IT-Asset-Register

사내 IT 자산 대여·반납·검수·통합관리 프로세스 DB 설계

향후 계획(요약)

1) 기준 정리(동기화)
- 용어/상태값 통일(대여·반납·검수 등) + 문서/ERD/DDL/PPT 동일 적용
- 요구사항 ↔ 테이블/컬럼 매핑표로 누락·불일치(감점 포인트) 선제 차단

2) 설계 고도화(무결성 우선)
- ERD 최종 확정 → DDL(FK/NOT NULL/UNIQUE) 반영 → 버전 태그로 동기화
- 상태전이 규칙 최소 보장(예: 반납 후 검수 완료 필수) 범위 확정

3) DB 로직 정리(업무단위)
- 프로시저 중심으로 흐름 고정(대여요청/반납요청/검수완료/제재 처리)
- 트리거는 로그·상태 자동처리 등 최소만 유지(남발 금지)

4) 테스트/성과 정리
- 시나리오 기반 테스트케이스(정상/예외/경계) 고정 + 결과 캡처 확보
- 오류 재현→원인→수정→재검증 흐름을 성과 파트로 문서화

5) 문서/발표 정리
- README 흐름: 프로세스→ERD→DDL/로직→테스트/성과 순으로 재배치
- 폴더/이미지 경로 표준화로 링크 깨짐 방지

6) 향후 확장 계획(B2B 재도입 옵션)
- 현재는 범위를 줄이기 위해 B2B(기업/지점/거래처 단위 운영) 제거
- 추후 확장 시:
  - 테넌트(회사) / 지점 / 조직(부서) 스키마 추가
  - 자산 소유/배정 단위를 "회사→지점→부서"로 확장
  - 권한 모델(관리자/담당자/사용자)과 데이터 분리(회사별 접근 제한) 적용
  - 통합 리포트(회사별/지점별 자산 현황, 대여율, 미반납, 검수 지연)까지 확장

B2B방식 도입

6.1 재도입 목적/범위(현재 대비 확장 포인트)

- 현재(단일 조직/단일 회사 기준): 사내 자산을 한 조직 단위로 대여·반납·검수·제재 관리
- B2B 확장(다중 회사/다중 지점/다중 조직): 여러 기업(테넌트)이 동일 시스템을 쓰되,
  회사별 데이터 분리 + 회사별 정책/권한/프로세스 커스터마이징을 지원

6.2 핵심 개념(멀티테넌시) 설계 방향

- 테넌트(Company/Tenant): "고객사" 단위 최상위 구분 키
- 조직(OrgUnit): 본사/지점/부서 계층(회사 내부 구조)
- 권한(Role): 회사 관리자 / 지점 관리자 / 자산 담당자 / 일반 사용자
- 데이터 분리 방식(중요)
  1) 단일 DB + tenant_id 컬럼 분리(권장: 학습/프로젝트 규모에 적합)
  2) 회사별 스키마 분리(운영 복잡도 증가)
  3) 회사별 DB 분리(보안은 강하나 비용/관리 증가)

6.3 ERD/테이블 확장(추가/수정 항목)

- 신규 테이블(예시)
  - company(tenant): 회사 기본정보, 계약상태, 사용여부
  - branch: 지점 정보(회사 FK)
  - org_unit: 부서/팀(회사 FK, 상위 org_unit FK로 트리 구조)
  - role / user_role: 회사별 역할 정의 및 사용자 매핑
  - policy: 회사별 정책(대여기간, 제재 기준, 승인 필요 여부 등) 파라미터화
  - contract/plan(선택): 요금제/사용량/계약기간(프로젝트 범위에 따라 제외 가능)

- 기존 테이블 수정(필수)
  - asset, user, lend, return, inspect, sanction, audit_log 등에 tenant_id(회사 FK) 추가
  - 자산 귀속 단위 추가: asset.owner_branch_id / asset.owner_org_id 등(선택)
  - UNIQUE 재설계:
    - 자산번호/시리얼 UNIQUE는 (tenant_id + asset_no) 복합 유니크로 변경
    - 사용자 사번/아이디도 (tenant_id + employee_no) 형태로 충돌 방지

6.4 권한/접근 제어(감점·운영 리스크 큰 구간)

- 기본 원칙: "자기 회사 데이터만 조회/수정"
- 역할별 권한 예시
  - 회사 관리자: 회사 전체 자산/사용자/정책/리포트 관리
  - 지점 관리자: 해당 지점 자산/대여현황/검수현황 관리
  - 자산 담당자: 자산 등록/검수/제재 처리
  - 일반 사용자: 대여요청/반납요청/내 기록 조회
- DB 관점 최소 보장
  - 모든 주요 테이블에 tenant_id 강제(FK + NOT NULL)
  - 프로시저 입력값에 tenant_id 포함 후, 내부에서 "tenant_id 일치" 검증
  - audit_log에 tenant_id + actor(행위자) 기록으로 추적성 확보

6.5 회사별 정책/프로세스 차이(파라미터화 전략)
- 회사마다 다른 항목을 "테이블 파라미터"로 흡수(하드코딩 금지)
  - 대여 최대 기간/연장 횟수
  - 승인 필요 여부(예: 고가 장비는 승인 필수)
  - 반납 후 검수 SLA(예: 24시간 내 검수)
  - 제재 규칙(지연 일수별 단계, 누적 경고 기준)
- 구현 방식
  - policy 테이블에 key-value 형태로 저장하거나(유연)
  - 컬럼화(명확/단순)해서 저장(프로젝트 규모엔 컬럼화가 보통 안정적)

6.6 B2B 리포트/통합 운영 기능(확장 시 "성과"로 보이기 좋은 영역)
- 회사별/지점별 자산 현황(보유/대여중/검수대기/수리중 등 상태별)
- 대여율/회전율(기간별), 미반납 리스트, 검수 지연 리스트(SLA 위반)
- 제재/경고 누적 상위 사용자/부서
- 감사로그 기반 "누가/언제/무엇을" 변경했는지 추적 리포트

6.7 확장 시 테스트 포인트(반례 중심)
- 다른 회사(tenant_id 다름) 자산을 조회/대여 시도 → 반드시 차단
- 동일 asset_no가 서로 다른 회사에 존재 → 복합 UNIQUE로 허용, 단일 UNIQUE면 오류
- 지점 관리자가 타 지점 자산을 변경 시도 → 권한/스코프 검증 필요
- 회사별 정책이 다를 때(대여기간/제재기준) 프로시저가 정책을 올바르게 적용하는지
- 통합 리포트가 tenant 기준으로 집계되며 혼합되지 않는지

6.8 단계적 도입(현실적인 로드맵)
- 1단계: tenant_id 컬럼 추가 + 주요 쿼리/프로시저 tenant 스코프 적용(최소 멀티테넌트)
- 2단계: branch/org_unit/role 적용(권한/조직 확장)
- 3단계: policy 기반 커스터마이징(회사별 규칙 차등)
- 4단계: 리포트/대시보드(운영 가치 극대화)

6.9 발표/문서에 넣을 포인트(왜 뺐고, 어떻게 확장하는지)
- "현재는 단일 조직 기준으로 범위를 축소해 핵심 프로세스(대여-반납-검수-제재) 완성도에 집중"
- "향후 B2B 재도입 시 멀티테넌시(tenant_id) + 권한/조직/정책 파라미터화로 확장 가능"
- "확장 시에도 기존 코어 테이블 구조를 유지하면서 FK/복합 UNIQUE/정책 테이블로 안전하게 확장"