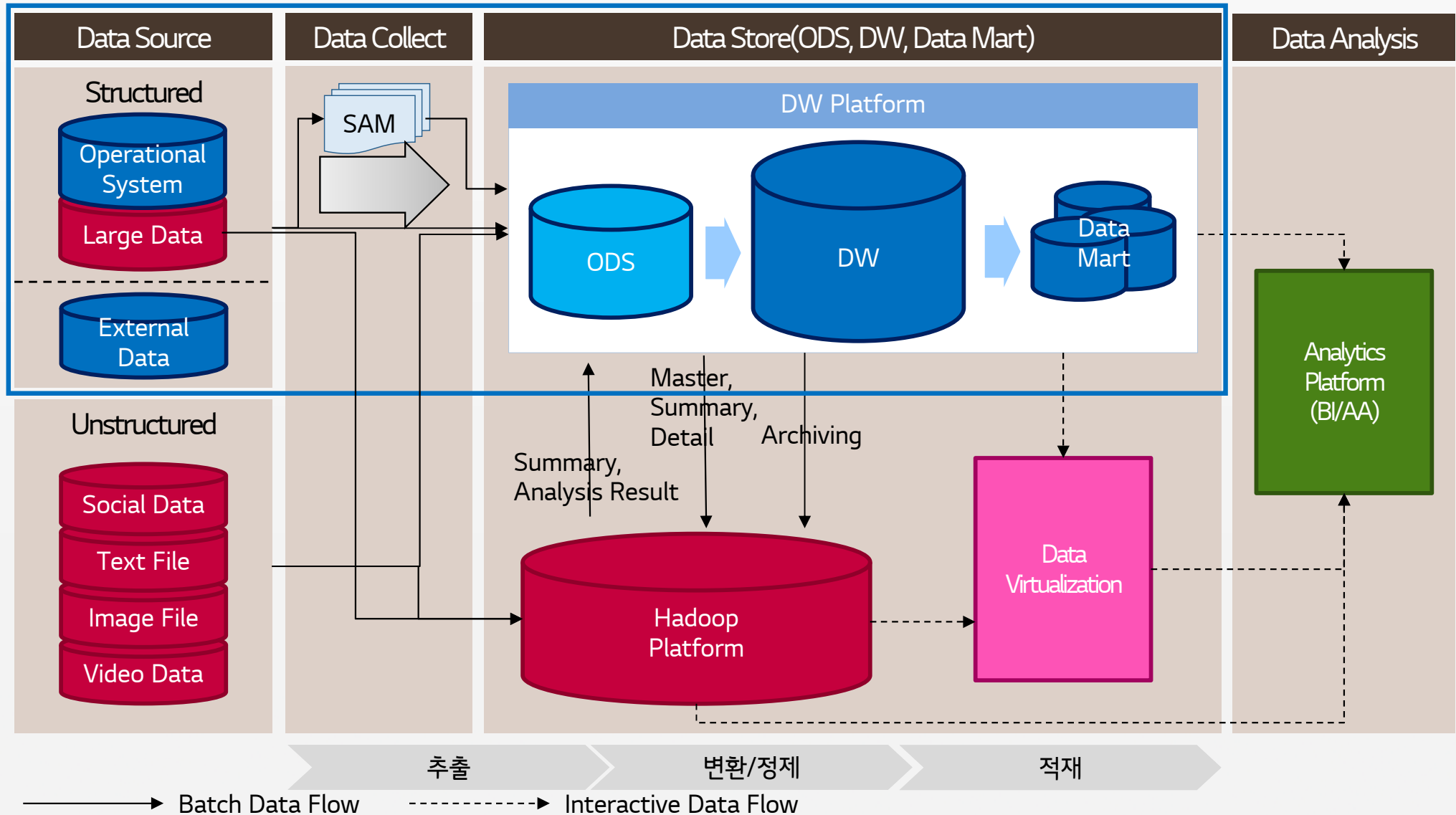


1) 현재 ETL(데이터전처리)은 Data Integration(DI) 와 동일한 의미로 사용됨

2. ETL(데이터전처리) 아키텍처

I. ETL(데이터전처리) 개요

내부(운영계) 및 외부 데이터를 Staging영역(ODS: Operational Data Store), DW DB, Data Mart에 추출/변환/적재하고 외부에 데이터를 제공하는 등 데이터 흐름 전체에 대한 아키텍처



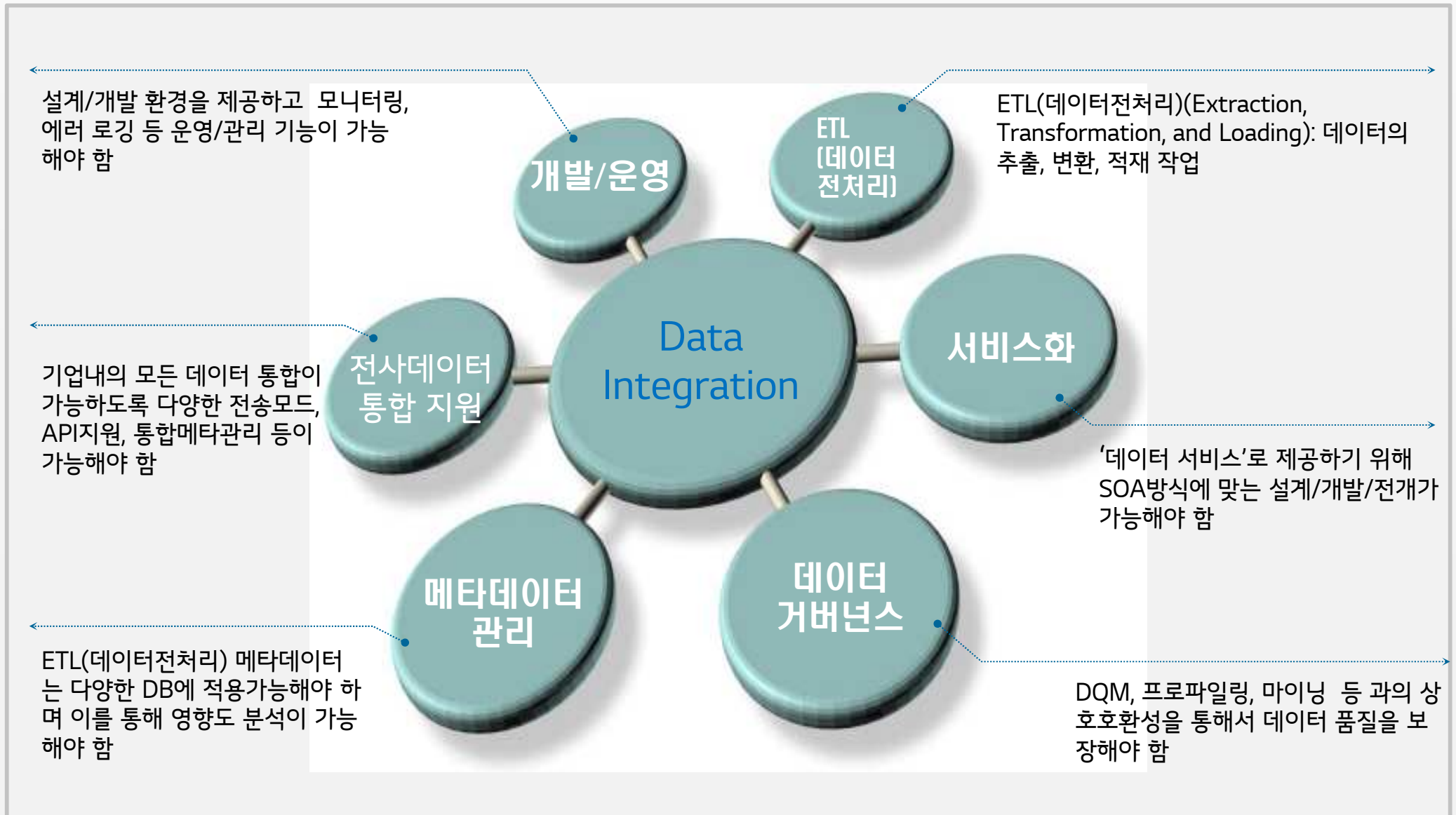
3. ETL(데이터전처리) 활용 분야

I. ETL(데이터전처리) 개요

Big Data를 제공하기 위한 Hadoop 인터페이스, In-Memory DBMS, LDW 아키텍처등과 같은 다양한 Repository를 지원하기 위해 솔루션들이 확장되고 있으며, 마스터 데이터 지원/통합, 데이터 Migration/전환, DB간의 데이터 동기화, 기업간 데이터 공유 등 다양한 데이터 통합 시나리오의 기반 기술로 활용

활용 분야	설명
비즈니스 인텔리전스(BI), 분석 및 데이터 웨어하우스를 위한 데이터 수집	Big Data를 제공하기 위한 Hadoop 인터페이스, In-Memory DBMS, LDW 아키텍처등과 같은 다양한 Repository를 지원하기 위해 솔루션들이 확장
마스터 데이터 관리 (MDM)를 지원하는 통합 및 마스터 데이터의 전달	마스터 데이터 통합, 동기화 기능 수행
데이터 Migration / 전환	기존 전환/통합 프로그램을 대체하는 기능 수행
운영계 간의 데이터 동기화	SaaS, Cloud 데이터를 포함한 기업 내/외부 데이터 일관성 보장하는 기능 수행
기업 간의 데이터 공유	고객, 공급자, 파트너 등 기업 간의 데이터 공유
SOA 환경에서의 데이터 서비스 제공	데이터 통합 뿐만 아니라 데이터를 하나의 서비스로 제공하는 영역
클라우드 기반 데이터를 포함하는 통합	기업 내부, SaaS Application, Cloud 데이터를 조합하는 데이터 통합
빅데이터 이행과제 지원	Hadoop, NoSQL DBMSs, Cloud 기반의 데이터저장소 등 빅데이터의 전달을 지원하는 영역

ETL(데이터전처리) 작업을 돕는 S/W로써 ETL(데이터전처리)도구가 기본적으로 갖추어야 하는 내용은 다음과 같습니다.



시장 현황과 국내/외 적용사례 및 Gartner Research 자료를 검토한 결과 Informatica의 PowerCenter, IBM의 DataStage, SAP의 Data Integrator가 ETL(데이터전처리) 시장을 리딩 하고 있습니다.

Figure 1. Magic Quadrant for Data Integration Tools



Source: Gartner (August 2019)

통합된 제품 군을 통해 사용자 중심적이고 비즈니스 속도에 맞춘 실시간/준 실시간 데이터 제공해야 하며 데이터와 응용 프로그램간의 통합 시너지 효과를 추구하고 있을 뿐만 아니라 빅데이터, LDW, 클라우드 지원도 가능해야 합니다.

- 비즈니스 속도에 부합하는 데이터 통합
 - 유연한 리드타임, 실시간(real-time) 또는 준 실시간(near real-time) 데이터 통합을 요구함
- 타 정보계 인프라와의 연계
 - DQM(Data quality management), MDM(Master Data Management) 등 주요 정보관리 및 거버넌스를 지원해야 함
- 데이터 및 응용 프로그램과의 통합 시너지 효과 추구
 - 데이터와 응용 프로그램, 두 기술을 통합하는 시너지 효과를 추구함
- 통합된 제품 군 제공
 - 통합된 메타데이터 지원, 다양한 데이터-응용프로그램 시너지 등 기능들 간의 통합 사용을 선호함
- 비즈니스 사용자 지향 데이터 처리
 - 사용자 직접 데이터에 접근하고 공유하고 패턴을 발견하도록 유연한 방법을 제공
- 외부 병렬처리를 통한 데이터 통합작업 부하 분배
 - Hadoop, NoSQL DB 등 BigData와 마이닝과 같은 고부하 작업에 외부 병렬처리 환경을 이용하고자 함
- 클라우드, LDW, 빅데이터 등 처리 영역의 성장가속화
 - 클라우드에서 데이터 획득, 다양한 데이터 위한 LDW, BigData연관 계획을 지원하는 부분이 증가함
- iPaaS 및 하이브리드 모델 등장
 - iPaaS로 데이터통합을 제공하거나 내/외부 환경을 결합하는 Hybrid모델 등장

기능평가 항목은 다음과 같이 정의하고 이에 공통항목인 업체 및 가격평가를 추가하여 평가 지표 별 가중치를 부여합니다.

구 분	평가 항목	평가항목 설명
데이터 추출	아키텍처 제약사항	지원 Platform의 형태 (지원 OS, 64bit 지원여부, Mainframe 환경 지원여부 등 Platform 제약사항 기술)
	소스, 타겟 접근 방법	- 다양한 형태의 소스, 타겟 대상을 추가작업(Coding, Adapter) 없이 접근 가능 - 이기종 DB에 동시접근 가능
	변경, Data 추출 방법	CDC(Changed Data Capture) 변경이 발생한 소스 데이터를 포착하여 자동으로 타겟에 반영하는 작업을 용이하게 수행
변환 적재	Many-To-One, One-To-Many, Many-To-Many 지원 여부	여러 Segment(Table)에서 하나의 Table로 또는 하나의 Segment(Table)에서 여러 Table로 분할이행
	외부 프로그램과의 연계	SQL, PLSQL, Pro*C 프로그램/C/Shell 프로그램 등을 플러그인 하여 사용
	표준 함수 및 사용자 정의 함수	제공하는 표준 함수의 종류 및 사용자 정의함수 제공여부
	데이터 적재 방법	대량 데이터 적재하기 위해 지원하는 방법의 효용성 및 다양성
	데이터 적재 방법의 유연성	Append, Refresh, Update, Insert 등 적재 방법의 종류
사용자 지원	통합 GUI 지원	일관된 GUI를 통한 시각적이고 사용자 중심의 작업수행
	오류처리 기능	자동 오류발견/ 오류처리/ 오류복구/ 재수행
	보안 및 권한 처리	사용자 인증 및 권한 설정 지원
	한글화 및 한글지원 용이성	제품의 메뉴, 매뉴얼, 도움말 등 모든 제품구성요소에 대한 한글화 지원여부

기능평가 항목은 다음과 같이 정의하고 이에 공통항목인 업체 및 가격평가를 추가하여 평가 지표 별 가중치를 부여합니다.

구 분	평가 항목	평가항목 설명
성능	병렬처리 기능 제공	추출, 변환, 적재 기능의 완전 자동화된 병렬처리를 통한 시스템 자원을 최대 활용
	Multi JOB 수행	2개 이상 JOB 동시 수행 시 지원하는 성능 보장 방안
	시스템 리소스 활용도	시스템 리소스 활용에 대한 다양한 옵션 제공 여부 (예, Job단위 병렬도 설정 가능여부 등)
메타데이터 관리	메타데이터 공유 및 타 도구와 인터페이스	다른 영역의 툴(메타데이터 툴, Sorting 툴, Cleansing 툴 등) 과의 메타데이터 통합관리(동기화가 가능해야 함)
	메타데이터 Repository의 개방성	Repository의 상용 RDBMS지원여부
	메타데이터 버전관리 기능	변경 이력을 추적할 수 있는 버전 관리 기능
	영향도 분석 기능 제공여부	소스/ 타겟 변경 시 작업의 영향력 분석(Impact Control) 기능
관리 용이성	Job 버전 및 이력 관리	변경 이력을 추적할 수 있는 버전 관리 기능
	모델 변경 시 용이성	모델링 변경 시 쉽게 적용 가능
	로그 유형 및 관리 범위	제공하는 로그 유형 및 관리 범위(예, 에러로그, Exception 로그의 상세 정도 등)
	Job 모니터링 기능	작업상황, 자원사용 및 데이터 처리 현황 및 결과에 대한 자세한 수치, 에러메시지 제공
자동화 지원	스케줄링 및 자동화 지원	Time Schedule에 의한 추출, Event에 의한 추출
	작업 소요시간 예측 기능	샘플링 기능을 통한 예측기능 제공 여부 및 제공하는 샘플링 유형
	ETL(데이터전처리) 매핑 자동화	소스와 타겟 테이블간 컬럼 자동 매핑 기능

분석 단계(IP)에 ETL(데이터전처리) 아키텍처를 설계하고 DW/DM 설계(DWD)시 ETL(데이터전처리) 개발가이드를 작성한 후 ETL(데이터전처리) 설계 및 개발을 진행합니다.

Stage	주요 Task	주요 Activity	주요 Output
Stage IP : 분석	개발자 교육	<ul style="list-style-type: none"> • 분석, 설계단계 작업내역 공유 • 솔루션 교육/ 개발 표준 교육 	
Stage DWP : 프로토타입			
Stage DWD : DW/DM 설계	DW/DM 데이터베이스 생성	<ul style="list-style-type: none"> • ODS/DW 데이터베이스 생성 • DM 데이터베이스 생성 • 인덱스 생성 	
Stage DMT : ETL(데이터전처리) 설계 및 개발	ETL(데이터전처리)프로그램 설계	<ul style="list-style-type: none"> • 추출 주기 및 방법 결정 • 매핑 흐름 정의 • 매핑 정의서 작성 	<ul style="list-style-type: none"> • 매핑 흐름도 • 프로그램 목록 • 매핑 정의서
Stage DSA : Application 설계&개발			
Stage IMP : 테스트 및 전개			
Stage MM : 메타데이터 관리	ETL(데이터전처리)프로그램 개발 및 단위테스트	<ul style="list-style-type: none"> • 단위테스트 계획수립 • 개발 및 단위테스트 • 선 후행 프로그램 식별 • 배치 프로그램개발 및 단위테스트 	<ul style="list-style-type: none"> • 단위테스트 결과서
Stage TA : 기술 아키텍처(TA)			

1. ETL(데이터전처리) 구축절차 - ETL(데이터전처리) 구축 전략 수립(예시)

성공적인 EDW 시스템을 구축하기 위하여 다음과 같은 ETL(데이터전처리) 아키텍처 설계 원칙 및 구현 전략을 수립합니다.

설계원칙

- ✓ 원천 시스템의 부하를 최소화한 설계
- ✓ Performance 향상 및 작업 시간 단축을 위한 배치 구성 최적화
- ✓ 향후 유지보수를 고려한 표준화되고 간결한 구조
- ✓ 안정적이고 용이한 운영을 고려한 설계

구현전략

자원 극대화

- ✓ 원천 시스템 자원 활용 최소화
- ✓ ETL(데이터전처리)도구를 활용한 원천 데이터 추출
- ✓ STG file 및 DB의 활용

효율

- ✓ 파일 처리 중심의 ETL(데이터전처리) 프로세스
- ✓ DW DB 특성을 고려한 아키텍처
- ✓ 배치 프로그램 구성의 최적화

표준화

- ✓ ETL(데이터전처리) 프로세스의 표준화, 정규화
- ✓ 메타시스템 연계를 통한 영향도 분석
- ✓ ETL(데이터전처리) 도구 및 프로그램의 형상관리

안정 및 통합

- ✓ 공통 로그 및 UI를 통한 통합 모니터링
- ✓ 검증을 통한 데이터 정합성 보장
- ✓ 오류 데이터, 검증 집계 조회

ETL(데이터전처리) 작업에서 가장 중요한 단계로 소스 데이터와 ODS 간, ODS와 DW의 타겟 테이블 간의 변환 로직, 추출 주기 등 매핑 정보를 정의합니다.

소스 시스템 분석

타겟 시스템 분석

데이터 매핑 정의

Task

• 소스 시스템 분석

• 소스 시스템 테이블 분석

• 대상 테이블 컬럼 분석

• Reverse Engineering을 통한 엔티티간 관계 분석

Task

• 물리 모델링을 통한 DB구현

• 타겟 테이블/컬럼 정의

Task

• 소스/타겟 매핑 정의

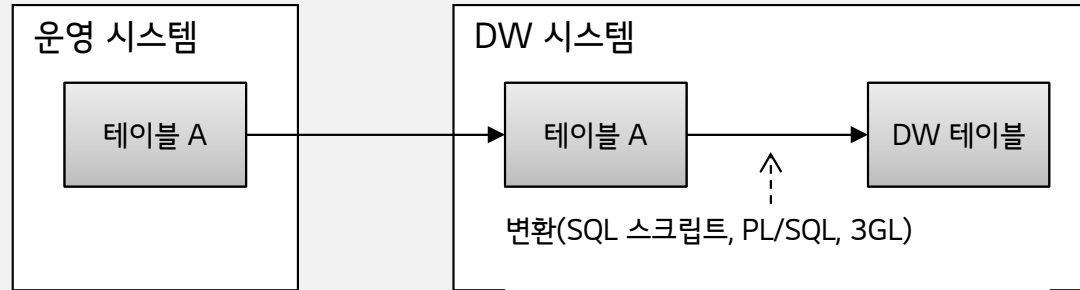
• 데이터 추출 주기 설계

• 소스/타겟 변환 로직 설계

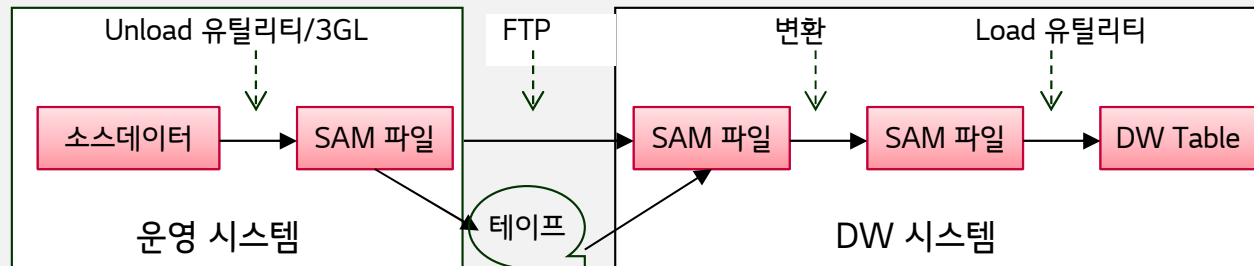
• 데이터 매핑 정의서 작성

추출 및 전송 방식에 따라 Online 방식과 Offline 방식으로 나눌 수 있습니다.

• 온라인



• 오프라인



	온라인(Stream)	오프라인(File)
방식	• ETL(데이터전처리) 도구에서 ODBC/Native API로 추출하거나, 게이트웨이/분산시스템기능/Message Queue 이용	• 소스 시스템에서 파일로 추출하여 FTP/공유디스크 등을 통해 옮겨옴
장점	• 개발이 간편함 • 전 ETL(데이터전처리) 프로세스가 중간파일 없이 끊김 없이 이어질 수 있음 (on-the-fly)	• ETL(데이터전처리) 작업의 재시작이 쉬움 • 전송 전에 암호화 및 압축이 용이
사용시점	• 소량 처리시 적절. 주로 운영 시	• 대량 처리시 적절. 주로 초기 적재 시 • 운영계 시스템 부하를 최소화할 경우

운영 단계에서 데이터 추출 방식을 다음과 같습니다.

❑ Refresh

- ✓가장 간단한 방법. 정기적으로 전체 데이터를 추출
- ✓주로 Dimension 테이블을 대상으로 함

❑ Timestamp

- ✓변경일자 등의 필드를 기준으로 최근 추출 시점 이후의 레코드들을 추출
- ✓원천시스템에 Timestamp용 시계열 컬럼이 필요함

❑ Snapshot 비교

- ✓덤프 파일의 전/후 이미지를 비교. Key를 기준으로 레코드의 모든 필드를 비교. 매우 복잡하며 이런 경우 대개 refresh 방식으로 감

❑ DBMS log file

- ✓추출용 파일로 변환
- ✓로그를 모니터링 하여 DB에 commit된 변경사항만 수집하는 애플리케이션 필요
- ✓한 DBMS 내에서도 버전에 따라 로그 파일 형식이 틀려 추출이 어렵고, 변경을 포착할 때까지 로그가 비워지지 않고 계속 유지되어야 함
- ✓주로 외부 솔루션을 사용함

❑ Application log

- ✓DW에 필요한 변경된 정보만 따로 큐에 기록하도록 운영계 어플리케이션을 수정
- ✓운영계 DB에 Trigger를 거는 방법도 있음
- ✓운영계 시스템에 부하가 크고 복잡

데이터 유형, 추출 주기, 데이터 사이즈(추출 성능) 등 에 따라 추출 기준을 정의합니다.

정의 기준		내 용	비 고
원천 시스템 데이터 유형		IMS DB, DB2, RDB(DB2 UDB, Oracle)	
데이터 요청 주기		배치 (일/월), 실시간	
추출 성능 및 부하 (데이터 사이즈)		추출 시간 기준 (예: 20분 이상/이하) 데이터 사이즈 기준 (예: 10Mb 이상/이하)	Prototype을 통한 검증 필요
기타	추출 구현을 위한 Effort	추출 구현을 위해 일정 수준 이상의 Effort가 소요?	
	기존의 user log 활용 가능 여부	기존의 user log를 추출 원천으로 활용할 수 있는지	
	변경 데이터 비율	전체 데이터 중 변경 데이터 비율 (예: 30% 이상/이하)	
	원천 데이터의 시계열 컬럼 포함 여부	원천 데이터에 변경 분 추출을 위한 시계열 컬럼이 있는지	RDB의 경우 시계열 컬럼 추가 가능한지 여부 확인

코드체계 통합, 데이터 타입 변환, Key체계 통합 등 을 수행합니다.

□ 종류

- 통합 : Key 및 코드 체계 통합
- 코드의 설명 부여
- 데이터 타입/포맷 변환 : date/numeric
- EBCDIC/ASCII 및 한글 변환
- 데이터 구조 변경 : 정규화/비정규화
- Biz. Rule 적용 : 계산 및 도출
- Sort/Merge/Sum
- 컬럼 선택/Filtering/Split/Comparison
- NULL 값 처리
- 기타 : Time 디멘전 생성, sequence 생성

코드체계 통합

m/f
1/0
x/y

→ M/F

데이터타입 변환

mm/dd/yy
yy/mm/dd
dd/mm/yy

→ yyyy/mm/dd

Key체계 통합

주민번호
고객번호
여권번호

→ 고객번호

파일 변환, DB내의 직접 변환 그리고 코드변환 기준을 정의합니다.

변환 유형	설 명	정의 기준	제약 사항/비 고
파일 변환	<ul style="list-style-type: none"> 변환 규칙에 따라 임시 파일 및 데이터베이스에 load하기 위한 적재 파일을 생성하는 방식 ETL(데이터전처리) Tool을 이용 	<ul style="list-style-type: none"> 기본 ETL(데이터전처리) 방안 Host 추출 파일의 ASCII 변환 변환 규칙에 따른 임시 파일 생성 적재 파일 생성 	<ul style="list-style-type: none"> 로직이 포함된 복잡한 변환 규칙 적용에 어려움
DB 변환	<ul style="list-style-type: none"> DB에서 직접 변환을 수행한 후 DB에 적재하는 방식 Procedure, ETL(데이터전처리) Tool DB to DB, UserSQL 모듈을 이용하는 방식 	<ul style="list-style-type: none"> 복잡한 변환 규칙의 작업 데이터베이스의 resource 활용 	<ul style="list-style-type: none"> 통합 모니터링 방안 고려
코드 변환	<ul style="list-style-type: none"> 인스턴스 코드 변환 및 당사자/상품 코드 변환이 해당 코드변환 공통모듈을 사용하여 코드 변환 수행 ETL(데이터전처리) Tool에서 코드변환 공통 모듈을 호출하는 형태 		<ul style="list-style-type: none"> ETL(데이터전처리) Tool을 활용한 파일 변환 또는 DB to DB 방식만 적용

□ 소스 데이터에서 흔히 나타나는 문제점

- 코드체계의 혼재 : 업무 시스템 별로 상이한 코드, 신/구 코드의 공존
- 잘못된 코드
- 중복된 용도를 갖는 컬럼
- free-form 필드에 묻혀있는 정보
- 고의로 부정확한 값 입력/오타
- 필드 추가로 인한 NULL 값
- 갱신이 안된 구 데이터
- 빠지거나, 틀리거나, 중복되는 text : 주소와 이름. LGCNS , LG-CNS

□ 정제 작업의 구성

- 깨끗한 데이터를 입력하도록 운영 App. 수정 + ETL(데이터전처리) 과정 중 소스 데이터를 정제

□ 정제 작업의 종류

- 데이터 무결성(Referential Integrity) 점검
- 중복 건의 제거
- 허용범위 점검 : 예) 잘못된 날짜의 교정
- 이름, 주소의 중복 제거, 교정 및 표준화

ETL(데이터전처리) 정제 대상 및 유형, 정제 방안은 다음과 같습니다.

유형	내 용	정제 및 변환 방안
Missing value (Null value)	<ul style="list-style-type: none"> Target이 not null 컬럼인데 매핑되는 데이터 값이 없는 경우 	<ul style="list-style-type: none"> Missing value에 대한 DBMS 수준의 default value 정의 ETL(데이터전처리) 단계에서 공통모듈을 이용한 default value 정의
Data type, Format 오류	<ul style="list-style-type: none"> 매핑 되는 데이터의 날짜, 시간 유형이 잘못된 경우 Target이 number type 컬럼인데 매핑 되는 데이터 값이 char type인 경우 매핑 되는 데이터 값의 길이가 target 컬럼 보다 큰 경우 	<ul style="list-style-type: none"> 오류 날짜, 시간 유형의 default value 정의 Space, 특수문자 등을 trim, replace하고 number type으로 전환 공통모듈을 이용하여 정제하고 오류 데이터를 파일로 작성
비정형 데이터 오류	<ul style="list-style-type: none"> 원천 데이터의 한글이 깨짐 전화번호, 주소 유형이 잘못된 경우 	<ul style="list-style-type: none"> 컨버전 모듈을 이용한 데이터 정제 수행 전화번호, 주소 등의 비정형 데이터의 경우 매핑 정의서에 정제 요구사항이 있는 경우 정제 수행
코드 오류, RI(참조무결성) 오류	<ul style="list-style-type: none"> 인스턴스코드 매핑 시 source 코드 값이 매핑 되는 target 코드 값이 존재하지 않는 경우 Child 컬럼에 매핑 되는 데이터 값이 Parent 테이블 key 값에 존재하지 않는 경우 	<ul style="list-style-type: none"> 인스턴스코드 매핑 공통 모듈에서 매핑 되지 않는 코드 값들을 매핑 오류 파일로 작성하고, 오류 데이터 처리 프로세스에 따라 처리 데이터 적재 시 RI 오류로 reject되는 파일을 작성하고, 오류 데이터 처리 프로세스에 따라 처리
중복 데이터	<ul style="list-style-type: none"> 원천 데이터에 PK 중복이 존재하거나, 2개 이상의 원천에서 target으로 merge되는 데이터에 중복이 존재하는 경우 	<ul style="list-style-type: none"> PK 중복에 따라 적재가 reject되는 데이터를 오류 파일로 작성하고, 오류데이터 처리 프로세스에 따라 처리 Merge할 경우 중복되는 데이터의 경우 어떤 데이터를 선택할지 전환 규칙을 정의

❑ 적재 방법과 속도

- Bulk loading > Bulk Insert > Insert

❑ Bulk Loader 이용 시 주의사항

- No logging
- Pre-sort
- Delete보다는 Truncate
- No Index & FK

❑ 운영 단계에서의 업데이트 방법

- REFRESH : Truncate & Insert
- INCREMENTAL UPDATE
 - ✓ Only Insert
 - ✓ Update & Insert or Delete & Insert
 - ✓ 그 외 특수한 변경 관리 로직

데이터 처리량, 데이터 특성, 구현 편의성에 따라 적재방안을 결정합니다.

유형	설 명	정의 기준	제약 사항/비 고
Append	<ul style="list-style-type: none"> 적재 파일을 기존의 테이블에 추가하는 형태로 적재 기 적재된 타겟 테이블과 중복이 배제되어 작성된 경우 적용 	<ul style="list-style-type: none"> 기본 적재 방안 	
Delete +Append	<ul style="list-style-type: none"> ETL(데이터전처리) 일자를 조건으로 데이터를 삭제하고 append - delete from .. where .. 수행 후 데이터 append 적재 적재 대상 테이블의 key를 사용하여 데이터를 삭제하고 append 	<ul style="list-style-type: none"> 재작업일 경우 기본 적재 유형 영역/통합DM의 기본 적재 유형 	
Truncate	<ul style="list-style-type: none"> 기존의 테이블 데이터 전체를 replace하는 형태로 적재 	<ul style="list-style-type: none"> 시계열 컬럼이 존재하지 않은 원장성 테이블 과거의 이력 데이터를 보존할 필요가 없는 경우 STG 적재의 기본 유형 	변환 규칙에 따라 원장성 데이터에 대하여 제한적으로 적용
Insert	<ul style="list-style-type: none"> Local DB 내에서 변환하여 직접 적재하는 경우 사용 - insert .. select .. 형태로 사용 Procedure, ETL(데이터전처리) Tool UserSQL 에서만 사용 		Fetch 방식 insert는 사용을 지양
Create	<ul style="list-style-type: none"> Local DB 내에서 변환하여 임시 테이블을 생성하면서 적재하는 방안 - create .. select .. 형태로 사용 Procedure, ETL(데이터전처리) Tool UserSQL 에서만 사용 	<ul style="list-style-type: none"> 임시 데이터 작성 	Index 등의 사용에 유의

□ 대용량 데이터 처리시 다음 사항을 고려

- 병렬 처리
- Process Tuning
- SQL문 Tuning
- ETL(데이터전처리) Tool Logic Tuning
- ODS DB의 memory/temp db size증가
- 코드 테이블 참조 방법의 개선
- Filing & Bulk Loading
- 적재 시 Index 제거 및 Foreign Key 해제
- 적절한 Index
- Sort Utility로 Sort/Merge/Sum

데이터를 신뢰하고 이를 활용하게 하기 위해서는 고 품질의 데이터를 제공해야 하므로 데이터 검증 방안을 체계적으로 수립해야 합니다.

검증 구분	검증 내용	결과 활용
1. 데이터 프로파일	공백, NULL 건수	- 매핑 및 개발 오류 검토 (신규/변경 컬럼)
	최소값, 최대값	- 데이터 분포 검토 (금액단위 등 검토)
2. 기초 검증	소스 및 타겟 건수 검증	
	코드, 여부 검증	- 코드,여부성 컬럼 허용값 검증 - 코드 오류 정재 및 코드 조사 대상 파악
	일자, 년월 검증	- ~일자, ~년월 컬럼 검증 - 오류 데이터 현황, 정재대상 파악 - 기본값 설정
	RI 검증	- 주요 RI 오류 현황 파악 - Parent Table에 없는 PK가 Child Table에 존재하는 경우 확인 - 별도 코드 테이블 관리 코드 허용 값 검증
3. Business 로직 검증	컬럼 또는 테이블단위 로직 검증	- 업무로직 위배 데이터 파악 - 클린징 및 매핑 반영 대상 파악
4. 소스 비교 검증	소스 데이터와의 1:1 검증	- 소스 데이터 Staging 적재 방안 별도 검토

Data 분포도 분석 및 검증 Test는 모든 Fact Table과 주요 Big Dimension Table에 대하여 각 Table의 디멘전 키 컬럼에 대해 Valid Data와 그들 각각에 대한 건수/구성비율을 파악합니다.

□ 총실도 검사

- 주요 디멘전 키 컬럼의 valid data, count, percent 조사

TABLE Data 총실도 Check List					
담당자	홍태헌	작업일자	00-12-05		
Table명	Column명	Data 총실도			check 내용 및 비정상 data 내역
		Distinct Value	Count	Percent	
D_계층조직	취급자ID	정상값	28944	100.0%	
	28944	사용인ID	정상값	3512	12.1%
		취급자대구분ID	1	6343	21.9% null 30
			2	9520	32.9% ASCII null 24
			3	10944	37.8%
		정상총실도	26807	92.6%	
		space	541	1.9%	
		z	2100	7.3%	
	취급자대구분DESC	식급	6343	21.9%	
		대리점	9520	32.9%	
		설 계사	10944	37.8%	
				0.0%	
				0.0%	

총실도 조사에서 불거진 문제에 대한 Test Case를 만들어, 각각의 오류유형(단순Move, 코드변환, 계산Logic/Sum)을 파악합니다.

□ Test Case 조사

- 총실도 조사 과정에서 오류가 있다고 판단되는 부분에 대해 Test Case를 설정하여 처리 후 조치결과를 기록

(TEST 주체) 테스트 계획 및 결과서

1. TEST 주체대상 : 개발자(ETT),고객,기타

2.오류유형 : 단순move - "M", 코드변환 - 'C', 계산Logic/SUM - "L", 기타 - "Z"

3.오류발생처 : Host - "H", DW - "D", Modeling - "M", OLAP - "O", 기타 - "Z"

순번	테스트 케이스	오류			test일 자	처리 완료 일	고객 담당 자	개발 담당 자	비고
		유형	내용	발생처					
	사용인명		null 25432건 => space처리						
	반id		반id => 팀id로 명칭 수정						
	반명		반명 => 팀명으로 명칭 수정 null => space						
	점포id		space 314건 => 해당 데이터 확인 요망						
	점포명		코드값이 들어있는것 13건 => space처리						
	지점id		space 327건 => 해당 데이터 확인 요망						
	본부id		space 327건 => 해당 데이터 확인 요망						
	사업부id		space 327건 => 해당 데이터 확인 요망						
	취급자소구분desc		전문대졸 => '전문' 으로 수정						

□ 계수 비교(Cross-Check)

- 기간계 화면의 계수와 DW에서 Query하여 나오는 값을 비교 – SUM 혹은 Sample record 비교

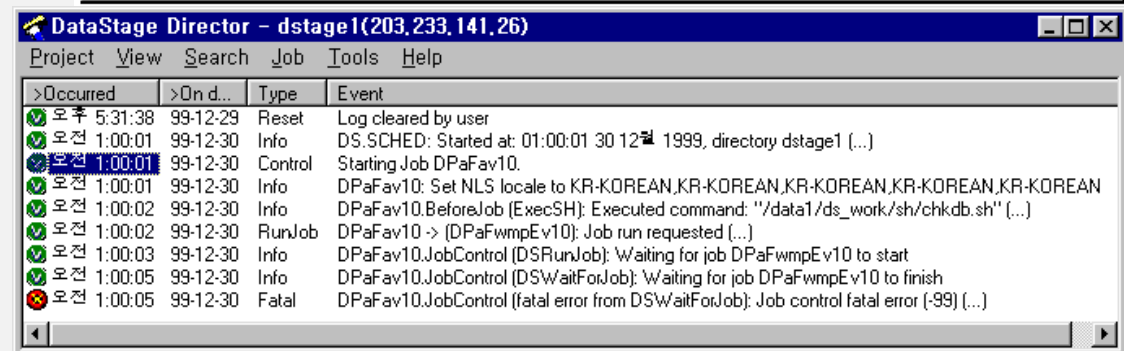
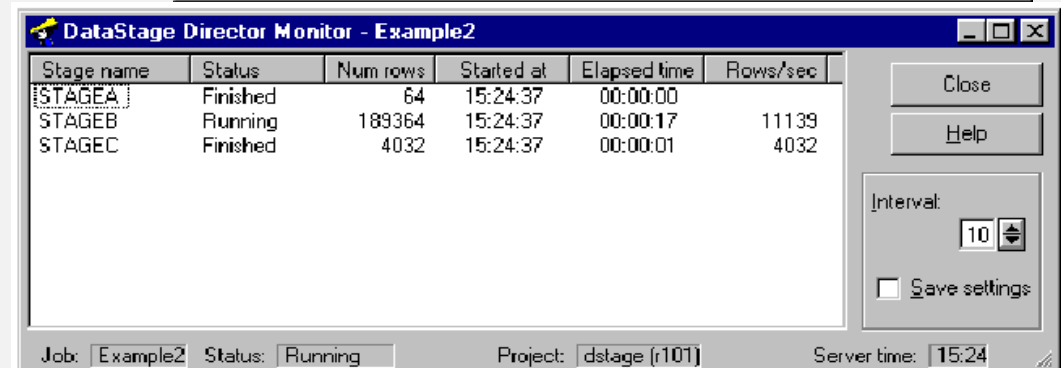
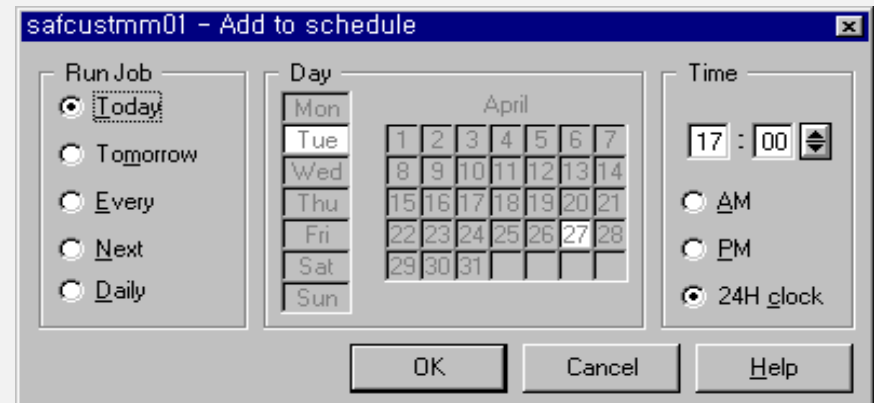
항 목	내 용
1. 검증 항목/요건 정의	<ol style="list-style-type: none"> 1) 검증항목 정의서 작성 <ul style="list-style-type: none"> - 여러 검증 후보 항목이 있을 수 있으며 이중 해당 ETL(데이터전처리) 작업 수행 후 가능하고 필요한 검증 항목만을 선택적으로 검증한다. 2) Source 데이터 유효성검증 시나리오 작성 <ul style="list-style-type: none"> - 테이블 내 특정항목에 대하여 업무적으로 정의되어지는 마이그레이션 요건 3) 참조 무결성 검증 시나리오 작성 <ul style="list-style-type: none"> - 테이블과 테이블간 업무적으로 정의되어지는 마이그레이션 요건 4) 코드 검증 시나리오 작성 <ul style="list-style-type: none"> - 정상코드대상정의 - 오류에 대하여 코드변환대상 정의 5) ETL(데이터전처리) 단계별 검증 대상 <ul style="list-style-type: none"> - 원천->수집, EDW내 ETL(데이터전처리) 처리, 통합->마트 등 단계별로 검증
2. 검증 프로그램 작성	<ol style="list-style-type: none"> 1) SQL을 운용자동화 메타데이터로 DB에 보관/관리 2) 수행시점에 호출변수로 SQL_ID를 받아 메타DB로부터 주어진 SQL_ID에 해당하는 SQL을 참조하여 Dynamic SQL을 통해 검증을 수행하는 공통 수행 프로그램 개발 3) 공통 실행 C 프로그램에서 메타DB에 저장된 SQL을 참조하여 해당 검증을 수행하고, 검증결과를 별도의 검증 테이블에 저장하는 방식으로 처리 4) 본 자동화/공통 모듈화된 검증 프로그램은 모든 검증에 반영될 수 없으며 원천->수집, 통합->마트 등의 단계를 제외한 EDW의 DB2 내에서 처리되는 부분에 대해 처리 가능함. 5) 원천->수집, 통합->마트 단계의 검증 작업도 공통 프로그램으로 구현할 수 있는 경우 부분적으로라도 최대한 적용한다.

항 목	내 용
3. 검증 수행	<ol style="list-style-type: none"> 1) Source 데이터 유효성 검증 2) 테이블 별 적재시간, 적재건수 집계 3) 오류 개수 집계 4) 오류 유형 파악/분류 5) 코드 별 건수,금액 집계 6) 코드검증 7) 참조 무결성 검증 8) Null 데이터 검증 9) 데이터 Eye Check 등
4. 모니터링 및 검증 오류 조치	<ol style="list-style-type: none"> 1) 검증보고서 작성 2) 오류 데이터 처리방안 정의(Cleansing 방안) 및 Cleansing 요청 3) 데이터 변환 로직 정의 4) 데이터 코드정의 5) 데이터 코드오류 변환정의 6) 검증 결과 기록 및 표시 <ul style="list-style-type: none"> - 각 ETL(데이터전처리) 검증 작업은 검증 결과를 별도의 단일한 검증 테이블에 기록한다. - ETL(데이터전처리) 검증 테이블의 구조는 각 개발팀과 협의하여 추후 확정한다. - 검증 테이블에 검증결과를 기록하는 공통 C API를 개발한다. - 공통 C API의 목적은 검증 테이블에 대한 입력 역할을 수행한다. - 기타 언어 또는 툴로 구현된 경우도 단일 검증테이블에 동일한 포맷으로 기록한다. - 검증테이블을 Web 화면을 통해 검증결과를 조회할 수 있는 화면을 구축한다. - Web 화면은 ReportNet 등 보고서 생성 툴 또는 OLAP등을 활용하여 개발한다.

□ ETL(데이터전처리) 프로세스 관리

- 메타데이터에 기반한 단일 작업 제어환경 구성
- ETL(데이터전처리) Tool 또는 외부 스케줄링 프로그램 이용

- Job 구성(일련의 step을 묶음)
- Job 스케줄링(진행상황 체크)
 - 현재 수행 STEP, 시작시간, 소요시간...
- 모니터링
- 로그 기록(실행결과수집)
- 예외 처리(REJECT 건)
- 장애 처리(복구 및 재시작)
- 통보(콘솔, E-MAIL, SMS 등)



□ 배치 프로세스 구성도의 작성

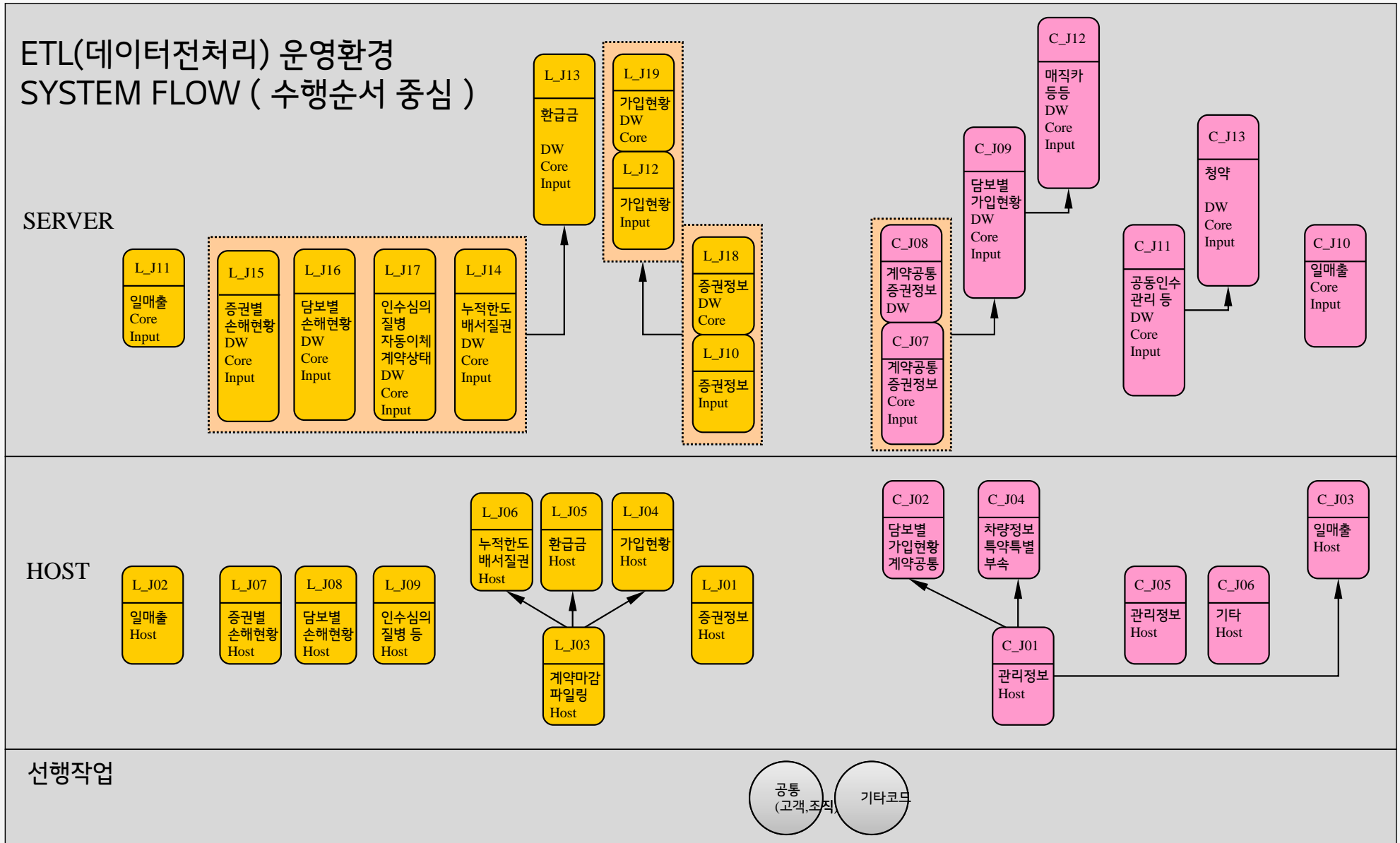
- ETL(데이터전처리) 프로세스에 포함되는 모든 Job들을 적절히 배합하여 배치 프로세스를 구성
- 고려사항
 - ✓ 소스 시스템 관련 선행작업
 - ✓ Job 간의 연관성
 - ✓ Job Grouping 방법
 - ✓ 작업주기 및 작업가능 시간대
 - ✓ 동시 수행 가능 작업 수
 - ✓ 가용 디스크 공간
 - ✓ Job Control 방법

□ 운영용 ETL(데이터전처리) 프로그램 구현

- 부분 반영 가능하도록 구현

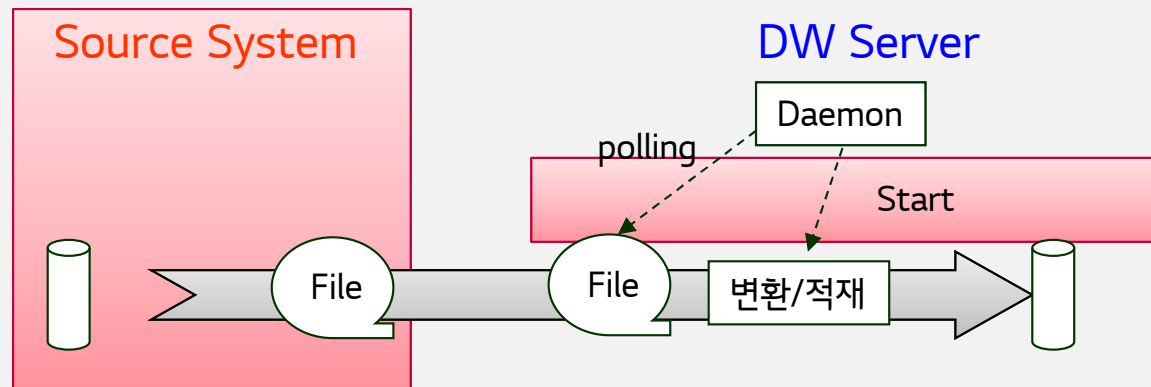
□ 배치 프로그램 구현

- 배치 프로세스 구성도를 기반으로 Job 순서 지정 및 스케줄링



- 목 표 : 담당자가 작업이 끝날 때까지 지켜보며 제어하지 않아도 되도록 함
- Issue : 이기종 플랫폼/어플리케이션 간의 연계
- 연계 방법
 - 파일의 존재 체크 : 파일이 추출되어 도착했는지를 모니터(Daemon)가 주기적으로 체크하여 후속 단계를 시작시킴
 - Flag 설정 : 추출 프로세스가 메타데이터 카탈로그에 완료표시를 해두면 모니터가 주기적으로 테이블을 조회하여 후속 단계를 시작시킴
 - ETL(데이터전처리) 도구가 제공하는 기능 이용
 - 수행시간 설정 : 선행 작업이 언제쯤 끝나리라 예측하여 다음 작업을 특정 시간에 예약. 권장사항 아님

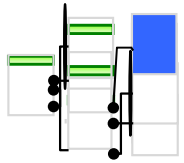
□ 구현 사례



□ 데이터 품질 관리란 ?

- 조직의 전략과 목적을 달성하기 위해 필요한 데이터를 요구하는 조직구성원 또는 이해관계자의 만족도를 충족시킬 수 있는 수준으로 유지하고 개선하기 위해 수행하는 모든 활동

Definition(정의)

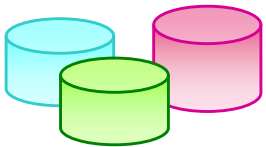


- 데이터 사양 및 메타데이터에 대한 품질
- 데이터 값이 업무를 수행할 수 있도록 정의되어 있는가?

표준화/모델

준수도, 충실도 관점

Contents(값)



- 데이터 값의 정확성에 대한 품질은?
- 데이터 값이 정확하게 정의된 업무 규칙을 준수하는가?

데이터

완전성, 유효성, 일관성
관점

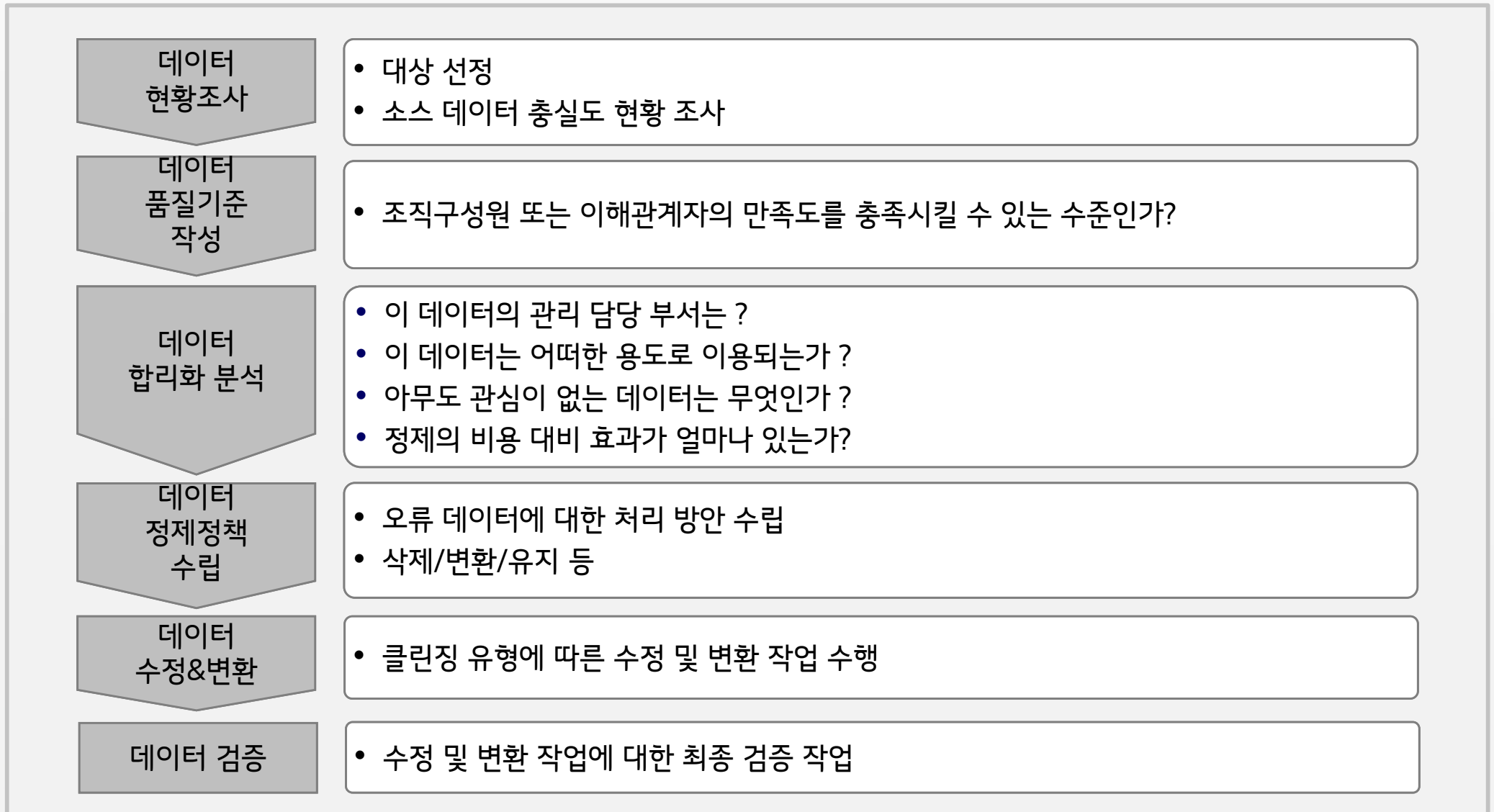
Presentation(활용)



- 지식 작업자에게 전달되는 정보 제품으로서의 품질은?
- 필요할 때 즉시 제대로 된 정보를 얻을 수 있는가?

정보

적시성, 편리성, 활용성
관점



구 분	품질 기준	설 명
유효성	정확성	<ul style="list-style-type: none"> • 실세계에 존재하는 객체(사건, 사물, 개념 등)의 값이 오류없이 저장되어 있음 • 소스 시스템과 일치. 다른 경우엔 그 이유를 문서화
	일관성	<ul style="list-style-type: none"> • 정보시스템 내의 동일한 데이터 간의 불일치가 발생하지 않음 • 내용이 같으면 형식도 같아야 함 - 중복이 없어야 함
활용성	유용성	<ul style="list-style-type: none"> • 조직이 요구하는 데이터의 범위와 상세화 정도를 충족시킬 수 있음
	접근성	<ul style="list-style-type: none"> • 사용자가 원하는 데이터를 손쉽게 이용할 수 있음
	적시성	<ul style="list-style-type: none"> • 응답시간과 같은 비기능적 요구사항 그리고 데이터의 최신성 유지와 같은 품질 요건을 얼마나 대처하고 있는지의 의미 • 데이터의 업데이트 주기는 적시/공개/준수되어야 함
	보안성	<ul style="list-style-type: none"> • 외부 및 내부 요인으로부터 데이터를 적절히 보호하고 있는지 여부를 의미

□ 정제 정책 : DRAK

* D : 삭제>Delete)

* R : Biz. Rule 적용해 교체(Replace)

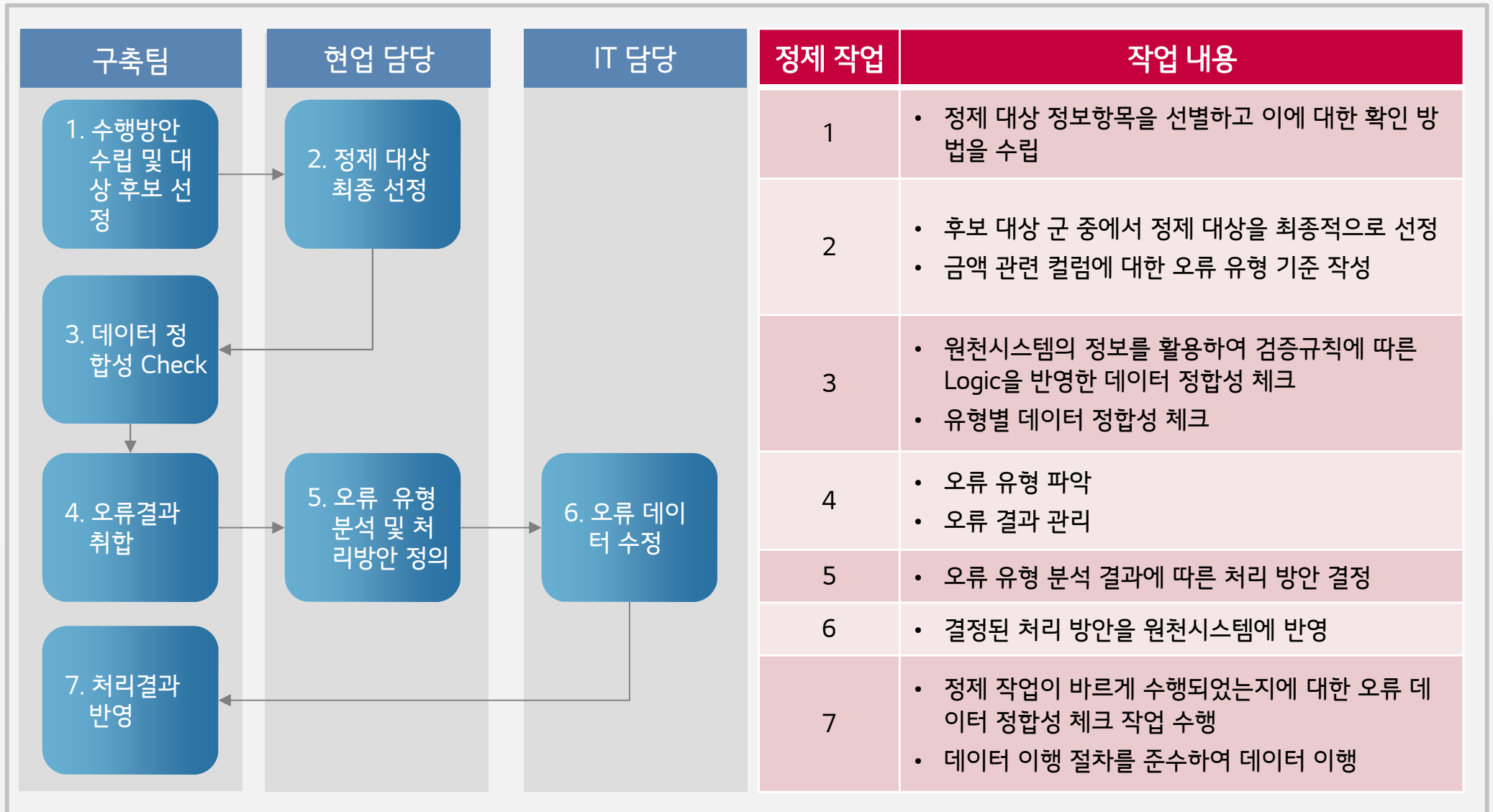
* A : 디폴트 값 지정(Assign) * K : 그대로 놔둠(Keep)

구분	허용범위 이탈	빠짐	더미값
Primay Key(고객 ID..)	D	D	D
코드성 컬럼 : 반드시 준수될 것	D	D	R/A
코드성 컬럼 : 중요하지 않은 것	R/A	K	R/A
PK 일부가 아닌 일자	R/A	R/A	R/A
PK 일부인 일자	R/A	R/A	R/A
추가적인 속성	R/A	K	R/A

- 운영계 시스템에서 오류를 수정할 수 있는가?
- 어떤 방식으로 오류 데이터를 정제할 것인가?
- 정제는 누가 담당할 것인가?

- ❑ 다른 소스가 있다면 가장 높은 품질의 소스를 선택
- ❑ 운영계 시스템 팀과 협업하면서 데이터 품질 문제점 및 대안을 도출하고 경영진에 공동으로 제기 - 품질은 BIZ. 문제 !
- ❑ 가능한 한 운영계 시스템에서 문제를 수정할 것. 당장 불가능하면 다음 번 운영계 시스템 개정 시 반영하도록 할 것
- ❑ 운영계 시스템에서 해결할 수 없는 것은 ETL(데이터전처리) 과정에서 해결
- ❑ 모든 문제를 해결하려 들지 말고 일부는 사용자에게 공개하여 데이터 품질 개선에 대한 공감대를 형성
- ❑ 운영계 시스템 팀과 협업하여 정기적인 검사 및 정제를 하도록 도움
- ❑ 정치력이 뒷받침된다면, 운영계 시스템 팀에서 추출한 데이터의 품질을 책임지도록 함

유형	클리닝 측정 항목	설명	클리닝 방안
완전성	Null Value Check	Null 값을 가지는 Attribute별 측정치	<ul style="list-style-type: none"> 메타데이터 관리시스템을 통한 Rule Set 정의 및 관리 ETL(데이터전처리) 및 데이터 이행 단계에서 Rule Set에 의해 정제
	Space	Space 값을 가지는 Attribute별 측정치	
	Default Value	Default 값을 가지는 Attribute별 측정치	
유효성	Format	Data Format의 유효성 체크	
	Range	Range를 벗어나는 Data Check	
	정규화	정규화 위배 Check	
무결성	PK 정의 (유일성 등)	PK 유일성 Check	
	Foreign Key와 참조 무결성	Foreign Key 참조 무결성 Check	
	Relation Rule	1:1, 1:N 등의 Cardinality Check	
정합성	Value간의 집합개념으로서의 일치성	두가지 이상의 조합된 Value Check	
	산술규칙에 의한 Value	계산식에 위배된 Value Check	
	Time 종속관계	시간과 관련된 Data Check	
	조건 종속 관계(Business Rule)	Business Rule 위배 Check	



❑ Cross-Checking

- 소스에 여러 레벨의 질의를 발행하여 그 결과를 비교해 봄. 프로세스를 자동화하고 결과를 메타데이터로 기록
- 수작업 검사 : 여러 소스에서 통합되어 비교가 곤란하다면 허용 가능한 에러 범위(예 : $\pm 5\%$)를 벗어나는지 검사
- 데이터 추출에는 운영 시스템에 반영되어 있지 않은 business logic이 들어있을 수 있고, 데이터의 시점의 차이로 인해 운영계 화면의 값과 일치하지 않을 수 있지만, 그 이유는 설명할 수 있어야 함

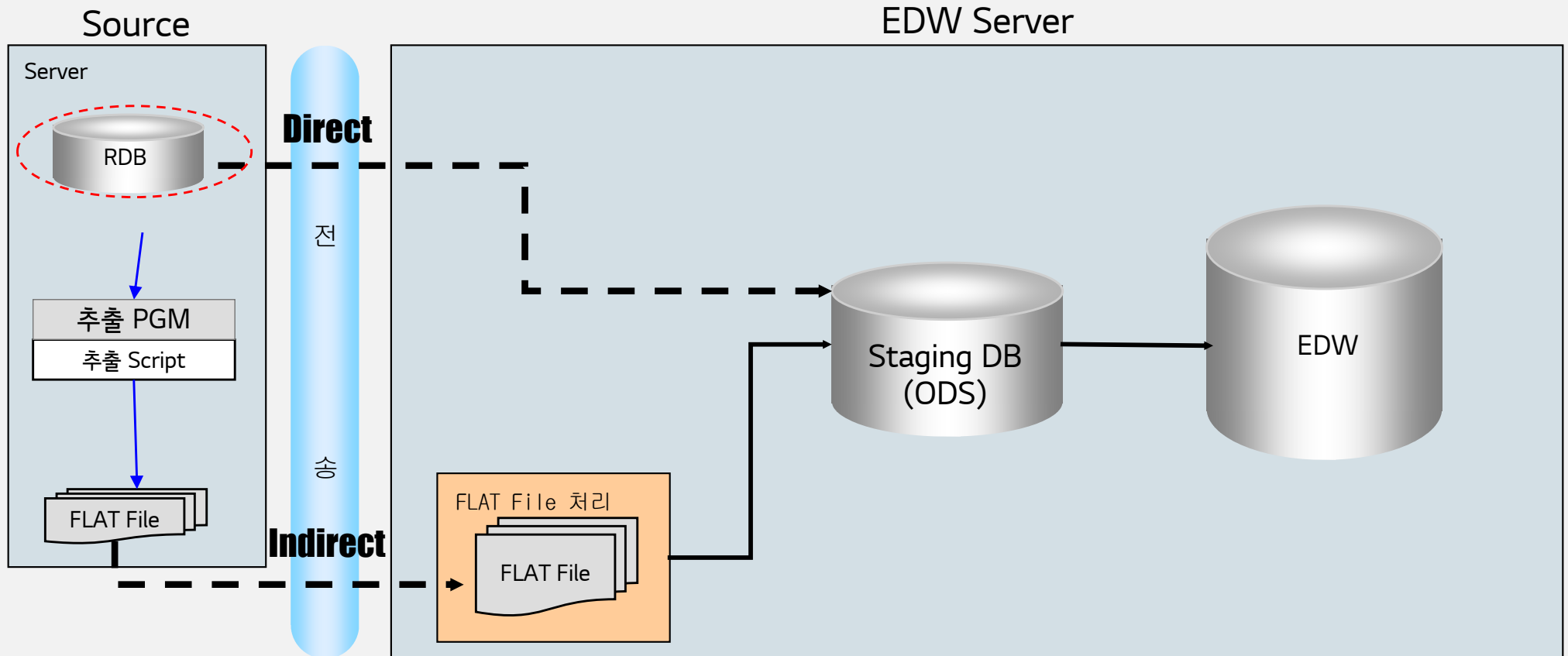
❑ ETL(데이터전처리) 프로세스 관리

- STEP별로 Job 이름, 시작/종료/소요시간, 처리건수, 성공여부 등을 메타데이터로 기록

❑ ETL(데이터전처리) 프로세스 검증

- 재무/회계 부서에서 계수의 차이에 민감해하므로 영업/마케팅보다는 이들과 협업하여 추출 프로세스와 DW 데이터를 검증하는 것도 좋은 방안

- Direct : 메인프레임 데이터를 loading을 통해 바로 Staging DB(ODS) 및 DW에 적재한다. (적은 용량의 DATA)
 - 직접 추출이 가능 하려면 VSAM / DB2 데이터에 변경일자 컬럼이 존재해야 한다.
- Indirect : Host Flat File을 Index File에 의해 Flat Data File을 Staging DB(ODS) 및 DW 에 적재한다. (대량의 DATA)
 - 야간 작업시간의 제한 (3시간)을 고려하여 호스트에서 미리 Flat File (ASCII)로 변환 후 적재한다.





1. ETL(데이터전처리)개발 표준 정의서 - 데이터 매핑 정의서(예시)

V. 프로젝트 적용 사례

No	Target Table Name	Target Column Name	Target Data Type/Length	Target Column Desc.	work group	Source System	Source Table	Source Column	Data Trasformati on	Data
1	OC_증권정보_장기	증권번호	CHAR(20)	증권번호	장기	장기보험	이_장기_증권정보	증권번호	Direct	
2	OC_증권정보_장기	변경회차	CHAR(3)	변경회차	장기	장기보험	이_장기_증권정보	변경회차		'ZZZ'
3	OC_증권정보_장기	마감일	DATE	마감일	장기	장기보험	이_장기_증권정보	마감일	Direct	
4	OC_증권정보_장기	마감년월	CHAR(6)	마감년월	장기	장기보험	이_장기_증권정보	마감년월	Direct	
5	OC_증권정보_장기	납입회수	INTEGER	납입회수	장기	장기보험	이_장기_증권정보	납입회수	Direct	
6	OC_증권정보_장기	계약일	DATE	계약일	장기	장기보험	이_장기_증권정보	계약일	Direct	보험시기
7	OC_증권정보_장기	상태일	DATE	상태일	장기	장기보험	이_장기_증권정보	현상태발생일	Direct	
8	OC_증권정보_장기	발생일	DATE	발생일	장기	장기보험	이_장기_증권정보	현상태발생일	Direct	
9	OC_증권정보_장기	영수일	DATE	영수일	장기	장기보험	이_장기_증권정보	영수일	Direct	
10	OC_증권정보_장기	회계일	DATE	회계일	장기	장기보험	이_장기_증권정보	영수일	Direct	
11	OC_증권정보_장기	청약일	DATE	청약일	장기	장기보험	이_장기_증권정보	청약일	Direct	
12	OC_증권정보_장기	보험시기	DATE	보험시기	장기	장기보험	이_장기_증권정보	보험시기	Direct	
13	OC_증권정보_장기	보험종기	DATE	보험종기	장기	장기보험	이_장기_증권정보	보험종기	Direct	
14	OC_증권정보_장기	최초입력일	DATE	최초입력일	장기	장기보험	이_장기_증권정보	최초입력일	Direct	새로생긴할목(최초청약일)
15	OC_증권정보_장기	최종입력일	DATE	최종입력일	장기	장기보험				'0001-01-01'
16	OC_증권정보_장기	책임개시일	DATE	책임개시일	장기	장기보험	이_장기_증권정보	책임개시일	Direct	보험시기
17	OC_증권정보_장기	실효기간ID	CHAR(5)	실효기간ID	장기	장기보험			Logic	실효일에서 마감일까지의 개월수 *현재상태구분 '실효'인 경우 상태일이 된다.
18	OC_증권정보_장기	경과기간ID	CHAR(5)	경과기간ID	장기	장기보험				'ZZZZZ'
19	OC_증권정보_장기	유효기간ID	CHAR(5)	유효기간ID	장기	장기보험			Logic	보험시기에서 유효일까지의 개월수 *유효일정의:상태코드 '20', '12', '99', '44', '03', '10', '11'인 경우를 제외한 나머지경우의 마감일이 유효일이 된다. * '88'인 경우는 마감일과 보험종기중 작은 값을 유효일로 한다.
20	OC_증권정보_장기	보험기간IID	CHAR(5)	보험기간IID	장기	장기보험			Logic	연금상품여부가 'Y'이면 GIKAN11에서 GIKAN12까지의 개월수이고 아니면 보험시기에서 종기까지의 개월수가된다.
21	OC_증권정보_장기	보험료납입기간ID	CHAR(5)	보험료납입기간ID	장기	장기보험	이_장기_증권정보	보험료납입기간ID	Translate	납입기간 코드성 디멘전 테이블
22	OC_증권정보_장기	모집자ID	CHAR(7)	모집자ID	장기	장기보험	이_장기_증권정보	모집자ID	Direct	

점검항목	세부점검 항목	체크리스트	점검항목 설명	점검방법
추출전략	추출전략	추출 전략을 수립하였는가?	추출전략 수립과 추출 작업의 표준이 이루어졌는지 점검함	<ul style="list-style-type: none"> • 소스타겟 매핑정의 • 추출주기와 추출방법 요건정의 • 변환정제규칙정의 • 데이터 흐름도 • 개발 표준 정의
		데이터의 추출주기와 추출방법이 정의되었으며 고객의 승인은 받았는가?		
		소스와 타겟의 테이블 간의 데이터 연관성 및 흐름을 정의하였는가?		
		소스와 타겟의 데이터 매핑과 비즈니스 로직을 정의하였는가?		
	작업 표준화	변환 및 정제 규칙을 상세하게 정의하고 적용하였는가?		
		문서작성표준, 개발환경 표준, Versioning, Naming Convention 등의 각종 표준을 수립하고 공유하였는가?		
		각각의 인터페이스 기능을 위하여 인터페이스명세서를 개발하였는가?		
데이터 품질	소스 데이터의 신뢰성&정합성	해당 Legacy 시스템 DBA로부터 Copy book을 확보했는가?	데이터 품질을 보장하기 위한 사전 분석 작업과 추출 작업 시 데이터 검증이 이루어 졌는지 점검함	<ul style="list-style-type: none"> • 데이터 신뢰도&충실도 검사 • 소스 시스템 ERD • 데이터 검증 체크리스트
		DW와 관련있는 Legacy 시스템 DBA와 인터뷰를 하였는가?		
		소스 데이터의 신뢰성과 정합성을 체크하였는가?		
		각 팩트 및 디멘전에 대한 소스 시스템을 정의하고, 소스 시스템의 데이터 구조를 파악하였는가?		
	신규/변경 데이터 검증	소스 변경사항에 대한 갱신처리 감시		
		새로운 데이터의 품질 유효성을 확보하였는가?(기본값, 오류시험, 구조, 데이터도메인)		
	에러처리	추출시 에러 데이터의 처리방안에 대해서 검토하였는가?		
		소스와 타겟의 계수가 맞는지 검증하였는가?		

점검항목	세부점검항목	체크리스트	점검항목 설명	점검방법
성능	추출전략	적재 병렬처리를 사용하였는가?	추출 성능 향상을 위한 작업이 이루어 졌는지 점검함	<ul style="list-style-type: none"> • 인덱스 정의서 • 추출 프로그램
		멀티 프로세스, 멀티 쓰레드를 이용하였는가?		
		데이터 로드 속도 등의 성능 테스트를 수행하였는가?		
		추출 성능을 향상시키기 위한 인덱스 설정을 수행하였는가?		
테스트	단위테스트	단위 테스트 계획서가 작성 되었는가?	데이터를 적재하여 데이터검증을 수행하였는지 검증하며, 외부 인터페이스, 스케줄링, 성능, 백업 및 복구에 대한 테스트가 수행되었는지 점검함	<ul style="list-style-type: none"> • 단위테스트 계획서 • 단위테스트 결과서 • 통합테스트 계획서 • 통합테스트 결과서 • 추출 백업 및 복구 절차
		작성된 ETL(데이터전처리) 프로그램 및 데이터 검증을 위해 추출할 초기 데이터를 선정하고, 데이터 적재 및 검증을 수행하였는가?		
	통합테스트	통합 테스트 계획서가 작성 되었는가?		
		외부와 인터페이스 되는 영역들에 대해 추출 테스트를 수행하였는가?		
		추출 스케줄링 된 작업들에 대해 테스트를 수행하였는가?		
		ETL(데이터전처리) 백업 및 복구에 관한 테스트를 수행하였는가?		

[Data Scientist와 AI Engineer를 위한 데이터 엔지니어링]

Module3. 데이터 모델링

I. 데이터 모델링 개요

1. 데이터 모델 개념
2. 데이터 모델 작성 지침
3. 데이터 모델 작업 절차

II. 정보계 모델링 개요

1. 등장 배경
2. 정보계 모델링의 정의
3. 운영계 모델과의 비교
4. 정보화 모델의 종류와 특징

III. 정보계 모델링 절차 및 기법

1. 정보계 모델링 절차
2. 개념 모델 설계
3. 논리 모델 설계
4. 물리 모델 설계

데이터 모델링은 정보화 시스템을 구축하기 위해 업무적으로 필요한 데이터가 무엇인지, 존재하는지를 분석하는 방법이다.

용어	설명
엔터티(Entity)	<ul style="list-style-type: none"> 시스템 구축에서 필요하고 관리대상이 되는 정보 유일한 식별자에 의해 식별이 가능해야 함 어커런스(Occurrence)의 집합으로 Occurrence가 반드시 두 개 이상 구분되는 것(예, 현대아산 병원 시스템의 '병원' 엔터티 타입 도출은 의미 없음) 업무 프로세스에서 반드시 이용되어야 함 다른 엔터티 타입과 최소 한 개 이상의 관계가 있어야 함
관계(Relationship)	<ul style="list-style-type: none"> 엔터티간의 관계 멤버십(Membership) : 사원은 부서에 소속된다 카디널리티(Cardinality) : 1;1, 1:M, M:M 관계참여도(Optionality) : Optional, Mandatory 예) 한 부서는 여러 명의 사원을 포함 한다. 한 명의 사원은 한 부서에 항상 소속된다.
속성(Attribute)	<ul style="list-style-type: none"> 기본속성 : 업무 분석을 통해 직관적으로 정의 (이름, 주민번호..) 설계속성 : 원래 속성을 표준화하여 코드화 한 속성 (성별 - 남:1/여:2) 파생속성 : 다른 속성으로부터 계산/변형되어 생성되는 속성 (매출금액=단가*판매수량) 파생 속성을 DB에 저장하는 것은 업무 프로세스 변경에 따라 데이터 무결성을 깨칠 수 있는 위험이 있음
식별자(Identifier)	<ul style="list-style-type: none"> 엔터티내에서 어커런스들을 구분할 수 있는 구분자 주식별자, 보조식별자, 내부식별자, 외부식별자, 단일식별자, 복합식별자, 원조식별자, 대리식별자

데이터 모델링(data modeling)이란 주어진 개념으로부터 논리적인 데이터 모델을 구성하는 작업을 말하며, 일반적으로 이를 물리적인 데이터베이스 모델로 환원하여 고객의 요구에 따라 특정 정보 시스템의 데이터베이스에 반영하는 작업을 포함한다.

종류	특성	산출물
개념 모델링	<ul style="list-style-type: none"> • 순수한 업무 관점의 개략적인 데이터 모델링 • DB에 독립적 • 데이터베이스를 구현하기 위한 사전단계 • 논리모델링의 바탕이 됨 	주제영역 정의 개략적인 ERD생성
논리 모델링	<ul style="list-style-type: none"> • DB/File에 적합하도록 Mapping하는 과정임 • 개념적 모델링을 논리모델링에 포함시키기도 함 	논리 ERD 생성 도메인정의 데이터사전 정의
물리 모델링	<ul style="list-style-type: none"> • 시스템을 고려한 논리적 데이터 모델을 해당 물리 데이터베이스 및 파일 구조로 전환함 • 실질적으로 데이터베이스 오브젝트와 파일을 생성 및 변경 • 데이터베이스 조건과 운영 요구사항을 최적화하는 과정임 	Database Table Space Table View 사용자 권한 참조 무결성 규칙 Index 등

데이터 모델링에서 사용되는 모든 한글, 영문 명칭은 반드시 데이터 사전에 등록된 명칭이어야 한다.

항목	작성 규칙
엔터티	<ul style="list-style-type: none"> 엔터티는 적어도 하나 이상의 다른 엔터티와 반드시 관계를 져야 함
속성	<ul style="list-style-type: none"> 필요한 데이터는 한 엔터티에 한 번만 존재해야 함(비중복성) 각기 다른 의미를 지닌 여러 개의 속성을 묶어 하나의 속성으로 정의하지 않음(Concatenated Attribute 배제) 가급적이면 Derived 성격의 Attribute는 업무적으로 중요한 의미를 지닌 경우에 한해서만 속성으로 정의하며 그러한 경우 반드시 원시 속성과 추출 공식이 기술되어야 함 정렬 순서는 PK -> Not Null -> Null 순서로 정렬하며, Null인 경우 값을 갖는 비율이 높은 속성, 업무적으로 중요한 속성, 길이가 짧은 속성을 먼저 기술함
기본 키	<ul style="list-style-type: none"> 복합 키(Minimal Set of Attribute) : 유일성을 보장해주는 최소한의 집합으로 구성함 안정성(No Change) : 한 번 설정되면 변경될 수 없음 비지능적 : 복잡한 의미를 내포하지 않아야 함 확정적 : 생성과 동시에 값을 부여할 수 있어야 함 Not Null : 반드시 값을 가져야 함 업무적으로 활용도가 높은 것, 길이가 짧은 것을 기본 키로 설정함
외부 키	<ul style="list-style-type: none"> 복합 키의 경우 자식 엔터티에서 기본 키와 일반 속성으로 분할되어 이동할 수 없음 외부 키는 자식 엔터티 내에서 이중 활용될 수 없음
도메인	<ul style="list-style-type: none"> 논리적 데이터 모델링에서는 신규로 생성한 도메인이 해당되는 기본 도메인(Blob, DateTime, Number, String)을 선택하여 한글 명칭을 등록하는 작업까지만 수행하며, 물리적 데이터 모델링 단계에서 보다 상세히 작성하도록 함 향후 도메인에 정의된 내용을 속성이 상속 받을 수 있도록 속성을 등록하는 경우 반드시 도메인을 지정하도록 하며, 데이터 모델에 도메인을 추가 하기 전에는 반드시 데이터 사전에 등록 유무를 확인하여 등록된 도메인을 사용하도록 함

데이터처리에 있어 입력,수정,삭제의 이상현상을 제거하여 안정적 처리를 보장하기 위한 방법으로 기본 원칙은 테이블에 중복된 데이터가 없도록 하는 것이며, 다양한 유형의 검사를 통해 데이터 모델을 더욱 구조화 하고 개선시켜 나가는 절차이다.

정규화 단계	정규화 내용
1차 정규화	<ul style="list-style-type: none"> 복수의 속성값을 갖는 속성을 분리
2차 정규화	<ul style="list-style-type: none"> 주식별자 전체에 종속적이지 않은 속성의 분리 부분종속 속성(Partial Dependency Attribute)을 분리
3차 정규화	<ul style="list-style-type: none"> 속성에 종속적인 속성의 분리 이전종속 속성(Transitive Dependency)의 분리
3.5차(보이스-코드 ¹⁾) 정규화	<ul style="list-style-type: none"> 주식별자 안에서 함수적 종속관계가 발생 다수의 주식별자 분리

주1) ER 모델의 발견자인 에드거 F. 커드는 1970년에 제 1 정규화(1NF)로 알려진 정규화의 개념을 도입하였다. 에드거 F. 커드는 이어서 제 2 정규화(2NF)와 제 3 정규화(3NF)를 1971년에 정의하였으며, 1974년에는 레이먼드 F. 보이스와 함께 보이스-코드 정규화(BCNF)를 정의하였다. 4NF 이상의 정규화는 이후에 다른 이론가들에 의해서 정의되었으며, 가장 최근에 소개된 정규화는 2002년에 크리스토퍼 J. 데이트, 허그 다위, 니코스 로렌츠에 의해 소개된 제 6 정규화(6NF) 이다.

비공식적으로 관계형 데이터베이스 테이블(컴퓨터 공학적 표현으로는 관계)가 제 3 정규(3NF)화가 되었으면 정규화 되었다 라고 한다.

2. 데이터 모델 작성 지침 - 1차 정규화(Remove Repeating Group)

1차 정규화는 반복적인 속성값이 나타나는 경우 별도의 엔터티로 분리하는 것이다.

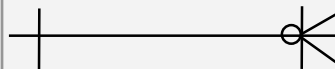
주문번호	주문일자	제품번호	제품명	재고수량	주문수량	고객번호	사업자번호	우선순위	수출여부
AB345	10.01	1001	모니터	1990	150	4520	398201	1	N
AB345	10.01	1007	마우스	9702	300	4520	398201	1	N
CA210	10.02	1007	마우스	9702	120	3280	200212	8	N
CB230	10.03	1007	마우스	9702	690	2341	563892	3	N
AD347	10.04	1001	모니터	1990	600	2341	--	3	Y
CB231	10.05	1201	스피커	2108	80	8320		2	Y

주문

주문번호
주문일자
고객번호
사업자번호
우선순위
수출여부

주문목록

주문번호(FK)
제품번호
제품명
재고수량
주문수량



- 반복속성이 분리되었으므로 주문번호를 식별자로 가지는 단일한 Row를 생성함.
- 반복되어 발생하는 속성을 분리하여 별도의 엔터티로 구성함

2. 데이터 모델 작성 지침 - 2차 정규화(Remove Partial Dependency)

I. 데이터 모델링 개요

2차 정규화는 주식별자 전체에 종속적이지 않은 속성을 별도의 엔터티로 분리 하는 것이다.

주문목록

주문번호	제품번호	제품명	재고수량	주문수량
AB345	1001	모니터	1990	150
AB345	1007	마우스	9702	300
CA210	1007	마우스	9702	120
CB230	1007	마우스	9702	690
AD347	1001	모니터	1990	600
CB231	1201	스피커	2108	80

- 2차 정규화 대상인 제품번호에만 종속적인 속성을 식별함
- 제품에 완전히 종속적인 속성을 분리하여 별도의 엔터티를 구성하고 중복데이터를 삭제함

주문

주문번호
주문일자
고객번호
사업자번호
우선순위
수출여부

주문목록

주문번호(FK)
제품번호
제품명
재고수량
주문수량

제품

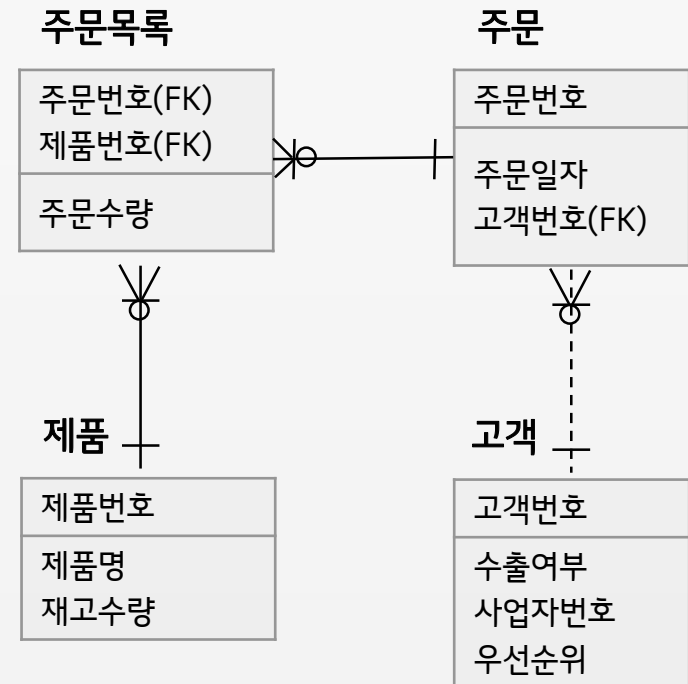
제품번호
제품명
재고수량

2. 데이터 모델 작성 지침 - 3차 정규화(Remove Transitive Dependency)

I. 데이터 모델링 개요

3차 정규화는 비식별자에 종속적인 속성을 별도의 엔터티로 분리 하는 것이다.

주문					
주문 번호	주문 일자	고객 번호	사업자 번호	우선 순위	수출 여부
AB345	10.01	4520	398201	1	N
AB345	10.01	4520	398201	1	N
CA210	10.02	3280	200212	8	N
CB230	10.03	2341	563892	3	N
AD347	10.04	2341	--	3	Y
CB231	10.05	8320		2	Y



- 고객번호는 주식별자인 주문번호에 종속된 속성이지만 자신이 결정자가 되어 다른 속성이 자신에게 종속적인 관계를 가지고 있음
- 고객에 종속적인 속성을 분리하여 별도의 엔터티를 구성하고 데이터를 정비함

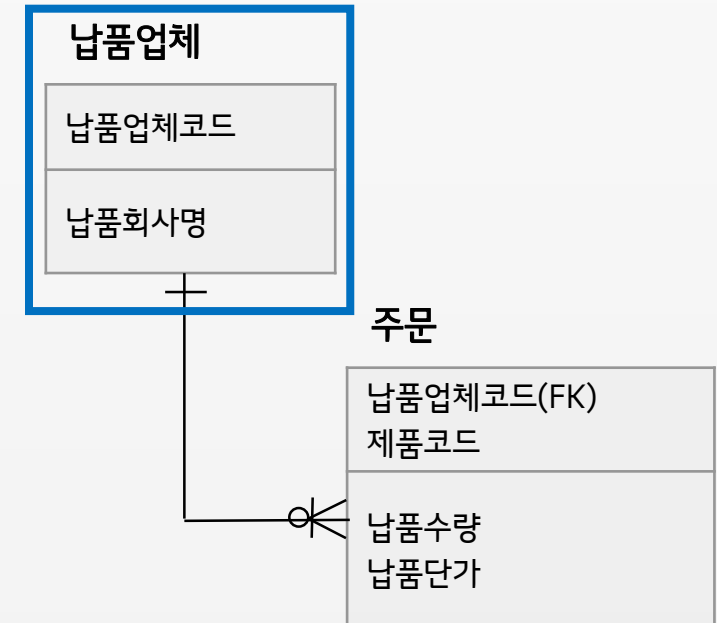
2. 데이터 모델 작성 지침 - 3.5차 정규화(보이스-코드 정규화)

I. 데이터 모델링 개요

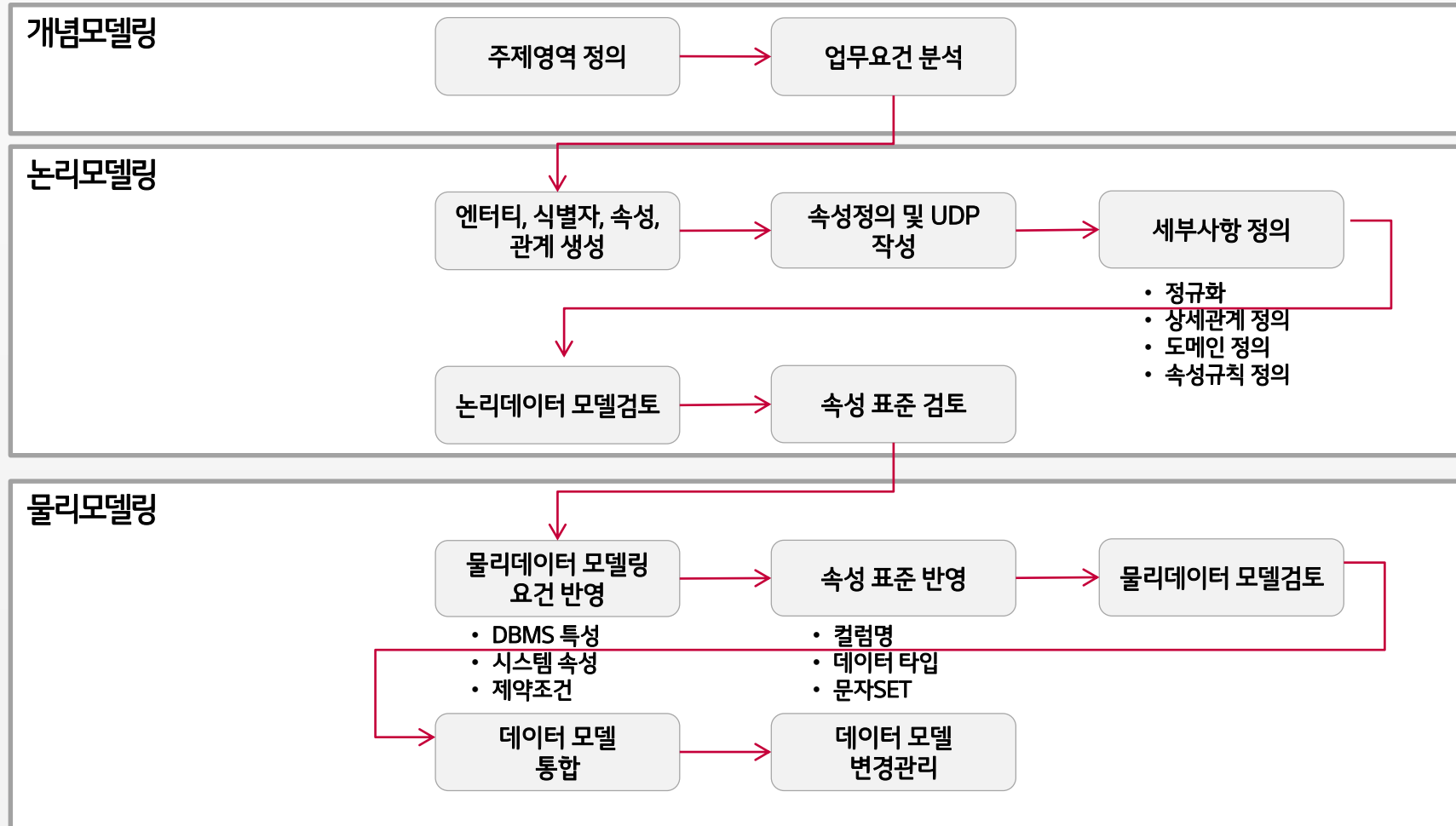
3.5차 정규화는 식별자 간에 종속성이 있는 경우 별도의 엔터티로 분리 하는 것이다.

납품

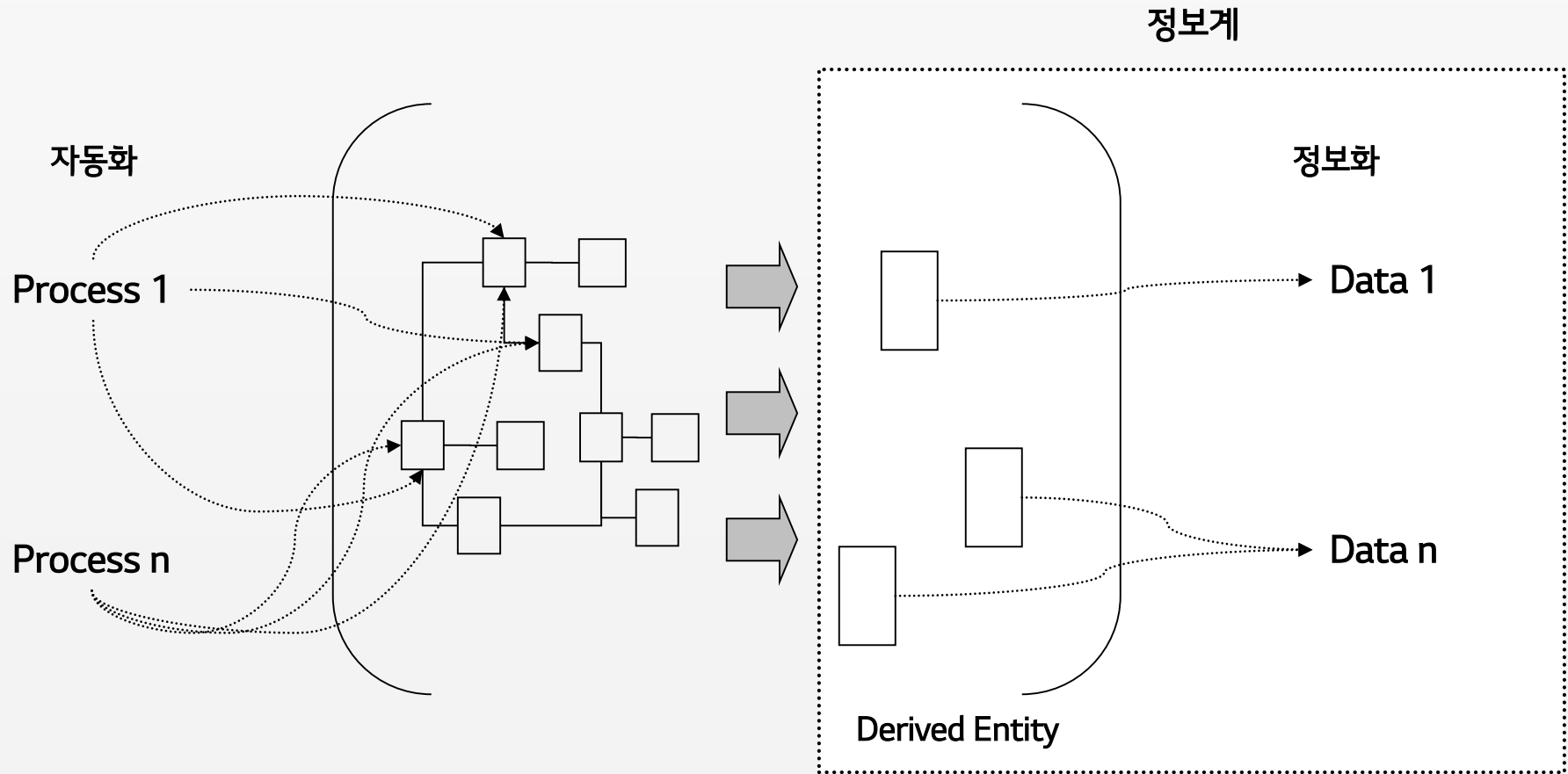
납품업체코드	납품회사명	제품코드	납품수량	납품단가
01	사반(주)	B001	10	20,000
01	사반(주)	A011	12	40,000
01	사반(주)	A012	12	30,000
02	(주).COMMUTER	B001	10	20,000
02	(주).COMMUTER	A011	9	35,000
03	시그마(주)	B001	9	21,000



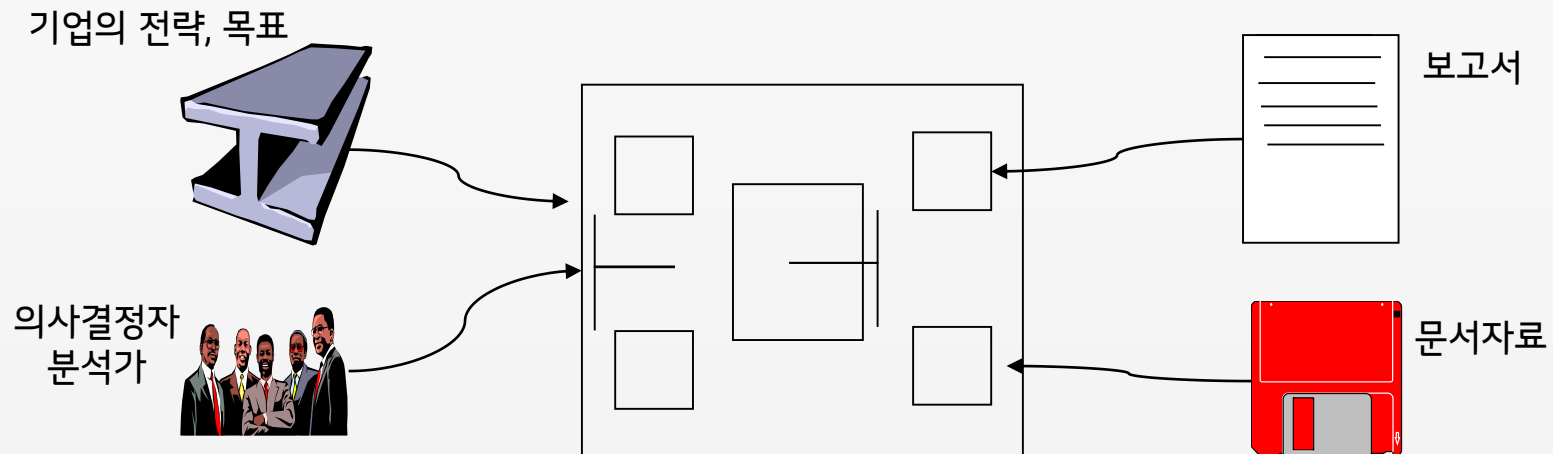
- 주식별자는 납품업체코드 + 제품코드, 납품회사명 + 제품코드, 납품업체코드 + 납품회사명 + 제품코드 모두 가능함
- 납품업체코드와 납품회사명을 납품업체 엔터티로 분리함



운영 프로세스 처리에 최적화된 정규화된 E-R 모델은 대용량 데이터 처리 및 다차원 데이터 분석 중심의 모델에 적합하지 않아 정보계 모델링을 수행함



정보계 모델링은 기업의 정보분석 구조를 체계적으로 나타내는 방법으로 기업의 데이터를 분석사용자 관점에서 인식, 분석하여 이를 이해하고 활용하기 용이한 형태로 표현하는 기법이다.



운영계 데이터 모델

- 짧고 빠른 Transaction 위주
- 작고, 사전에 정의된 질의 위주
- 현재 값 또는 최근 값들을 보관(시계열 데이터가 부족)
- 업무 단위 별로 구성되어 있어
연관분석(통합보고서)을 위해 많은 테이블
조인이 요구됨
- 사용자가 사용하기에 어려운 구조

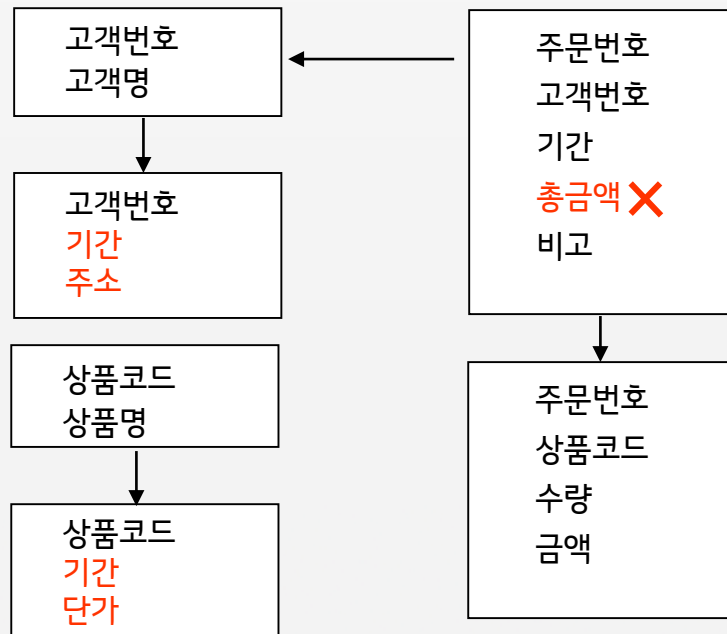
정보계 데이터 모델

- 다양한 관점의 분석 위주
- 비정형적 질의 위주
- 시계열 데이터 보관
- 통합분석 위주로 주제영역별로 구성되어 있음
- 분석관점, 분석수치 위주로 구성되어있어
사용자가 이해하기 용이한 구조

운영계 ER 모델을 반정규화하는 방법과 디멘전, 팩트 테이블로 구성된 다차원 모델로 구성하는 방법이 있다.

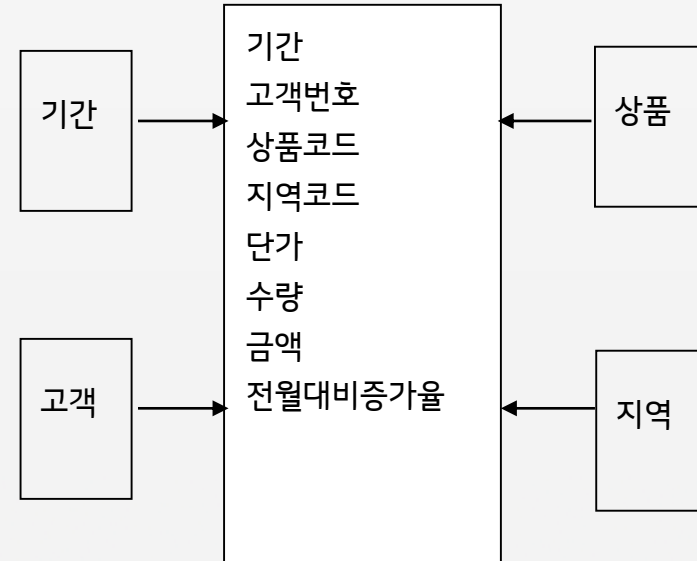
ER 모델

- 기본 키 항목에 시간요소 추가
- 자동화 처리만을 위한 데이터 항목의 제거
- 파생(예-총액) 데이터의 추가



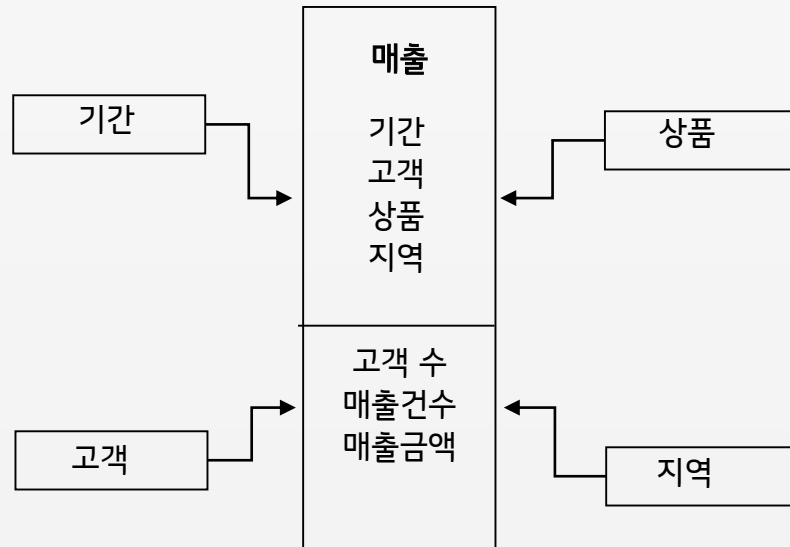
다차원(Multi-Dimensional) 모델

- 사용자 View를 기준으로 분석 관점을 제공
- 코드성 데이터는 차원(Dimension) 테이블로 정의
- 수치 데이터는 사실(Fact) 테이블로 정의

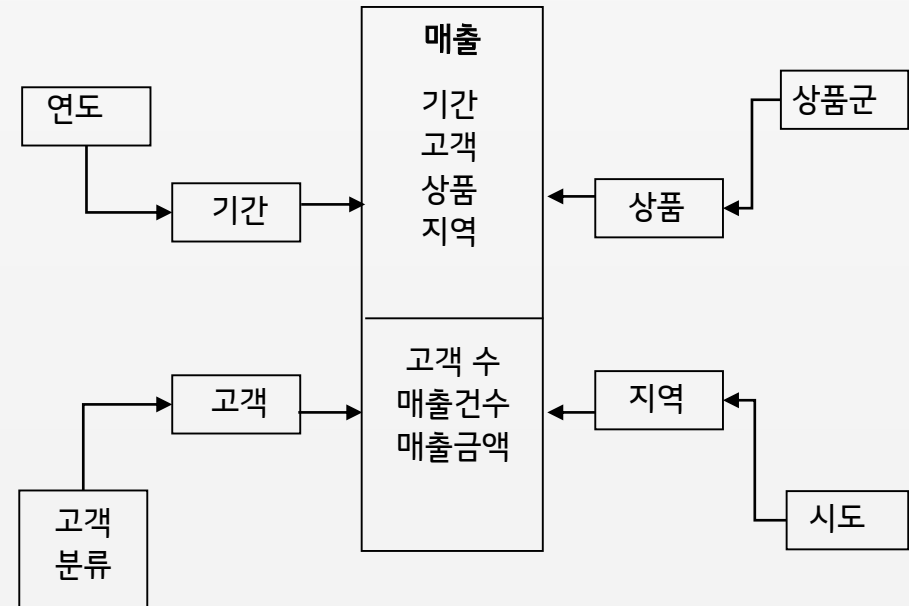


다차원 모델의 개념은 1970년대 초 AC Nielson에 의해 만들어졌으며, Ralph Kimball에 의해 대중화되었으며 Star, Snowflake와 같은 형태가 있다.

Star 스키마



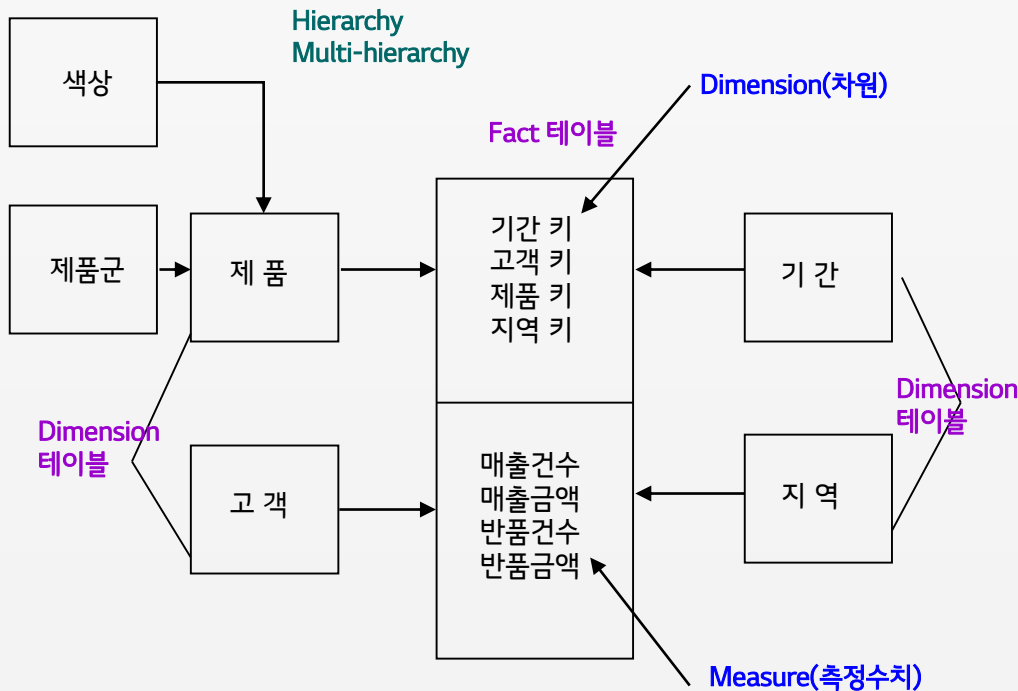
Snowflake 스키마



- Star 스키마와 Snowflake 스키마의 차이점은 디멘전 테이블의 정규화 또는 비정규화이다.
- Star 스키마는 Snowflake에 비해 디멘전 테이블과 팩트 테이블간의 Join이 작아 성능이 유리하지만 디멘전 테이블이 비대해 지는 단점이 있다.

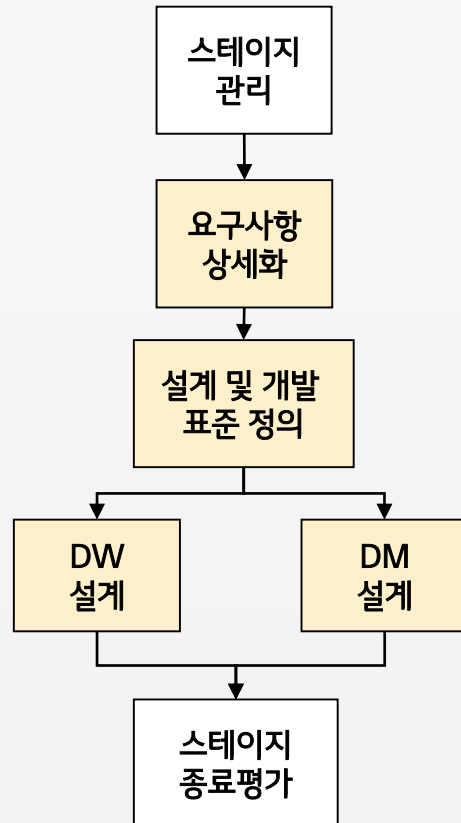
구분	Star 스키마	Snowflake 스키마
장점	<ul style="list-style-type: none"> • 간단히 모델링 • 사용자에게 친숙 • Join을 줄여 성능 향상 • 복잡한 구조가 쉽게 모델링 • 복잡한 질의 문도 사용자가 간단히 표현 가능 	<ul style="list-style-type: none"> • 저장공간의 축소 • 유연성을 제공 • 많은 도구들에 의해 지원됨 • 데이터의 중복성 제거 • 상하구조 추가/변경관리의 용이 • 관리의 용이성 제공
단점	<ul style="list-style-type: none"> • 유연하지 못함 • 데이터가 중복됨 • 데이터의 불일치 가능성 • 다양한 요약 레벨 필요 • 확장성(Scalability) 제한 • Fact 테이블간 조인의 어려움(Dimension을 공유하지 않음) 	<ul style="list-style-type: none"> • 복잡한 구조로 사용자들이 이해하기 어려움 • Fact 테이블 조회 시 차원 테이블의 추가적인 • 조인 발생으로 성능에 영향을 미칠 수도 있음

Fact 테이블의 분석관점 각각의 컬럼은 다른 테이블의 참조키(Foreign Key)가 되고, 이 참조 키를 관리하는 테이블을 Dimension 테이블이라 합니다.



용어	설명
Fact 테이블	<ul style="list-style-type: none"> Major 테이블 독립적이고 통합적인 분석의 사용자 View 관심의 수치로 항목 값의 연속적 변화를 관리 (예) 매출건수, 매출금액, 비율
Dimension 테이블	<ul style="list-style-type: none"> 분석의 관점을 관리 (예) 시간, 고객, 지역, 상품
Hierarchy	분석 관점의 계층
Multi-hierarchy	분석 관점의 다중 계층
Dimension	분석 관점
Measure	측정 지표

작업 흐름



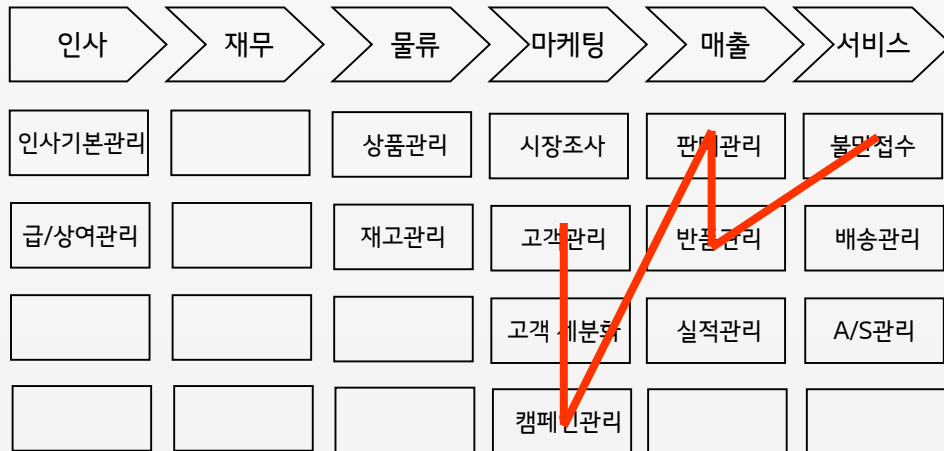
Task 상세내역

작업	수행활동	산출물
요구사항상세화	<ul style="list-style-type: none"> • 디멘전 정의 • 팩트 정의 • 팩트디멘전 매트릭스 작성 	<ul style="list-style-type: none"> • 디멘전 정의서 • 디멘전 계층 정의서 • 팩트 정의서 • 버스 아키텍처 정의서
설계 및 개발표준 정의	<ul style="list-style-type: none"> • 데이터모델링 표준정의 • ETL개발표준정의 • Application개발표준정의 	<ul style="list-style-type: none"> • 데이터모델링 표준 정의서 • DB Naming표준정의서 • ETL개발표준정의서 • Application개발표준정의서
DW 설계	<ul style="list-style-type: none"> • 원천자료 데이터목록 작성 • ODS논리모델설계 • DW논리모델설계 • 코드설계 • 물리모델설계 • DW용량산정 • 추출 및 로드전략 개발 • DW설계 검토 및 보완 	<ul style="list-style-type: none"> • Dimensional모델 • 코드설계서 • 용량산정서
DM 설계	<ul style="list-style-type: none"> • DM논리모델설계 • DM용량산정 • 추출 및 로드전략 개발 • DM물리모델설계 • DM설계 검토 및 보완 	<ul style="list-style-type: none"> • Dimensional모델 • 코드설계서 • 용량산정서

※ 주제영역정리는 분석단계의 요구사항정의 Task에서 수행함

정보계 주제영역은 정보계 구축 목적과 배경을 바탕으로 상위 주제영역을 도출하고, 세부 업무별 분석 요건을 분석하여 세부 주제영역을 정의한다.

운영계 업무 배경도



❖ 주제영역 정의 도출

- 정보계 구축 목적에 부합되는 기존 보고 자료 및 요구사항 바탕으로 정의함
- 운영계 업무 영역과 1:1로 매핑 하기 보다는 Cross Functional 하게 구성함

❖ 주제영역 예시

정보계 주제영역 도출

상위주제영역

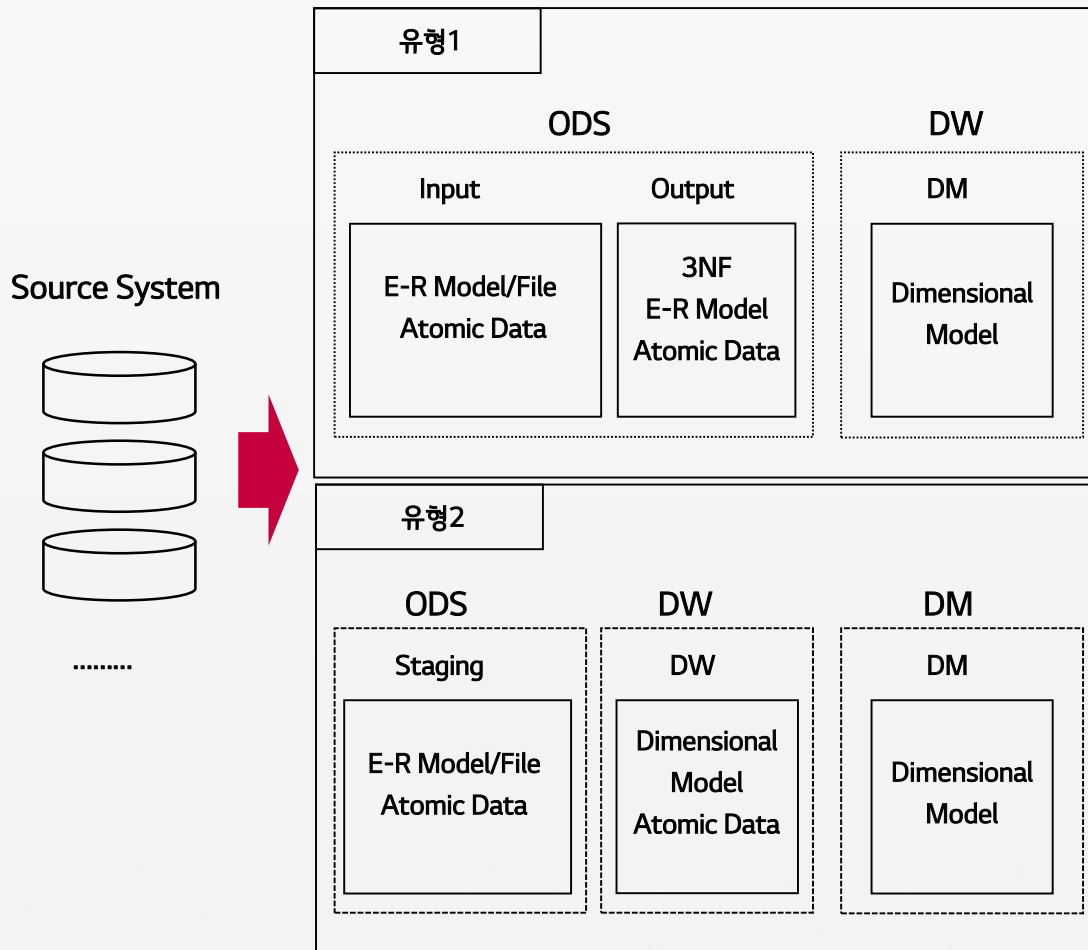
세부주제영역

고객	고객등급, CLV, 멤버십/로열티, ...
영업	매출, 조직성과, 캠페인, ...
경영	경영성과, 재무성과, ...
...	...

❖ Kimball Bus Architecture

- Kimball이 정의한 EDW Architecture Framework
- 비즈니스 프로세스와 디멘션간의 관계를 Matrix화하여 Conformed Dimension을 도출하는 방법을 통해 연관성 높은 주제영역을 정의하는 방식으로 전체 데이터아키텍처를 설계함
- 각 주제영역별 상세 팩트 테이블 또는 OLAP 큐브를 정의함

DW 전체 Data Architecture 표준 및 각 Layer별 모델 표준을 정의한다.



Data Architecture 유형 정의

- EDW 전체를 구성하는 Staging, ODS, DW, DM에 대한 표준을 정의함
- ODS의 용도 및 구축 방향을 정의함
 - ✓ 단순 Staging VS 통합데이터 저장소
- 유형1과 유형2 외에도 여러 가지 표준이 있을 수 있음

Data Schema 표준 정의

- 각 Layer별 데이터모델 표준을 정해야 함(3NF ER Model, Dimensional Model)
- ODS에 대한 정의와 함께 운영계 모델에 대한 정규화/반정규화 기준을 명시해야 함

정보계 요구사항으로부터 디멘전을 도출하고 디멘전 속성, 디멘전 Key 정의, 디멘전 변경 관리 및 계층을 설계한다..

매출일자	제품	발주처	주문수 량	매출금액	재고
20140901	G3	한양통신	1,151	1,105,000	1,201
20140902	GPro 2	매일텔레콤	989	901	1011
20140903	L70	유한통상	2,892	3,121	3,510
20140904	G3	하이마트	1,151	1,105	1,201
20140905	GPro 2	베스	989	901	1011

디멘전

- 매출일자 -> 기간 디멘전
- 제품 -> 제품 디멘전
- 제품 -> 제품카테고리 디멘전
- 발주처 -> 고객 디멘전

팩트

- 주문수량 -> 주문 팩트
- 매출금액 -> 매출 팩트
- 재고 -> 재고 팩트

정보계 요구사항으로부터 디멘전/팩트 도출

기존 보고서

수작업으로 작성하는 보고서 취합 및 디멘전 도출

AS IS 통계보고서

기존 In-house 통계시스템 보고서 중 DW 구축 대상

신규 요구사항

신규 보고서 및 분석 요건으로부터 도출

유관시스템 요구사항

DW 데이터를 활용하는 2차 시스템
요구사항(캠페인시스템, IFRS, FDS 등)

디멘전 및 계층 정의

- 디멘전 명칭 부여
- 디멘전 설명 작성
- 공통 디멘전 도출
- 디멘전 하위 속성 정의
- 디멘전간의 Hierarchy 정의(예 : 팀 - 담당 - 사업부- 사업본부)
- Multi Hierarchy 정의

정보계 요구사항으로 부터 팩트를 도출하고 팩트명, 팩트 설명, 추출/산출식 등을 정의한다..

팩트 정의

- 팩트명(한글, 영문) 정의함
- 팩트에 대한 상세한 설명 정의함
- Derived Fact인 경우에는 추출로직 또는 산출식을 정의하는게 중요함

지표	정의	산출식
평균유지 가입자수	전기 말 유지 가입자 수와 당기 말 유지 가입자 수의 평균	$(\sum [\text{유지 가입자 건수}]; <\text{당기말}> + \sum [\text{유지 가입자 건수}]; <\text{전기말}>) / 2$
무선 ARPU	무선 사용에 대한 청구 금액 기준에서 "가입자당 평균 수익 금액", ARPU(Average Revenue Per User)	$(\sum [\text{청구 금액}]; <\text{당월}>) / [\text{ACTIVE 가입자 건수}]; <\text{기간=당월말}>$
자동납부 신청건수	자동납부를 신청한 청구계약 건수	기초지표
신규고객 수	신규로 가입한 고객수	기초지표
무선수수료근거요율	대리점에 지급된 항목별 수수료 근거 요율	기초지표

팩트 종류

❖ Base Fact VS Derived Fact

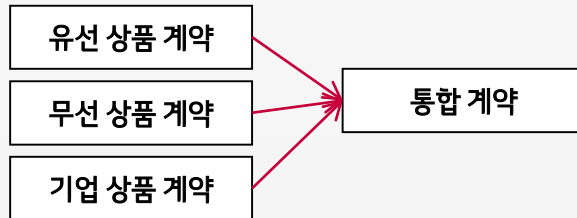
- 운영계 시스템의 Transaction 처리에서 생성된 주문 수량, 매출 금액과 같은 Measure를 Base Fact라 함
- Base Fact를 비즈니스 목적에 의해 계산하여 생성되는 팩트를 Derived Fact라고 함
- 예) ARPU
 - ✓ $\text{ARPU(Average Revenue Per User)} = \text{Total Revenue} / \text{Number of Subscribers}$
 - ✓ 통신회사의 전체 매출을 가입자로 나눈 가입자당 평균 매출액

❖ Additive, Semi Additive, Non Additive

종류	설명
Additive	상위 디멘전으로 Roll-up할 수 있음
Semi Additive	상위 디멘전 중 일부 멤버에만 Roll-up됨(예: 잔고, 재고)
Non Additive	상위 디멘전으로 Roll-up되면 데이터 오류 발생(예: 비율, 평균값)

정보계 논리 모델링은 L0 (ODS), L1(DW) 설계, DM 설계로 구분되며, 각 데이터마트는 Grain 정의, 디멘전 테이블 설계, 팩트 테이블 설계의 흐름으로 진행된다.

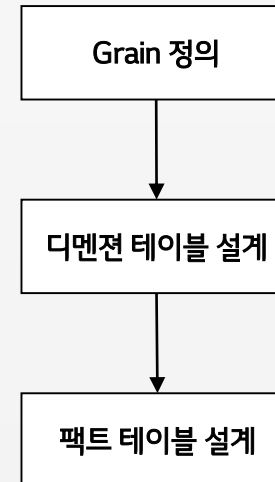
L0 (ODS), L1 (DW) 설계



❖ L0 통합 테이블 비정규화 설계

- 운영계에 산재된 데이터를 분석 목적으로 통합 테이블을 구성하면서 비정규화를 실시하기도 한다.
- 위 예제와 같이 통신 산업의 유선 상품, 무선 상품, 기업 상품별 계약이 운영계에서 별도 시스템으로 관리되고 있을 때 DW에서는 통합계약 테이블을 구성하여 ODS Layer에서 통합 작업을 수행한다.
- ODS 내 통합 테이블 구성은 비정규화를 통해 구현된다.

L2 (DM) 설계



❖ Grain 정의

- 분석 목적에 가장 적절한 데이터의 레벨을 지정한다.

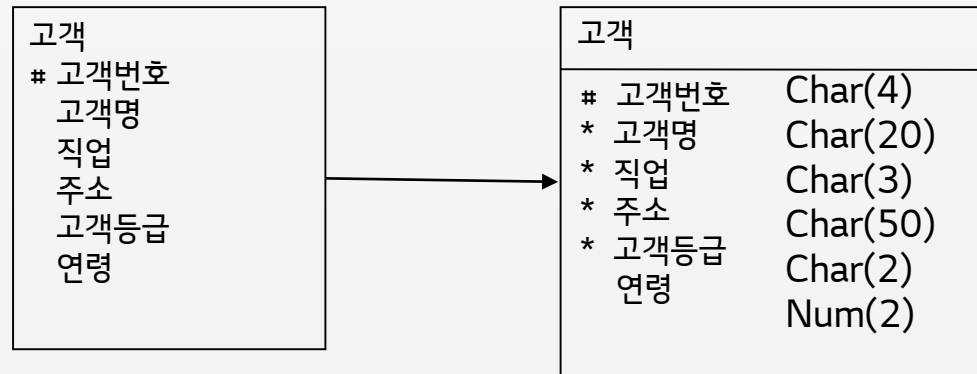
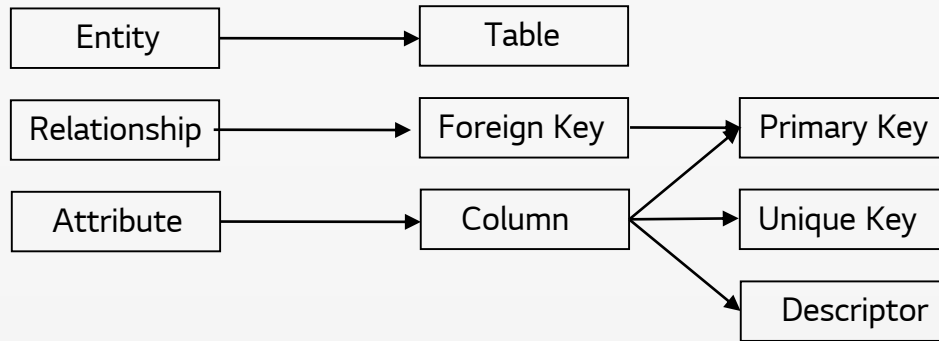
❖ 디멘전 테이블 설계

- 개념 설계에서 도출된 각 디멘전을 테이블화 시킨다.
- 각 디멘전의 Key를 정의한다
- 디멘전 테이블의 속성을 정의한다.

❖ 팩트 테이블 설계

- 개념 설계에서 도출된 각 팩트를 테이블화 시킨다.
- 해당 Measure와 관계된 디멘전을 찾고 관계를 맺는다.
- 관계된 디멘전의 PK를 FK로 구성한다.
- 팩트 테이블의 Measure 컬럼을 설계한다.

개념설계에서 도출된 주제영역별 디멘전, 팩트를 데이터 마트별 테이블과 속성으로 정의하며, 유관시스템 제공용 테이블을 정의한다.



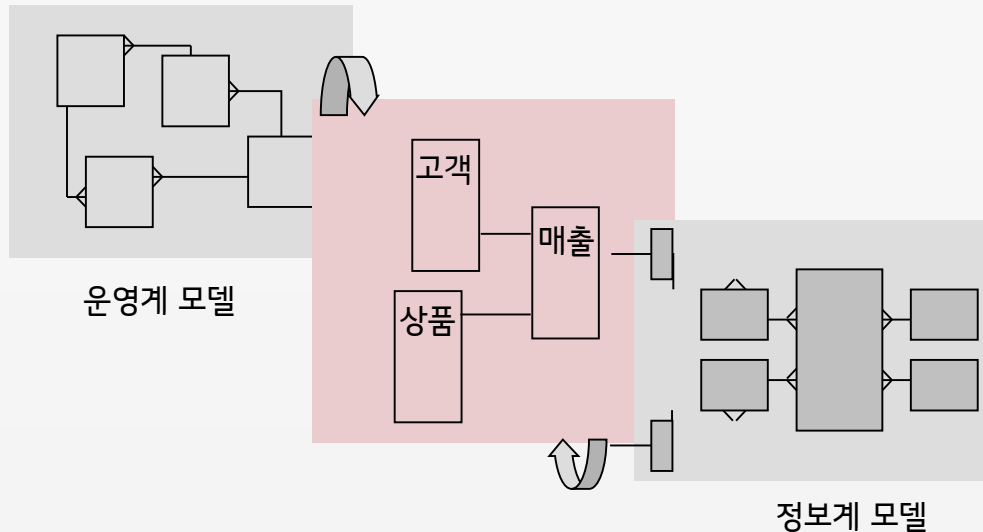
개념모델의 논리모델 매핑

- Entity -> Table, Attributes -> Column
- 기본키(Primary 정의)
- 참조키(Foreign Key 정의)
- 정보계 시스템 Data Architecture 표준 적용
 - ✓ 기본키 정책(Surrogate Key 적용여부)
 - ✓ 시스템 관리용 표준 컬럼 정의(ETL로그, 디멘전 예외값 처리)

디멘전/팩트 테이블의 Key는 엔터티에서 데이터의 유일성을 보장해주는 속성이나 속성 집합을 아래 유형에서 선택하여 설계한다.

유형	설명
후보키(Candidate Key)	<ul style="list-style-type: none"> 인스턴스의 유일성을 보장해 주며 하나의 엔터티에는 여러 개의 후보키가 존재할 수 있음 기본키가 될 수 있는 후보임
기본키(Primary Key)	<ul style="list-style-type: none"> 후보 식별자 중에서 선택된 오직 하나의 식별자를 말하며, 일반적으로 PK로 호칭되는 식별자임 Natural Key : 비즈니스적 의미가 있는 속성으로 이루어짐 Surrogate Key : 성능, 관리상의 효율성을 위해 인위적으로 생성함(=Generate Key, 인공키)
대리키(Alternate Key)	<ul style="list-style-type: none"> 후보키 중에 기본키가 아닌 것
외부키(Foreign Key)	<ul style="list-style-type: none"> 엔터티간의 관계에서 발생하는 속성으로 참조무결성에 활용되는 키
복합키(Composite Key)	<ul style="list-style-type: none"> 두개 이상의 속성으로 구성된 키

운영계 시스템과 정보계 주제영역과 Mapping을 위한 L0 (ODS) 설계를 수행한다.



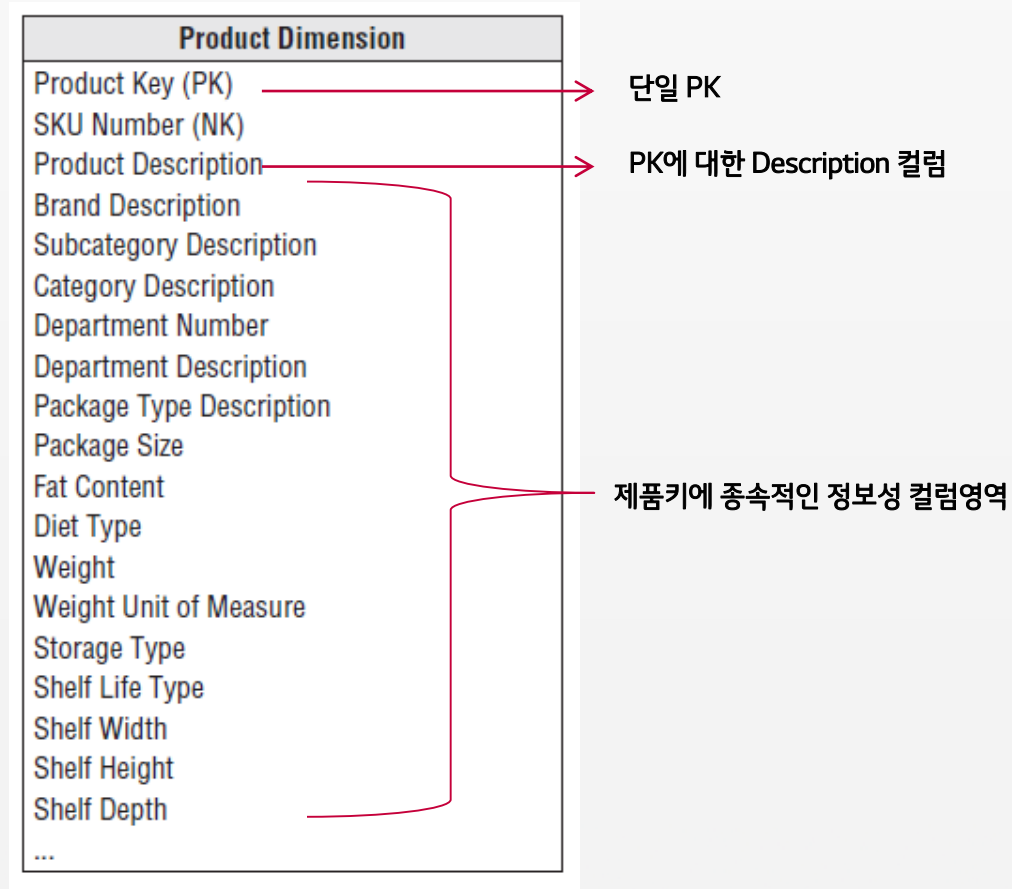
❖ L0 역할: Staging VS 통합영역

- 학계나 실제 현장에서는 ODS에 대한 다양한 정의가 존재함
- Staging 영역 VS 통합영역
- 운영계 시스템에서 고객통합, 상품통합이 되어 통합 분석을 위한 별도의 통합이 필요 없다면 운영계와 DW논리모델과의 매핑 역할만 수행하면 됨
- 채널별로 운영계 시스템이 통합되지 못하고 DW에서 통합해야 한다면, Staging 영역외에 통합 영역을 구성해야 함

운영계 VS 정보계 모델 매핑

- Gap 분석
 - 현재 시스템에 관련 정보 존재 여부 확인(외부 데이터 포함), 데이터의 상세 정도(Granularity), 데이터 변경 관리, 데이터 레벨 관리
- 소스 데이터 분석
 - 데이터 중복, 분산, 통합, 데이터 품질 및 신뢰도 분석 (참조 무결성-Referential Integrity, Domain 오류, 데이터 형식), DB size와 증가율/패턴
- Gap 해결 방안 수립
 - 협약에 의한 Ignore(오류에 대한 Default 처리, 오류 인정),
 - 데이터 정제(Cleansing)
 - ✓ 사전적 운영계 시스템 데이터 정제
 - ✓ 데이터 추출 시 정제

디멘전 테이블은 해당 디멘전의 단일 PK 컬럼과 Description 컬럼 및 부가적인 정보성 컬럼으로 구성된다.



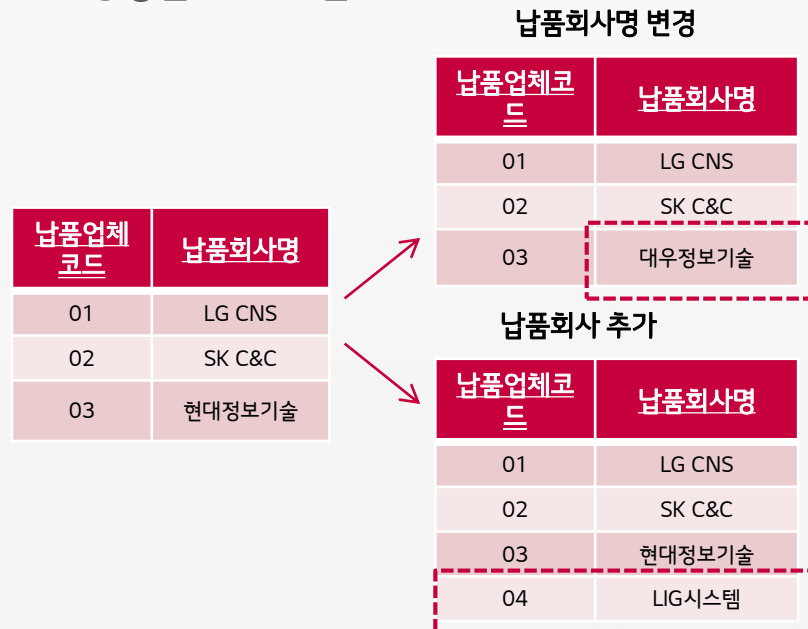
디멘전 테이블 Key 구성

- 모든 디멘전 테이블은 하나의 PK 컬럼을 가지게 된다.
- 디멘전 테이블의 PK는 관계를 가지는 팩트 테이블의 FK가 된다.

디멘전 테이블 속성 순서

- 디멘전 테이블의 속성 순서는 디멘전 테이블 PK -
 > PK Description 컬럼 -> PK에 종속적인 컬럼
- ETL 관리용 컬럼은 마지막에 위치한다.
 ✓ 예) 데이터 적재일자

L1(DW) 데이터 모델의 Surrogate Key 적용 여부는 매우 중요한 사안이며 PK구조를 변경하는 것은 불가능하기 때문에 신중하게 고려되어야 한다. 디멘전 테이블의 PK를 운영계의 Natural Key와 다른 Surrogate Key로 적용해야 하는 상황은 다음과 같다.



Surrogate key 적용

납품업체코드	납품회사명	데이터생성일
0001	LG CNS	20140901
0002	SK C&C	20140901
0003	현대정보기술	20140901
0004	대우정보기술	20140910
0005	LIG시스템	20140914

코드 이력 관리

- 운영계는 트랜잭션 처리 목적으로 최신 코드만 관리하고 DW에서 시계열 데이터 분석을 위해 과거 이력성 코드 데이터 관리하고자 할 때 별도 Key를 구성해야 함
- 왼쪽 그림과 같이 납품회사명이 변경되는 경우와 신규 납품회사명이 추가되는 상황을 모두 수용할 수 있도록 납품업체 디멘전 테이블의 Key를 Surrogate 키로 구성
- 왼쪽 사례는 디멘전 변경관리(SCD, Slowly Changing Dimension) 방안 중 Add Row 방식을 적용하였음


코드 통합 관리

- 운영계에 동일 코드가 서로 다른 이름으로 산재되어 있는 경우 DW에서는 통합된 단일 디멘전으로 설계하여야 하며 이때 Surrogate key를 적용할 수 있다.

디멘전 테이블의 소스데이터는 대부분 변화하게 되고 이러한 변화를 정보계에서 어떻게 관리할 것인지 정책을 결정하고 결정된 정책에 맞도록 설계한다.

소스시스템 데이터 변경 및 DW 요건 분석

Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Education



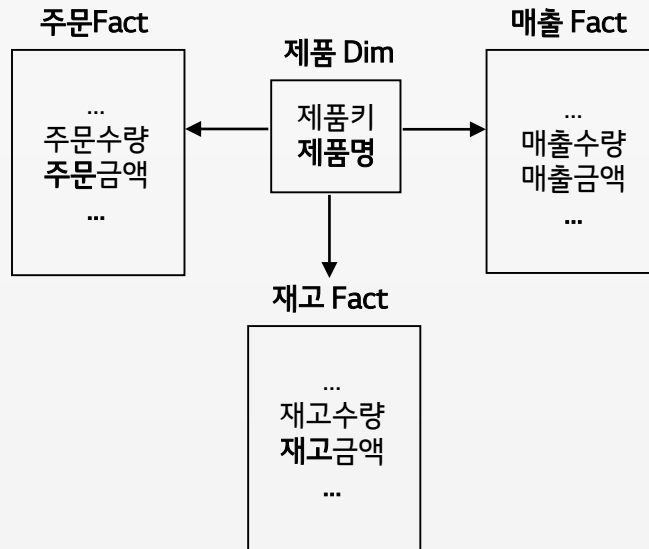
Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Strategy

- 위의 예제와 같이 소스 시스템의 제품 관리 부서가 계속 변하는 경우에 DW의 제품 디멘전도 같이 변경되어야 데이터 정합성을 유지할 수 있다.

디멘전 변경관리 방안

방안	설명
고정값 유지	<ul style="list-style-type: none"> • 원천 시스템의 변경 사항을 무시하고 고정값을 유지
Overwrite	<ul style="list-style-type: none"> • 이전 데이터 값을 무시하고 항상 최신 값으로 Overwrite함 • 과거 데이터 이력 관리가 안 되는 단점이 있음
Add Row	<ul style="list-style-type: none"> • 변경된 새로운 데이터 값을 추가하는 방식 • 과거 데이터와 최신 데이터 모두를 유지 • 데이터 용량이 늘어나는 단점이 있음
Mini Dimension	<ul style="list-style-type: none"> • 별도 디멘전 테이블로 관리함 • 본 디멘전에는 최신 데이터 값만 유지하고 별도 디멘전 테이블에 과거 히스토리 데이터를 관리함
Before 컬럼 추가	<ul style="list-style-type: none"> • Before 컬럼을 추가하여 관리 • 직전 History만 관리함

여러 팩트에서 중복해서 사용되는 디멘전이 발생하는데 단일 디멘전으로 설계하여야 하며 이를 Conformed 디멘전이라고 한다.



제품명	주문수량	재고수량	매출수량
G3	1,151	1,105	1,201
GPro2	989	901	1011
L70	2,892	3,121	3,510

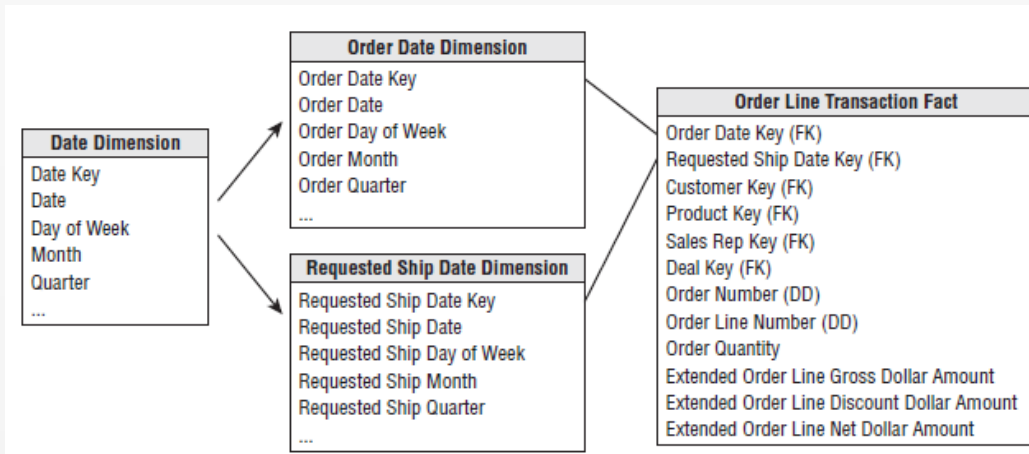
Conformed 디멘전 구성

- 왼쪽 그림과 같이 주문, 매출, 재고 Fact를 제품별로 분석하고자 하는 분석 요건이 존재할 때 제품 디멘전은 각각의 Fact 테이블에 중복으로 나타남
- 각각의 Fact 테이블에서 서로 다른 디멘전으로 설계하지 않고 단일 디멘전으로 설계하여야 여러 Fact 테이블간 연관 분석이 가능함

업무 확장 기준으로 활용

- 현재 구현된 데이터 마트와 별도로 새로운 주제영역을 구축하는 경우 이러한 Conformed 디멘전을 활용하면 업무 확장시 통합된 View를 제공할 수 있음
- 따라서 신규 주제영역이나 데이터마트 구현 시 기존 Conformed 디멘전과의 통합 설계를 고려해야 함

물리적으로는 동일한 데이터를 보유한 디멘전이 아래 예시와 같이 여러 버전의 디멘전이 필요한 경우 Role-Playing 디멘전으로 처리하여 구성한다.



View나 Alias를 통해
Role Playing 디멘전
처리

Role Playing 디멘전 구성

- Order Line Transaction 팩트 테이블에는 여러가지 Data FK를 가지고 있다.(Order Date, Requested Ship Date)
- 해당 디멘전과 같이 데이터는 동일하고 비즈니스 의미만 다른 경우 View를 구성하거나 SQL 구문에서 Alias로 처리할 수 있다.

```
create view order_date
(order_date_key, order_day_of_week, order_month, ...)
as select date_key, day_of_week, month, ... from date
```

- 대부분의 OLAP 솔루션은 자체 메타 데이터에서 Alias 테이블로 등록하는 기능을 제공한다.

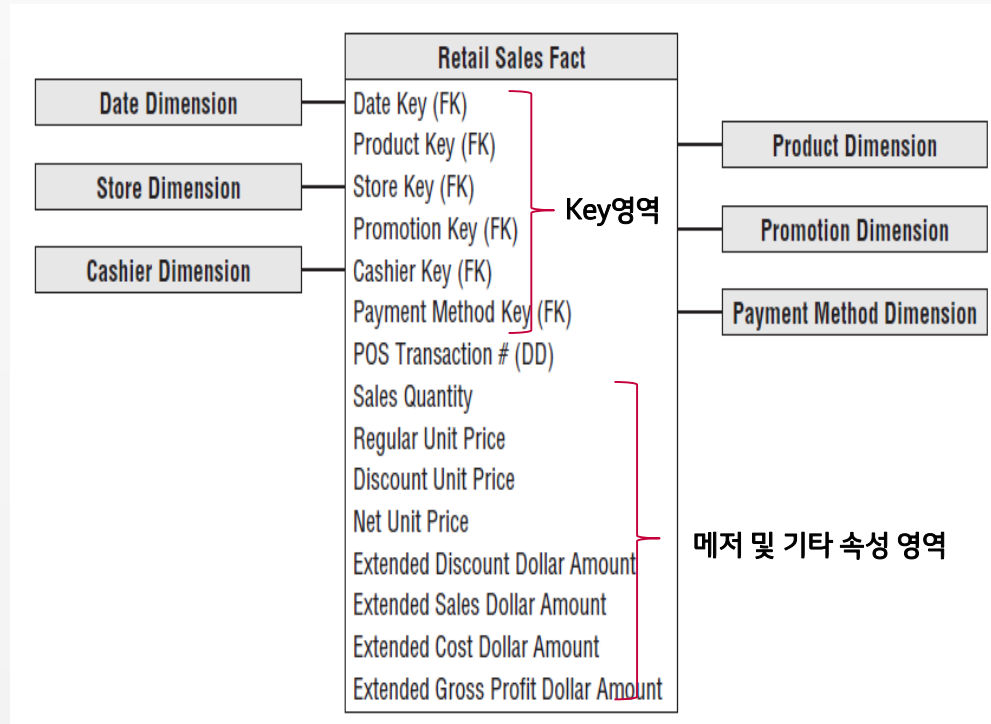
디멘전 테이블의 일부 속성을 상세하게 설명하는 Outrigger 디멘전은 팩트 테이블과 FK 관계를 가지지 않고 디멘전 테이블과 관계를 가지게 구성한다.

Fact	Date Dimension	Country-Specific Date Outrigger
Date Key (FK) More FKs ... Facts ...	Date Key (PK) Date Day of Week Day Number in Epoch Week Number in Epoch Month Number in Epoch Day Number in Calendar Month Day Number in Calendar Year Day Number in Fiscal Month Last Day in Fiscal Month Indicator Calendar Month Calendar Month Number in Year Calendar Year-Month (YYYY-MM) Calendar Quarter Calendar Year-Quarter Calendar Year Fiscal Month Fiscal Month Number in Year Fiscal Year-Month Fiscal Quarter Fiscal Year-Quarter Fiscal Year ...	Date Key (FK) Country Key (FK) Country Name Civil Name Civil Holiday Flag Civil Holiday Name Religious Holiday Flag Religious Holiday Name Weekday Indicator Season Name

Outrigger디멘전 구성

- 디멘전 테이블은 별도의 참조 디멘전을 가질 수 있는데 이를 Outrigger 디멘전이라 함
- 왼쪽 그림의 Country-Specific Date Outrigger 디멘전은 Date 디멘전의 Date Key를 FK로 받아 국가별로 다른 휴일 등을 나타내도록 구성됨

팩트 테이블 Key는 Relation을 맺고 있는 여러 개의 디멘전 테이블의 PK를 FK로 구성하며, 속성 순서는 PK -> Non-key 속성 -> Measure 순서이다.



팩트 테이블 Key 구성

- 팩트 테이블은 Relation을 맺고 있는 여러 개의 디멘전 테이블의 PK를 FK로 받아 PK를 구성한다.

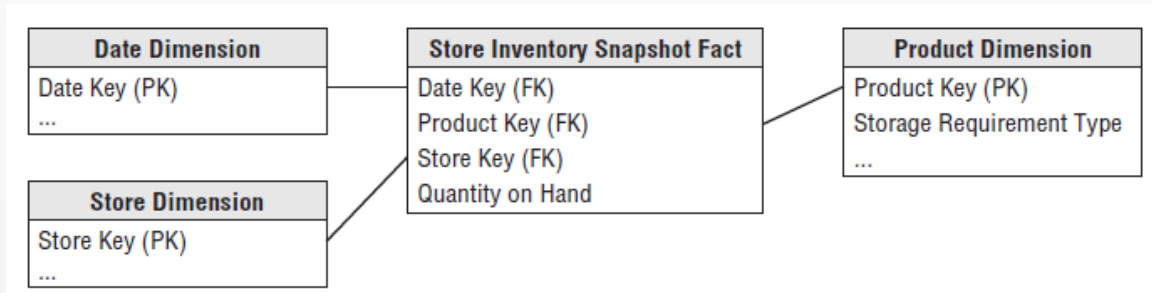
팩트 테이블 속성 순서

- 팩트 테이블의 속성 순서는 디멘전테이블 PK(FK) -> Non-key 컬럼 -> Measure 컬럼 -> 관리용 컬럼
 - 일반적으로 Data Key가 맨처음 위치한다,
 - ETL 관리용 컬럼은 마지막에 위치한다.
- ✓ 예) 데이터 적재일자

❖ FK가 아닌 정보성 속성(Non-key 속성)

- Fact 테이블에는 FK가 아닌 정보성 속성이 존재할 수 있다.
- 예를 왼쪽의 Retail Sales 팩트에 'Discount 여부'와 같은 컬럼이 있다고 가정하면 디멘전으로 구성할 만큼 중요하지는 않지만 총매출액을 계산할 때 필요한 경우 추가한다.

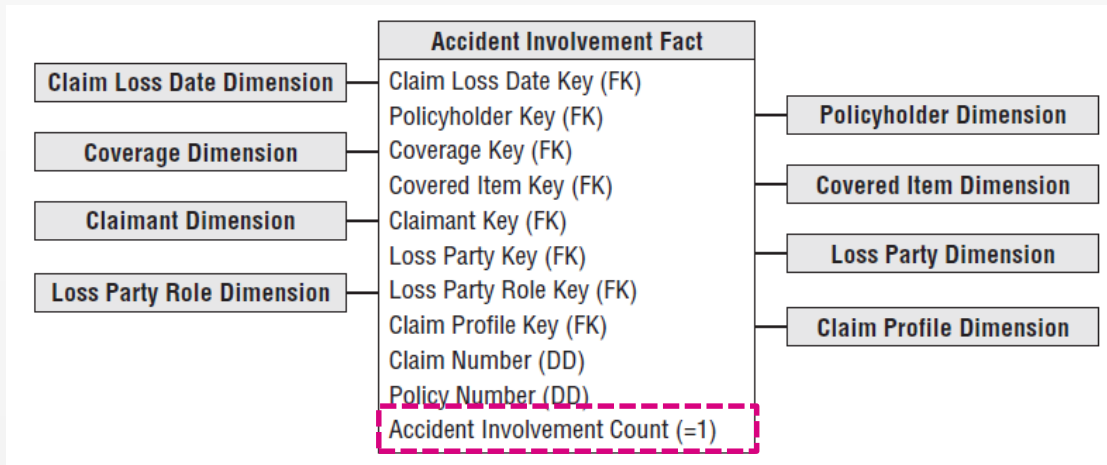
실시간으로 데이터가 발생하는 운영계 데이터를 시계열 관점으로 분석하고 할 때 DW에서는 Snapshot Fact로 구성한다.



- ❖ Store Inventory Snapshot Fact
- 매장별 재고현황을 일별 snapshot하여 관리함

유형	설명	Key 구조
Transaction Fact Table	<ul style="list-style-type: none"> 운영계 Transaction 테이블과 동일한 Grain을 가진 Fact Table 대용량 데이터이므로 시계열 분석 시 성능이 떨어짐 	<ul style="list-style-type: none"> 소스 테이블과 동일
Periodic Snapshot Fact Table	<ul style="list-style-type: none"> 일, 월, 분기, 반기, 년 등의 기간 디멘전 중 분석 요구사항에 맞는 일정 시점의 데이터로 집계됨 데이터 생성시점(예, 월말)에만 운영계와 데이터가 일치 함 	<ul style="list-style-type: none"> 기준 기간 디멘전

아래 보험 사고 팩트 테이블과 같이 사고에 따른 보상번호와 증권번호를 조회하고자 하는 분석요건을 위해 팩트 테이블을 설계한 경우에 특정 Measure가 없는 경우 인공적인 Measure(=1)를 생성하여 구성한다.

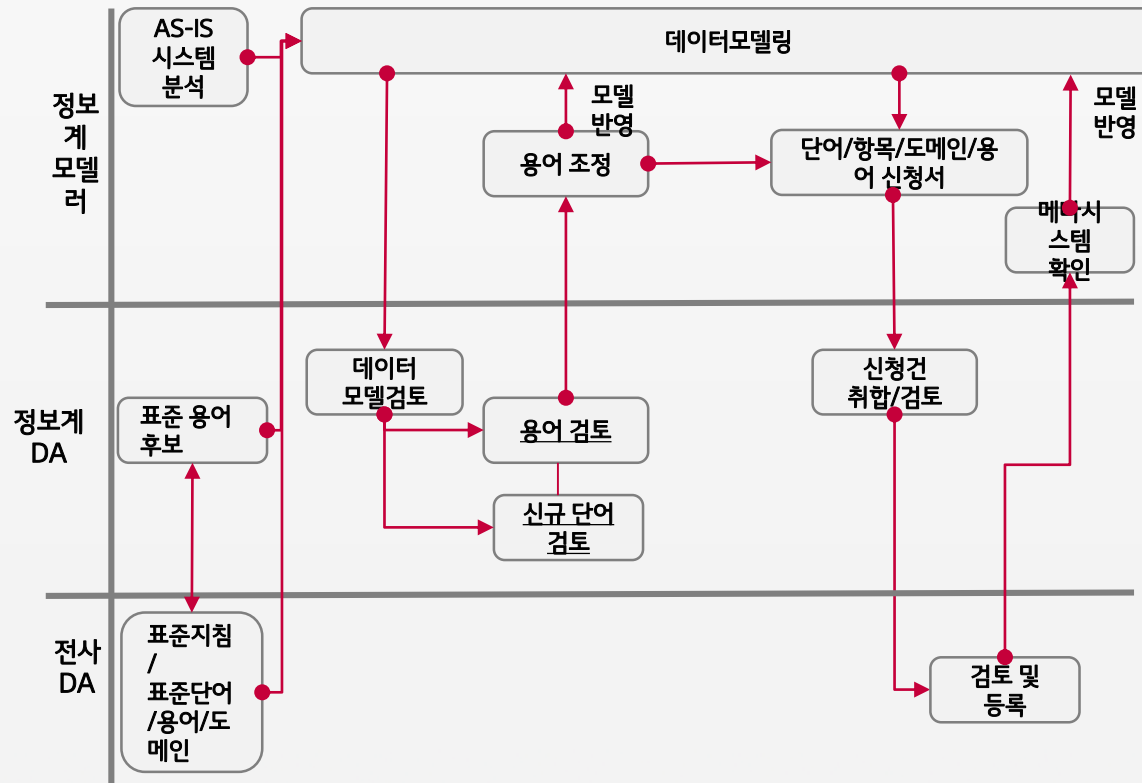


❖ Accident Involvement Fact

- 사고 발생에 따른 손실 보상 현황을 관리하며 보상 번호와 증권번호 목록을 보기 위한 목적임
- Accident Involvement Count(사고연루건수) 컬럼의 값은 모두 1이 됨
- 관련 용어
 - ✓ Claim Loss Data : 손실보상일자
 - ✓ Policyholder : 보험계약자
 - ✓ Coverage : 보상범위
 - ✓ Claimant : 청구인

정보계 모델러는 테이블 설계 시 전사 DA 표준가이드에 명시된 표준화 절차에 의거하여 표준화를 수행한다.

표준화 절차



표준화 Task

- 용어 표준화 : 업무별/시스템 별로 상이하게 사용하고 있는 용어를 표준지침에 따라 표준화함
- 도메인 표준화 : 시스템의 모든 속성에 대하여 데이터의 성격을 분류할 수 있는 도메인을 정의하고 표준화함

정보계 모델러는 물리모델링가이드에 따라 성능과 데이터정합성을 고려하여 물리모델을 설계한다.

Task	설명
물리모델 전환	<ul style="list-style-type: none"> 모델링 도구(ER-Win)에서 물리모델 적용 DBMS 설정하면 자동 전환됨
테이블 상세 설계	<ul style="list-style-type: none"> 데이터 증가량, 트랜잭션 빈도 및 업무 중요성을 고려한 중점 관리 대상 테이블 선정 및 성능 및 관리를 고려한 테이블 상세설계 용량 산정 : 각 테이블 별 예상 초기건수, 주기, 증가건수를 산정 테이블 UDP 정보생성 <ul style="list-style-type: none"> ✓통합모델러/DBA가 제시한 UDP(User Defined Property)를 ER-WIN에 입력 ✓예 : 보관주기, 초기건수, 증가건수 등 컬럼 순서 지정 <ul style="list-style-type: none"> ✓자주 사용되는 컬럼을 앞으로 이동 ✓NULL이 허용되는 컬럼은 저장공간 효율성을 위해 Table의 뒤쪽에 위치
NOT NULL, Default 지정	<ul style="list-style-type: none"> NOT NULL/NULL에 대한 허용여부 지정 데이터가 입력되는 시점에 반드시 값이 채워지는 것은 NOT NULL, 향후에 채워지는 것은 NULL로 지정 금액, 수량 등 숫자 칼럼은 DEFAULT(0), NOT NULL 무결성 제약처리를 함 (번호,일련번호,아이디 속성 제외)
인덱스	<ul style="list-style-type: none"> 데이터 활용패턴 분석에 의해 정교화된 인덱스 생성을 통해 질의 응답속도 개선 기회를 마련하여야 함
데이터 파티셔닝	<ul style="list-style-type: none"> 테이블 분할과 데이터의 분산 적재와 같은 데이터 Partitioning을 통해 질의 응답속도 개선 기회를 마련하여야 함
요약 테이블 설계	<ul style="list-style-type: none"> 조회 응답시간 향상, 자원 효율 최적화, 분석작업 능력강화를 위해 추가 생성

테이블 분할과 데이터의 분산 적재와 같은 데이터 Partitioning을 통해 질의 응답속도 개선 기회를 마련하여야 한다.

기능	설명	고려사항
테이블 수평/ 수직적 분할	컬럼에 대한 사용빈도의 차이가 많고, 각기 다른 사용자 그룹이 테이블 내의 특정한 부분만을 지속적으로 사용하는 경우에 고려함	<ul style="list-style-type: none"> 분할되어 있는 데이터를 동시에 요구하는 경우 다중 테이블을 조인해야 하므로 SQL 구문이 복잡해지고, 응답시간이 길어질 수 있음 중복 키(Primary Key) 관리로 인한 디스크 낭비를 가져올 수 있음
데이터 분산 적재	데이터 분산 적재는 데이터의 크기가 큰 테이블을 물리적으로 나누어 관리하는 것으로 대용량 데이터베이스 처리에 필수적인 기능임	<ul style="list-style-type: none"> 데이터 특성(지역적, 매체별 적재정도 등), 질의 패턴에 대한 분석이 선행되어야만 효과를 극대화 시킬 수 있고, 물리적인 데이터 분산 적재를 통해 Disk I/O를 최소화하는 방안이므로 시스템(H/W) 구성과도 밀접한 관계를 가짐

❖ 정보계 DBMS에 종속적인 파티션 설계

- Oracle, Sybase와 달리 Teradata, DB2와 같은 MPP기반 DBMS는 모델링에 의한 파티션보다는 DBMS에서 제공하는 Node별 분산환경의 자동파티션기능을 적용하게 된다.

데이터 활용패턴 분석에 의해 정교화된 인덱스 생성을 통해 질의 응답속도 개선 기회를 마련하여야 한다.

기능	설명	고려사항
인덱스 생성	데이터의 검색을 위해 전체 테이블을 읽지 않고 해당 행의 주소를 통하여 직접 접근하여 질의 속도를 개선시키기 위해 사용하는 기능	<ul style="list-style-type: none"> 데이터 분산 적재와 동일하게 데이터 특성, 데이터 질의 패턴에 대한 분석이 선행되어야만 작업의 효과를 극대화 시킬 수 있음 복잡하고, 비정형적이고 자료 요구가 많은 DW에서는 신중함이 요구됨 인덱스 저장을 위한 디스크 낭비와 과부하가 발생할 수 있음

❖ 정보계 DBMS에 종속적인 인덱스설계

- Oracle, Sybase와 같은 DW에 특화된 Index를 제공하는 경우와 Teradata, Vertica와 같이 별도 인덱스가 없는 DBMS도 있으므로 실제 적용되는 DBMS에 종속적이다.

감사합니다

