데이터 표준화 및 협업 프로세스 개선

강아현

목차

1. 프로젝트 개요

일관성 있는
데이터 환경 구축

데이터 기반
의사결정 효율 제고

목표

- 전사 데이터의 일관된 명명 규칙 및 저장 형식 정의
- 데이터의 동일한 이해 및 연계성 확보
- 데이터 기반 의사결정의 효율성 제고

핵심 키워드

- 표준 단어 · 용어 · 도메인 · 코드
- AI 활용성 (자연어 쿼리 품질 향상)
- 협업 방식 개선 및 기록 자산화

2. 주요 역할 및 책임

데이터 표준화

- 표준 단어/용어 정의
- 프로세스 설계
- Dataware, 쿼리파이로 컬럼 현황 분석

작업 프로세스 개선

- Python, Kiwi 활용
- 표준 단어 후보 자동 추출

협업 방식 개선

- Wiki, Slack 캔버스 적극 활용
- 회의록 / 개인일지 도입

가이드라인 정립

- Q&A 카테고리화
- 표준화 가이드라인 수립

3. 데이터 표준화 프로세스

Phase 1 표준 단어 정의

최소 단위

용어 정의

Phase 2 표준 용어 정의

단어 조합(컬럼)

메타데이터 정의

Phase 3 표준 코드값 정의

코드 도메인

데이터 정의

Phase 1: 표준 단어 정의

표준 단어는 일정한 의미를 가지는 최소 단위의 용어로, 표준 용어를 구성하는 기반이 됩니다.

표준화 대상 DB 전체 컬럼 추출

column_id	column_comment
created_at	생성일시
updated_at	수정일시

컬럼을 최소 단위로 분할

ASIS 영문 단어	column_comment
created	생성일시
at	생성일시
updated	수정일시
at	수정일시

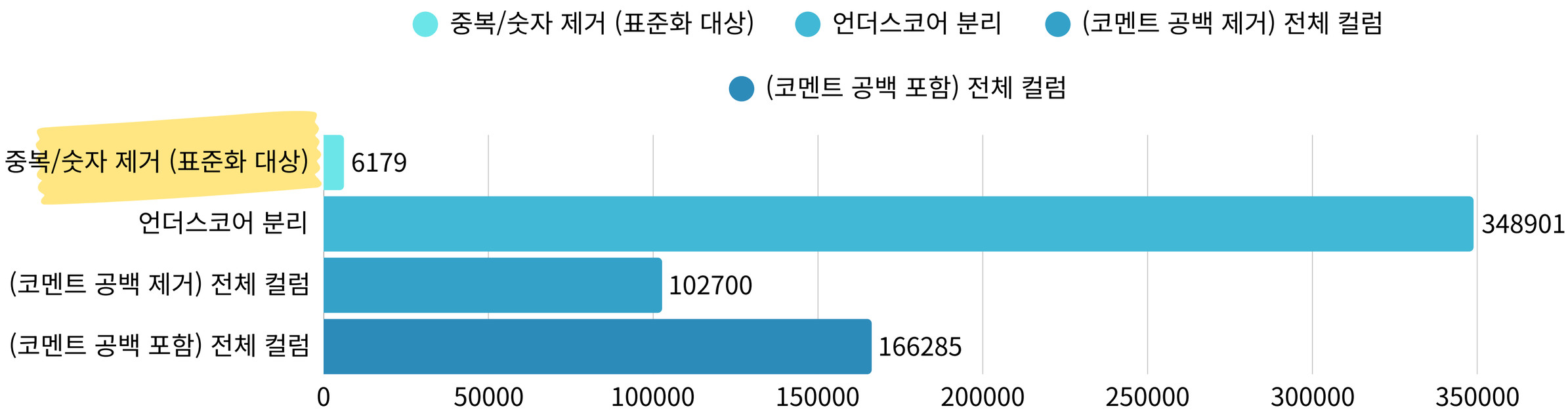
표준 단어 정의

한글명	영문명
created	생성
at	일시
updated	수정

추출한 column_id를 언더스코어(_) 기준으로 split

표준 단어 생성 과정을 통해 표준 사전을 구축

1-1. 표준화 대상 단어 선정



표준화 대상 단어 선정 과정

- DB 내 모든 컬럼 조회 후 언더스코어(_) 기준으로 단어 분리
- 형태소 단위로 분리하여 표준화 대상 선정

1-2. 표준 단어 생성

>> 1차 (정성적 판단 기반)

프로세스

1. 표준화 대상 단어가 포함된 컬럼명 탐색
2. 해당 컬럼명만 필터링
3. 컬럼 설명을 참고해 정성적으로 단어 정의
4. 추정 한글 단어 사용 비율 정성적 파악
5. 유사 비율 단어가 여러 개일 경우 새로운 영어 물리명 생성

결과 예시

“created_at”, “modified_at” 컬럼 분석

→ “at”은 ‘일시'라는 의미로 사용

→ 논리명: 일시, 물리명: at 로 정의

전사 DB 기준 한글 단어 기반 영문 물리명 선정

SQL 예시

```sql
# '가능'으로 쓰이는 영어 단어 중 'yn'의 사용 빈도 확인
select *
from (
  select column_id, column_comment
  from table
  where column_comment like "%가능%"
    and (column_id like "%yn%" or column_id like "%able%")
) a
order by 2 desc;
```

출력 예시

한글 단어를 입력하세요: 속성

ko_word	exploded_column_id	count	instance_names
속성	**ATTR**	**32**	[inst_1, inst_2, inst_3 …]
속성	property	23	[inst_4, inst_5, inst_6 …]

→ 표준 단어 결정: **논리명: 속성 / 물리명: ATTR**

Python 예시

```python
# 한글 단어로 가장 많이 쓰이고 있는 영문명 찾기
def db_en_most(ko, df):
    # 1. 추정 한글 단어(ko)를 포함한 column_comment 행 필터링
    filtered_df = df[df['column_comment'].str.contains(ko, na=False)]

    # 2. exploded_column_id 별 카운트 및 instance_name 리스트 집계
    if not filtered_df.empty:
        exploded_id_info = filtered_df.groupby('exploded_column_id').agg(
            count=('exploded_column_id', 'count'),
            instance_names=('instance_name', lambda x: list(x.unique()))
        ).reset_index()
        # 카운트 기준으로 내림차순 정렬 후 상위 20개 추출
        exploded_id_info = exploded_id_info.sort_values(by='count', ascending=False)
        top_20_list = exploded_id_info.head(20)[['exploded_column_id', 'count', 'instance_names']].values.tolist()
    else:
        top_20_list = []
    return top_20_list

def main():
    # column_id를 언더바 기준으로 split한 전체 데이터 가져오기
    file_path = 'std_term_to_select_en_word_single_data.csv'
    df = data_load(file_path)

    # 찾을 추정 한글 단어 입력 받기
    to_find_en_word = str(input("한글 단어를 입력하세요: "))

    # 결과 반환 데이터프레임 생성
    word_df = pd.DataFrame(columns=['ko_word', 'exploded_column_id', 'count', 'instance_names'])
    result_list = db_en_most(to_find_en_word, df)

    if result_list:
        word_df['ko_word'] = [to_find_en_word] * len(result_list)
        word_df['exploded_column_id'] = [item[0] for item in result_list]
        word_df['count'] = [item[1] for item in result_list]
        word_df['instance_names'] = [item[2] for item in result_list]
    else:
        print("검색 결과가 없습니다.")
    print(word_df)
```

>> 2차 (정량적 분석 기반, Kiwi 활용)

도입 배경

- 1차 정성적 판단의 한계 보완
- 형태소 분석 기반 통계 → 단어 정의의 객관성 및 정확성 강화

프로세스

1. ASIS 영문 단어가 포함된 모든 comment 리스트 추출
2. Kiwi로 모든 comment를 토큰 단위로 분리
3. 토큰별 카운트 후 상위 20개 추출
4. 각 토큰의 비율 계산 = (토큰별 카운트 / 전체 단어 수) * 100
5. 통계 기반 한글 단어 선정

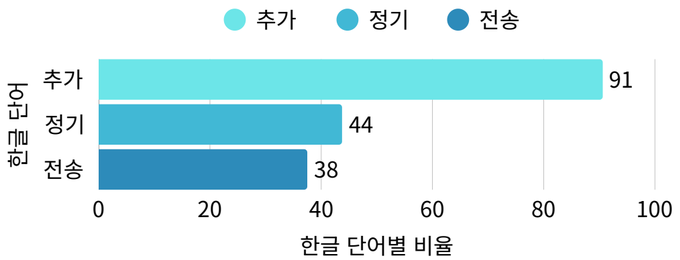
결과 예시

예시: add가 들어간 comment 전체 수집

column_id	word	column_comment
add_xxxxxx_xxxx	add	부가상태코드
add_xxxxx	add	추가건수
add_xxxxxx_xx	add	추가노출여부
add_xxxx	add	추가정보
…	…	…

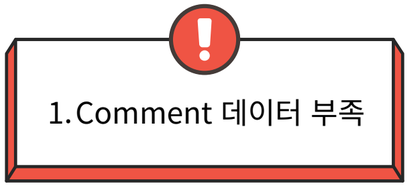
→ 출력:
{’add’: [’부가상태코드’, ‘추가건수’, ‘추가노출여부’, ‘추가정보’, …]}

'add'의 연관 한글 Token 빈도



→ **'add'**의 표준 한글 단어는 **'추가'**로 선정

🚨 문제점 및 한계



예: abusing 조회 결과

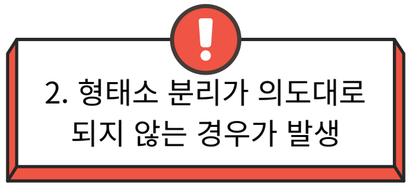
- abusing 포함한 컬럼의 comment는 총 2개

xxxxxx_xxx_abusing_xxxxxxx_xx	abusing	적립취소 어뷰징 히스토리 아이디
xxxxxx_xxx_abusing_xxxxxxx_xx	abusing	사용취소 실패 히스토리 아이디

- 토큰별 개수를 바탕으로 도출한 comment 분석 비율

word	token	count	percentage
abusing	취소	2	100
abusing	히스토리	2	100
abusing	아이디	2	100
abusing	적립	1	50
abusing	어뷰징	1	50
abusing	사용	1	50
abusing	실패	1	50

→ **결과: "어뷰징"이 표준 단어가 되어야 하나, 통계상 단어 비율이 낮게 계산되어 신뢰성 저하**



예: '약어명'이 포함된 comment

- 예상 결과: "약어 + 명"
- 실제 결과: "약 + 어명"

Phase 2: 표준 용어 정의

표준 용어 생성 프로세스 및 원칙

표준 용어 생성 작업은 아래의 컬럼을 기준으로 관리되며, key(컬럼명+데이터타입)를 통해 표준화 대상을 식별하고 정의합니다.

- **작업 관리 기준 컬럼**
  - **key(컬럼명+데이터타입):** 표준화 대상의 고유 식별자
  - **표준화작업:** 그룹화된 컬럼명
  - **컬럼 코멘트, 데이터 타입, 데이터 길이:** 컬럼의 메타 정보
  - **건수:** 해당 표준화 대상이 전사 DB에서 사용된 횟수
  - **표준한글명, 분류어, 표준영문명, 도메인명, 단어수:** 표준화 결과물

2-1. 표준한글명 생성 방법: `건수`에 따른 사례별 접근

표준한글명을 정의하는 기준은 해당 **표준화 대상**이 전사 DB에서 **몇 번 사용**되었는지를 나타내는 **건수**입니다.

Case 1 (건수 多): 포괄적 용어 생성 → 여러 테이블에서 공통으로 사용되는 컬럼이므로, 포괄적인 대표 표준한글명을 생성

예시: 표준화 대상 **unit_type** (건수: 6)

1. 분석: 해당 컬럼이 포함된 모든 테이블의 정보(테이블명, 코멘트)를 종합적으로 참고

unit_type 컬럼 사용 정보 예시

table_name	table_comment	column_id	column_comment
xxxxxx_xxxxxxx_xxxxxxx	예산 제어 이력	unit_type	단위
xxxxxx_xxxxxxx	예산 제어	unit_type	단위
xxxxxx	한도	unit_type	한도 단위

2. 결론: 다양한 테이블의 용도를 포괄할 수 있도록 **단위_유형_코드**로 표준용어조합으로 정의

(원칙) 이때 사용되는 '단위', '유형', '코드'와 같은 한글 단어는 사전에 정의된 표준 단어 사전에서 가져와 매핑해야 합니다.

Case 2 (건수 1): 구체적 용어 생성 → 특정 테이블에서만 사용되는 컬럼이므로, 해당 테이블의 고유한 의미와 맥락에 맞게 표준한글명을 생성

예시: 표준화 대상 **all_yn** (건수: 1), 코멘트: 모두표시

1. 분석: 컬럼이 사용된 테이블의 정보(notice - 공지사항)를 참고합니다.

all_yn 컬럼 사용 정보 예시

table_name	table_comment
notice	공지사항

2. 결론: 테이블의 맥락을 구체적으로 반영하여 **고지_전체_여부**로 표준용어조합으로 정의합니다.

(원칙) 이때 사용되는 '고지', '전체', '여부'와 같은 한글 단어는 사전에 정의된 표준 단어 사전에서 가져와 매핑해야 합니다.

2-2. 주요 고려사항 및 해결 방안

문제점 1: 포괄적 용어 생성으로 인한 의미 불일치 발생

상황: 동일한 컬럼명과 데이터타입을 기준으로 용어를 통합하다 보니, 특정 DB에서는 컬럼의 실제 의미와 표준한글명이 달라지는 문제 발생.

해결 방안:

1. **1단계 (전체 표준화 우선):** 개별적으로 수정하면 표준화의 의미가 깨질 수 있기 때문에, 먼저 전체 컬럼에 대한 포괄적인 표준한글명 정의를 완료하여 일관된 기준을 세운다.
2. **2단계 (DB별 검토 및 수정):** 전체 표준이 확정된 후, DB별로 표준 용어를 재검토하여 특정 테이블에 더 적합한 표준한글명으로 수정.
    - **(원칙)** 단, 용어를 수정할 때는 사전에 정의된 '표준 용어 생성 규칙'을 반드시 준수하여 표준화가 깨지지 않도록 한다.

문제점 2: 메타데이터만으로는 정보가 불충분

상황: 테이블과 컬럼 코멘트만으로는 컬럼의 정확한 의미를 파악하기 어려운 경우가 많다.

해결 방안: 용어 생성 이후, 코드값을 조회하는 단계에서 각 컬럼의 실제 데이터(Data Value)를 반드시 함께 확인하면서 표준한글명을 지속적으로 검증하고 수정하는 보완 작업을 병행

Phase 3: 표준 코드값 정의

DB별 코드성 컬럼 데이터 추출

용어 표준화 결과(컬럼별 표준한글명) 중 **~_코드** 컬럼만 선별

필터링된 컬럼에 대해 다음 정보 저장

- 인스턴스명 / 테이블명 / ASIS 컬럼명

해당 DB에 **connect** → 각 컬럼별 데이터 값 조회 (= 코드값 조회)

코드인 경우, 코드가 아닌 경우에 따라 데이터 처리

코드 여부 판별

🚫 코드가 아닌 경우

- 값이 수천 개 이상 나열되어 코드 데이터로 보기 어려움
- 처리 방식 → ~_코드 → **~_값 / ~_내용** 등으로 표준 용어의 **분류어를 수정**하여 데이터 실제 성격 반영

코드 여부 판별

✅ 코드인 경우

① 숫자 코드

- 특징: 값이 숫자로 구성되고 종류 제한적
- 처리 방식:
  - 동일 테이블 내 코드값명 컬럼이 있는지 탐색 → 1:1 매핑
  - 대응되는 컬럼이 없는 경우 → 유관 부서에 의미 정의 요청 후, 데이터 갱신

- 예시: 컬럼명 **bank_corp_code**
- 결과:

인스턴스명	테이블명	컬럼명	표준용어	코드값	코드값명
INST_01	TB_01	bank_corp_code	은행_기업_코드	1	BC
INST_01	TB_01	bank_corp_code	은행_기업_코드	2	KB
…	…	…	…	…	…

② 문자 코드

- 특징: 값이 대문자 알파벳 등 문자로 구성되고 종류 제한적
- 처리 방식:
  - 숫자 코드와 동일한 방식으로 코드값명 컬럼 탐색 및 매핑
  - 의미 유추 불가 시 → 코드값명을 코드값으로 지정 후, 추후 보완

- 예시: 컬럼명 **status**
- 결과:

인스턴스명	테이블명	컬럼명	표준용어	코드값	코드값명
INST_02	TB_02	status	상태_코드	DEL	삭제
INST_02	TB_02	status	상태_코드	OK	OK
…	…	…	…	…	…

4. 협업 및 프로세스 개선 기여

데이터 표준화 프로젝트를 원활하게 수행할 수 있도록 팀의 효율성을 높이고,
통일성 있는 결과물을 만들어내기 위한 협업 방식과 작업 프로세스 개선에 적극적으로 기여했습니다.

Wiki 활용을 통한 회의 효율화

문제 (기존 상황)

- 반복되는 이슈 발생
- 회의 시간 낭비
- 이슈 공유 부족

제안

- 회의 전, 팀원별 발생 이슈를 위키 페이지에 사전 정리
- 작업 내용 + 진행 상황 기록 → 히스토리 관리 가능
- 사전 공유 → 회의 시간 효율적으로 활용

개선 결과

- 회의 전 이슈 파악 가능 → 회의 효율성 + 집중도 향상
- 회의 시 1차 공유 내용 기반 사수와 2차 확인 → 해결 / 미해결 사항 명확화
- 위키 기록 이슈 히스토리 → 추후 참고 및 활용 가능

작업 프로세스 개선

자동화 도입

- Python 로직 구현 → 전사 빈도 기반 한글명 자동 추천

질의응답 체계

- 슬랙 캔버스 활용 → 즉각 질의응답
- 위키 Q&A 페이지 구축 → 반복 질문 검색·재확인 가능

지식 자산화

- 날짜별 Q&A 기록 + 이슈 모음 페이지 생성 (용어 표준화 이슈 정리 페이지)
- Q&A 카테고리화 → 단어 생성 규칙·가이드라인 정의 근거 자료로 활용 가능

감사합니다